Computer Networks 102 (2016) 96–108

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Decentralized RFID coverage algorithms using writeable tags

Ahmed Jedda*, Mazen G. Khair, Hussein T. Mouftah

University of Ottawa, Ottawa, Canada

ARTICLE INFO

Article history:
Received 15 March 2015
Revised 3 March 2016
Accepted 17 March 2016
Available online 19 March 2016

Keywords:
RFID
RFID coverage
Topology construction
Distributed algorithms

ABSTRACT

A Radio Frequency IDentification (RFID) reader network is as a collaboration of RFID readers that aim to cover (i.e., identify, monitor, and track) every RFID tag in a given area. The RFID coverage (RFC) problem is defined as follows. Given a reader network, assign to each tag t a specific reader v in its proximity such that v is responsible for covering t (called its owner), while minimizing the number of owner readers. The problem has applications in energy conservation and in eliminating readers and data redundancy from the reader networks. We introduce a number of decentralized algorithms for the RFID coverage problem: 1) algorithms RANDOM, RANDOM$^+$, and MAX–MIN which are randomized algorithms that run in $O(1)$ write/read rounds, 2) algorithm GDE which is an efficient decentralized implementation of the greedy set cover algorithm, and 3) an improvement of GDE which is called . Our algorithms assume that the RFID tags are writeable, where a writeable tag is a passive RFID tag with writeable memory. We show using simulation experiments that our algorithms outperform major RFID coverage algorithms in various scenarios with respect to a number of performance metrics.

1. Introduction

A Radio Frequency IDentification (RFID) system generally consists of an RFID *reader* and an RFID *tag*. A reader sends a radio signal to the tag. Upon the reception of the reader's signal, the tag replies to the reader with a signal that contains the tag identifier and other parameters about the tag if possible. This working principle allows RFID systems to be used for identifying, tracking, and monitoring physical objects by simply attaching tags to them. Advances in hardware manufacturing led to significant improvements in the cost, size and performance of RFID systems. As a result, the use of RFID systems became an economically feasible option for many applications. For instance, RFID is notably used in the logistics, defence, aerospace, health and pharmaceutical sectors.

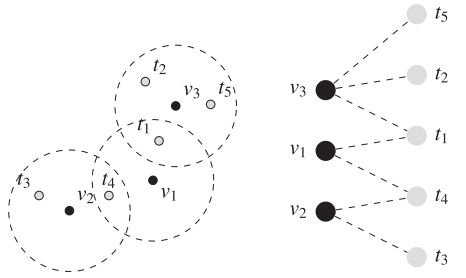
A main factor contributing to the recent widespread use of RFID is the low cost and high performance of RFID *passive tags*. A passive tag consists of an embedded circuit, a memory, and a transceiver, but no battery as it is empowered by the energy of the signals received by readers in its proximity. Its size can be in the orders of millimetres [1] allowing it to be attached to various objects. Some types of passive tags, called *writeable tags*, contain writeable memory [2]. Readers in proximity may write in the memory of writeable tags by sending radio signals. We focus in this paper on RFID systems that consist of passive writeable tags. There are other types of RFID tags that contain batteries. Some of these tags are allowed to initiate communication with the readers, and hence called *active tags*, while some others do not have this feature, and hence called *semi-passive tags*. These types of tags are of higher cost compared to passive tags, and thus have limited applicability.

The large scale of RFID systems is foreseen due to the low cost and small size of RFID tags and due to the large number of RFID applications. The main drivers of such networks are: 1) the Internet of Things (IoT), where every identifiable physical object (or, a *thing*) is expected to be connected to the Internet by attaching RFID tags or other uniquely identifiable tags, and 2) large supply chains such as those of the US Department of Defence, WalMart, Toyota, and others. The main problem in large scale RFID systems is the coverage of tags (i.e. identifying, monitoring, and tracking). The basic approach to overcome this problem is the use of collaborations of readers, called *reader network*, in order to cover all tags in a given area. Each reader in a reader network is responsible for covering a subset of the tags and report its readings to a special server that collects and processes the data gathered by all readers. An example of a reader network that consists of three readers and five tags is illustrated in Fig. 1(a). The coverage relationships between readers and tags are usually modeled as a bipartite graph as shown in Fig. 1(b).

We study the problem of optimizing the energy consumption of a reader network by eliminating unnecessary redundancy at the

* Corresponding author. Tel.: +15147062637.
E-mail addresses: ahmed.jedda@uottawa.ca (A. Jedda), mkhair@site.uottawa.ca (M.G. Khair), mouftah@uottawa.ca (H.T. Mouftah).

http://dx.doi.org/10.1016/j.comnet.2016.03.003
1389-1286/



(a) A sample reader network consisting of three readers (black dots) and five tags (gray dots). The dashed circles represent the covering range of readers.

(b) A bipartite graph model for the reader network on the left. A reader is linked to a tag if it covers it.

Fig. 1. Sample reader network illustrating readers redundancy.

readers level. The problem we consider is called the *RFID coverage* (RFC) problem. There are two objectives of the RFID coverage problem. The first is to assign each tag to a reader in its proximity called its *owner*. The owner of a tag is the only reader in the network that is responsible for reporting readings about the tag. The second objective of the RFID coverage problem is to minimize the set of owners in a given reader network (also called *non-redundant readers*).

A solution to the RFID coverage problem eliminates two types of redundancies; 1) data redundancy and 2) readers redundancy. *Data redundancy* occurs in situations where two readers or more report the same readings about the same tag. This type of redundancy a) causes problems in processing and mining the data generated by a reader network [3], and b) causes an increase in network traffic. Eliminating data redundancy can be done by assigning to each tag an owner reader, since only the owner of a tag is allowed to report readings about it. Note that a tag does not need to know which reader owns it, however, a reader must be aware of the tags it owns. This subproblem of the RFID coverage problem is called the *tag reporting* problem. *Readers redundancy* occurs if a tag or more in the reader network is covered by more than one reader. Minimizing the number of readers in a given reader network, while preserving the network coverage, improves the network energy consumption. This subproblem is called the *redundant readers elimination* problem. As an example, consider the reader networks of Fig. 1(a). We can assign v_2 as the owner of t_4 and t_3 and v_3 as the owner of t_1, t_2 and t_5. Reader v_1 therefore can be switched off. The negative impact of data and readers redundancies on reader networks become clearer as the reader network increases in scale.

The RFID coverage problem is similar to some variants of the sensor coverage problem [4,5]. It was introduced in [6] under the names of the *tag reporting* problem and the *redundant readers elimination* problem where it is assumed that the only means of communication to solve the problem is *reader-tag communications*. Herein, the readers cannot directly exchange messages, but they are allowed to write and read the memory contents of the tags in proximity using what is called *write/read rounds*. This model was later used in [7–11], and others. Another version of the RFID coverage problem, introduced in [12], does not allow the use of writeable tags, but allows direct message exchange between the readers using wireless communications. We focus on the first type of RFID coverage; the *reader-tag RFID coverage* problem.

Write/read rounds. A basic component in reader-tag RFID coverage algorithms[1] are *write/read rounds*. A randomized implementation of write/read rounds was introduced in [6]. Abstractly, a write/read round consists of two phases; write phase and read phase. In the write phase, every reader v writes a set of bits, called the *weight* of v and denoted by $\mathcal{W}(v)$, in the memory of all (or some) neighbor tags (i.e., tags that are covered by v). The readers wait for a specific period of time to allow every reader to do the same. In the read phase, the readers read the content of the memory of neighbor tags. At that time, the memory of a tag t contains the weights of all the neighbor readers of t that wrote in it during the write phase. More details on write/read rounds are given in Section 2.2.

Contributions. An RFID coverage algorithm is evaluated by the number of non-redundant readers it generates and the number of write/read rounds it executes. Many existing algorithms aim to achieve this objective by using a single write/read round but with different, sometimes sophisticated, definitions of the reader weights. A tag t is owned by the neighbour reader v that has the maximum weight $\mathcal{W}(v)$. In our *first set of contributions*, we set $\mathcal{W}(v)$ for every reader v to be a random number combined with the unique identifier of v. This introduces a simple single write/read round algorithm called `RANDOM`, which should be considered as a benchmark to similar algorithms due to its simplicity. Nevertheless, the simulation experiments in Section 7 show that `RANDOM` outperforms similar algorithms in practical scenarios. We also introduce algorithm `RANDOM`$^+$ and `MAX-MIN`, which both further improve the performance of `RANDOM` using additional write/read rounds, where each round is ran with a new randomly generated weight. These algorithms are shown to generate a low number of non-redundant readers with the cheap cost of one additional round (or few more).

The *second set of contributions* consists of two algorithms. The first is called the Greedy Decentralized Elimination (`GDE`) algorithm. It is the first decentralized algorithm that gives the same result of the centralized greedy set-cover algorithm. This algorithm generates the least number of non-redundant readers compared to existing RFID coverage algorithms. However, `GDE` runs in at most $|\mathcal{R}|$ iterations, where $\mathcal{R}$ is the set of readers. Each iteration consists of two write/read rounds. To improve `GDE` write/read complexity, we introduce `LIMITED-GDE` which limits the number of write/read rounds to $O(1)$ while keeping the number of non-redundant readers within an acceptable level that is still better than many other major algorithms.

The RFID coverage problem may appear with additional constraints, such as multihop communication connectivity between readers, k-coverage for improved fault-tolerance [13], handling faulty communication links, or achieving load balancing between readers. None of these constraints are considered in this paper because:

1. There is still room for improvements in the unconstrained version of the RFC problem as will be shown later, and
2. Studying the problem without constraints provides a better understanding of it, which helps later in a better understanding of its constrained versions.

Paper organization. Section 2 gives a survey of related work. Section 3 formalizes the problem and the mathematical model used. Algorithms `RANDOM`, `RANDOM`$^+$, `MAX-MIN` are described in Section 4. `GDE`, and `LIMITED-GDE` are described in sections 5 and 6 respectively. Each algorithm is given with a theoretical proof of correctness and complexity analysis. In Section 7 we use simulation experiments to study the empirical performance of our algorithms. Section 8 concludes the paper.

[1] We use the term RFID coverage in this paper to denote reader-tag RFID coverage.

2. Related work

RFID Coverage algorithms are categorized into *centralized* and *decentralized* algorithms. More details about these categories are given below.

2.1. Centralized algorithms

Centralized RFC algorithms assume the existence of a programmable centralized node, which can be a dedicated server or an elected reader. Practically, even if a centralized dedicated server is available, it may not be possible to program this server for purposes other than what it was designed for. Other than that, the use of centralized nodes degrades the scalability of the network, increases its cost, and may lead to inefficient use of available resources. In these algorithms, each reader is connected to the centralized node. Most centralized RFC algorithms do not specify how such connections are made. Each reader v sends its neighbour tags set $N_T(v)$ to the centralized node (a tag t is a neighbor tag of reader v if v can cover it. A detailed definition is given in Section 3). The centralized node builds a complete view of the network, and executes a sequential algorithm that assigns to each reader the tags it owns.

Centralized algorithms reduce the RFC problem to the *unit-cost set cover* problem (or the set cover problem for short), defined as follows. Given a collection of sets $\mathcal{S} = \{S_1, \ldots, S_k\}$ and a universe of elements $U = \{e_1, \ldots, e_n\}$, find a subcollection $\mathcal{S}' \subseteq \mathcal{S}$ such that $\bigcup_{S_i \in \mathcal{S}'} S_i = U$. A minimum sized set cover is called an *optimal set cover*. The problem of finding an optimal set cover is known to be **NP Hard** [14]. A solution of the set cover problem can be used to solve the RFC problem. It is sufficient to set U as $\mathcal{T}$ (where $\mathcal{T}$ is the set of all tags in the network), and set $\mathcal{S}$ as the collection of sets $\{N_T(v) \mid v \in \mathcal{R}\}$ (that is, the collection of tag neighbors sets for all readers).

The Centralized Greedy Approach:. Algorithm GREEDY [6,15] is a centralized algorithm that uses the reduction given above. The algorithm is equivalent to a well-known greedy set cover algorithm, which is referred to in the following as the *standard greedy set cover algorithm*. This algorithm runs in iterations. The set (i.e., the reader) that covers the maximum number of not-yet covered elements (i.e., tags) is included in the solution set $\mathcal{C}$ in each iteration. This procedure continues until all the tags are covered. A reader included in the solution $\mathcal{C}$ owns all its neighbors tags that are not already in $\mathcal{C}$ (i.e., not-yet owned tags). This guarantees that every tag is owned by exactly one reader. GREEDY is frequently used in this paper given its important properties. For instance, GREEDY is known to have a $O(\log n)$ approximation ratio of the set cover problem, where n is the number of elements in the universe (or, the number of tags in this context). [2] Another similar greedy algorithm is NTE [16]. The main difference between GREEDY and NTE is how the maximum node is defined in each iteration. In NTE, the maximum node is the node that has the maximum number of coverage neighbor readers[3] which are also running the same iteration.

2.2. Decentralized algorithms

Decentralized reader-tag RFC algorithms use write/read rounds to accomplish their objectives. Carbunar et al. introduced in [6] algorithm RRE (Redundant Readers Elimination), which requires only one write/read round. The algorithm consists of a single write/read round. In the write phase, a reader v sets its weight $\mathcal{W}(v)$ to $(|N_T(v)|, \mathbf{id}(v))$, where $N_T(v)$ is the set of neighbor tags of v, $\mathbf{id}(v)$ is the unique identifier of v and $|.|$ is the size of a set. The weight of v is written in the memory of all neighbor tags of v. The readers read the memories of their neighbor tags. A reader v is able then to decide whether it is an owner of a tag t by checking if it has the maximum weight $\mathcal{W}(v)$ among all the readers that wrote in the memory of t. The comparison among the readers weights is done in lexicographical order. That is, the weight of reader v_i is larger than the weight of reader v_j, denoted by $\mathcal{W}(v_i) \succ \mathcal{W}(v_j)$, if v_i has a larger number of neighbor tags $|N_T(v_i)|$. In case both readers have the same number of neighbor tags, then $\mathcal{W}(v_i) \succ \mathcal{W}(v_j)$ if v_i has a larger identifier (see Fig. 2(a) for an example of RRE).

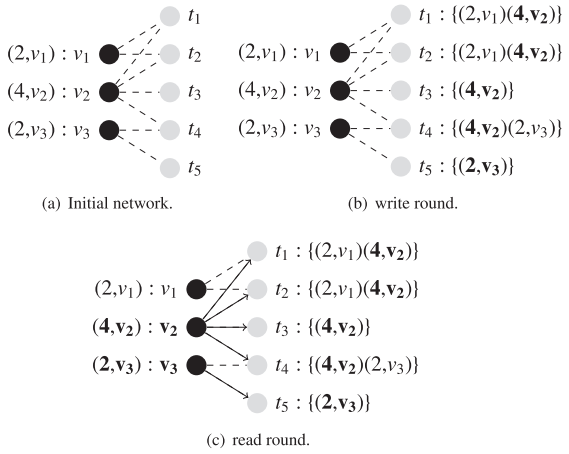


Fig. 2. Example illustrating a write/read round in algorithm RRE. The pairs beside the readers are the weights of the readers. We assume that $\mathbf{id}(v_3) \succ \mathbf{id}(v_2) \succ \mathbf{id}(v_1)$ according to some total order. For simplicity, we assume $\mathbf{id}(v) = v$. The sets beside the tags are the memory contents of the tags after the write round. Readers v_2 and v_3 are the non-redundant readers selected by the algorithm.

A set of algorithms similar to RRE were later introduced, each changing how the weight of a reader is defined. Algorithm DRRE [7] (Density-based Redundant Readers Elimination) sets the reader weight v to $\mathcal{W}(v) = (|N_{R_s}(v)|, \mathbf{id}(v))$ where $N_{R_s}(v)$ are the readers that share at least one common neighbor tag with v. Algorithm LEO [8] assumes that a reader owns a tag if it is the first to write on it. Irfan and Yagoub introduced in [9] a more sophisticated definition of $\mathcal{W}(v)$ that requires the involvement of all nodes to be computed.

Executing RFC algorithms in Sequence:. To reduce the number of non-redundant readers, Hsu et al. introduced in [8] algorithm LEO+RRE which consists of two write/read rounds. In the first round, LEO [8] is executed and generates a set of non-redundant readers $\mathcal{R}_1$. In the second round, RRE [6] is executed over the set $\mathcal{R}_1$ instead of $\mathcal{R}$. This step is expected to reduce the number of non-redundant readers furthermore. The simulation experiments results of [8] show that LEO+RRE outperforms LEO and RRE. The main advantage of LEO+RRE is the introduction of the novel approach of combining RFC algorithms (that is, executing a sequence of RFC algorithms to reduce the number of non-redundant readers). A similar algorithm to LEO+RRE is introduced in [17] where the first round is an execution of LEO [8] and the second round is an execution of the RFC algorithm introduced in [9].

More constraints:. Two randomized algorithms are introduced in [10] with the objective of balancing the coverage load among non-redundant readers. The first algorithm runs in two write/read rounds. The second algorithm runs in multiple write/read rounds.

[2] An algorithm **A** is said to have a ρ-approximation ratio for a given problem **P** if it guarantees that any solution it outputs for **P** has a cost within $\rho \times OPT$ in the worst case, where *OPT* is the optimal cost of solving **P**.

[3] A reader v is said to be a *coverage neighbor* of a reader u if both readers are neighbors to at least one common tag t.

The load balancing performance of the solution of the second algorithm is improved as the number of executed write/read rounds increases. Algorithm RANDOM$^+$, introduced in Section 4, uses a similar approach to improve its quality. Dhas et al. introduced in [11] an interesting load balancing RFC algorithm that takes into consideration tags mobility. Each reader writes a time-stamp in its neighbor tags. The reader writes other weights as well. The timestamp is rewritten continuously. The readers use their neighbor tags timestamps to change the ownership of a tag. Some factors that lead to ownership changes in a reader network are failure of a non-redundant reader or tag mobility (that is, a tag changed its location such that it is not covered by its owner any more).

Why not using the reader-reader communication model instead?. The reader-reader communication model is neither equivalent to, nor more powerful, than the reader-tag model. It is shown in [12] that the RFID coverage problem requires $\Omega(|\mathcal{R}|\log|\mathcal{R}|)$ exchanged messages to be solved in the reader-reader model in the general case. This an expensive cost compared to the single write/read round required to solve the problem in the reader-tag model. This is caused by the fact that a pair of readers u and v may cover the same tag t but cannot necessarily exchange messages using a direct link (i.e., u and v are not neighbors). Ultimately, our objective is to combine both models. However, we think that in order to achieve this objective we will need to understand both models rigorously (for example, by studying the problem under more constraints such as fading, interference). There is still room for improvements for the non-constrained version of this problem as it is shown in this paper.

3. Problem formulation

A *reader network* consists of a set of readers $\mathcal{R}$ and a set of tags $\mathcal{T}$. A reader v is said to *cover* a tag t if v can read the memory content of t, and write in the memory of t, denoted $\mathcal{M}(t)$, if t is *writeable*. The coverage relationships are modelled as a bipartite graph $\mathcal{G}_B = (\mathcal{R}, \mathcal{T}, E_s)$, where E_s is a set of *coverage edges* (or, *coverage relationships*). The set of *neighbor tags* of a reader v, denoted $N_T(v)$, is defined as the set of tags covered by v (that is, $N_T(v) = \{t \mid (v,t) \in E_s\}$). The set of *neighbor readers* of a tag t, denoted $N_R(t)$, is defined as the set of readers that cover t (that is, $N_R(t) = \{v \mid (v,t) \in E_s\}$). A reader v is said to be a *coverage neighbor* of a reader u if both readers cover at least one common tag t (that is, if $N_T(v) \cap N_T(u) \neq \emptyset$). The set of all coverage neighbors of a reader v is denoted by $N_{R_s}(v)$. We assume that an edge (v,t) is in E_s if the Euclidean distance between v and t, denoted $d(v,t)$, is within a constant distance r_i (called the *interrogation range* and is common to all readers and tags). We assume that:

1. There are no communication links between the readers.
2. The readers are assumed to be able to write in the memory contents of their neighbor tags. However, a tag cannot transmit any message to a reader without first being interrogated by that reader.
3. The readers have no previous knowledge of the reader network topology, have no knowledge of their positions, nor their neighbor tags positions.
4. There are no centralized nodes.
5. Each reader v is assumed to have a unique comparable identifier, denoted $\mathbf{id}(v)$. Each tag t is assumed to have a unique identifier, denoted $\mathbf{id}(t)$.[4]

Problem definition:. a reader v *owns* a tag t if v is delegated to read tag t. The set of tags owned by a reader v is denoted by $S(v)$ (i.e., *slaves* of v). The readers that own at least one tag are called *non-redundant readers* and denoted by $\mathcal{C}$ (that is, $\mathcal{C} = \{v \mid S(v) \neq \emptyset\}$). If every tag is covered by at least one reader in $\mathcal{C}$, then $\mathcal{C}$ is called a *cover*.

The RFID coverage (RFC) problem consists of the following subproblems:

1. **Tag reporting**: for each reader v, find a set $S(v) \subseteq N_T(v)$ such that $\bigcup_{v\in R} S(v) = \mathcal{T}$ and $\bigcap_{v\in\mathcal{R}} S(v) = \emptyset$ (pairwise disjoint). That is, each tag has exactly one owner.
2. **Redundant readers elimination**: minimize the size of the set $\mathcal{C}$. That is, minimize the number of non-redundant readers in the network.

Minimizing the number of non-redundant readers turns the problem into an **NP Hard** problem, as the minimum disk coverage problem may be reduced from it [6]. This leads to the following definition.

Definition 1. An algorithm **A** solves the the RFC problem if **A** *correctly* solves the *tag reporting* problem[5].

Lastly, the main terminologies used in this paper are summarized in Table 1.

Table 1
Terminologies used in this paper.

Term	Definition
$\mathcal{R}$	The set of readers
$\mathcal{T}$	The set of tags
$N_T(v)$	The set of neighbor tags of reader v
$N_R(t)$	The set of neighbor readers of tag t
$N_{R_s}(v)$	The set of coverage neighbor readers of v
$S(v)$	The set of tags owned by reader v
$\mathbf{id}(v)$	The unique identifier of reader v
$\mathcal{C}$	The set of readers which owns at least one tag (non-redundant readers)
$\mathcal{M}(t)$	The memory content of tag t
r_i	The interrogation range of the readers

4. Algorithms RANDOM, RANDOM$^+$, and MAX–MIN

This section introduces the decentralized algorithms RANDOM and RANDOM$^+$, which are randomized decentralized RFC algorithms that use reader-tag communications. RANDOM consists of a single write/read round, whereas RANDOM$^+$ consists of ψ iterations, where ψ is a constant greater or equal than 1. Each iteration of RANDOM$^+$ is an execution of algorithm RANDOM over the set of non-redundant readers generated in the previous iteration. A variant of RANDOM$^+$ is MAX–MIN, which runs in two iterations only. Before introducing the details of our algorithms, we present a generalized sequential single-round RFC algorithm, called SEQ, that will help in providing the motivation and analysing the algorithms of this section. We also give an efficient decentralized version of SEQ.

4.1. SEQ: generalized sequential single-round RFC algorithm

Single-round decentralized RFC algorithms, such as RRE and DRRE, can be described as the following sequential algorithm. First, the readers $\mathcal{R}$ are sorted by a *ranking function* $\pi(\mathcal{R})$, where the output of $\pi(\mathcal{R})$ is the set $\{v_{(1)}, \ldots, v_{(k)}\}$ and $v_{(i)}$ is the reader with the *ith* rank according to $\pi(\mathcal{R})$. For instance, $\pi(\mathcal{R})$ may order the readers according to their weights. That is, a reader

[4] Such feature can be guaranteed by the EPCglobal standards in [18].

[5] This is because of the requirement of eliminating all redundant readers turns the problem into an **NP Hard** problem

Algorithm 1 Algorithm SEQ.

1: $\mathcal{C} \leftarrow \emptyset$
2: $U \leftarrow \mathcal{T}$
3: sort $\mathcal{R}$ according to a ranking function $\pi(\mathcal{R})$, as $v_{(1)}, \ldots, v_{(k)}$
4: **for** i : 1 to k **do**
5: **if** $\{N_T(v_i) \cap U\} \neq \emptyset$ (that is, if $v_{(i)}$ covers at least one not-yet covered tag) **then**
6: $\mathcal{C} \leftarrow \{\mathcal{C} \cup v_{(i)}\}$
7: $U \leftarrow \{U \backslash N_T(v_i)\}$
8: **return** $\mathcal{C}$

Algorithm 2 Algorithm RANDOM at reader v.

1: $\mathcal{W}(v) \leftarrow (\alpha(v), \mathbf{id}(v))$;
2: **for** each $t \in N_T(v)$ **do**
3: write(v,t, $\mathcal{W}(v)$);
4: **for** each $t \in N_T(v)$ **do**
5: read(v, t);
6: **if** $\mathcal{W}(v)$ is maximum in $\mathcal{M}(t)$ **then**
7: v owns t
8: **if** v does not own any tag **then**
9: v is redundant;

$v_{(i)} \succ v_{(j)}$ if $\mathcal{W}(v_{(i)}) \succ \mathcal{W}(v_{(j)})$. As an example, a reader $v_{(i)} \succ v_{(j)}$ if $(|N_T(v_i)|, \mathbf{id}(v_i)) \succ (|N_T(v_j)|, \mathbf{id}(v_j))$ according to the ranking function of RRE. The sequential algorithm, called SEQ, passes the readers in order from the highest ranked to the lowest. A reader is included in the solution $\mathcal{C}$ if it covers at least one not-yet covered tag. The algorithm continues until $\mathcal{C}$ is a cover. A reader not included in $\mathcal{C}$ is redundant. The pseudocode of SEQ is given in Algorithm 1.

There is a strong relationship between SEQ and GREEDY. Assume that the readers weights in SEQ are the same weights used in GREEDY. Both algorithms are greedy and terminate as soon as a cover $\mathcal{C}$ is created. GREEDY sorts the readers at each iteration and selects the reader with maximum weight in each iteration (i.e., the reader with maximum updated weight after eliminating all readers already included in $\mathcal{C}$). SEQ, on the other hand, sorts the readers only once. Such continuous re-sorting performed by GREEDY decreases the size of the cover $\mathcal{C}$. The problem of this approach is that it is expensive to implement by a decentralized algorithm. SEQ follows the other extreme of this approach, which is sorting the readers only once. This does not guarantee that at every iteration the reader with maximum weight is chosen, but it requires less computation. An approach that comes in between both extremes is to select readers randomly at each iteration. This approach, as will be shown next, is easy to implement by a decentralized algorithm and it generates covers with small sizes in several practical scenarios. This approach is implemented by algorithm RANDOM, described in the following.

The sequential version of algorithm RANDOM assumes that the ranking function $\pi(\mathcal{R})$ shuffles the readers according to a uniform probability. That is, a reader $v_{(i)}$ is ranked as the i^{th} reader with a uniform probability $\frac{1}{|\mathcal{R}|}$. Therefore, algorithm RANDOM selects, in each iteration, a random reader v with uniform probability and inserts it into $\mathcal{C}$ if it covers at least one not-yet covered tag. This procedure is repeated until $\mathcal{C}$ is a cover.

RANDOM as a benchmark:. Given the simplicity of its approach and its ranking function, RANDOM can be considered as a benchmark for other reader-tag RFC algorithms that aim to reduce the number of non-redundant readers. To outperform RANDOM, some reader-tag RFC algorithms apply simple heuristics which can be seen as different definitions of ranking functions $\pi(\mathcal{R})$. For instance, RRE orders the readers according to the number of their neighbor tags, whereas DRRE orders the readers according to the number of their coverage neighbor readers.

Another approach to outperform RANDOM is to run multiple executions of SEQ but with a different ranking of readers in each execution. This approach was used in [8] and [17]. Algorithm RANDOM$^+$ follows this approach as well. It runs in ψ iterations, for $\psi \geq 1$. A new cover set $\mathcal{C}_i$ is generated in each iteration i, for $1 \leq i \leq \psi$, by running algorithm RANDOM over the set $\mathcal{C}_{i-1}$, where $\mathcal{C}_0 = \mathcal{R}$. As a result, $\mathcal{C}_1$ is equivalent to the set of non-redundant readers generated by RANDOM. An execution of a RANDOM in a RANDOM$^+$ iteration is called a *shuffling* of $\mathcal{R}$. This is because in each RANDOM$^+$ iteration the readers are rearranged in a different random order. The shuffling procedure decreases the number of non-redundant readers generated by RANDOM. This improvement can be significant in practice as shown in the simulation experiments results in Section 7.

The intuition behind the shuffling of the readers in RANDOM$^+$ is the following. Consider an execution of algorithm RANDOM that generates the set $\mathcal{C}$ with size m. According to the sequential version of RANDOM (line 5 in Algorithm 1), a reader is included in $\mathcal{C}$ only if it covers at least one tag that is not yet covered. This means that the last reader to be included in $\mathcal{C}$, denoted by $v_{(m')}$ and $m' \geq m$ [6], covers at least one tag, denoted $t_{(n)}$, that is not covered by any other reader in $\mathcal{C}$. On the other hand, it is possible that a reader $v' \in \{\mathcal{C} \setminus v_{(m')}\}$ has all its tags covered by other readers in $\{\mathcal{C} \setminus v'\}$. Therefore, v' may be considered redundant if the readers are shuffled in a new RANDOM$^+$ iteration. The shuffling procedure reduces the size of $\mathcal{C}$ by at least one if 1) there is a reader v' that has all its tag neighbors covered by other readers, and 2) v' is ranked by the ranking function as the last ranked reader after the shuffling procedure.

4.2. Decentralized implementation of *RANDOM* and *RANDOM$^+$*

The next step is to implement RANDOM and RANDOM$^+$ on an RFID reader network in a decentralized manner with minimum amount of write/read rounds. To emulate the random selections of readers in RANDOM, each reader v draws a random number $\alpha(v)$ from a uniform distribution. Each reader v writes $(\alpha(v), \mathbf{id}(v))$ in $\mathcal{M}(t)$ for each neighbor tag t. The unique identifier $\mathbf{id}(v)$ guarantees the uniqueness of the readers weights. Each tag is owned by the neighbor reader v with the maximum pair $(\alpha(v), \mathbf{id}(v))$. This gives an emulation of SEQ with the ranking function $\pi(\mathcal{R})$ of RANDOM. This is a decentralized implementation of RANDOM in RFID reader networks that is executed in a single write/read round.

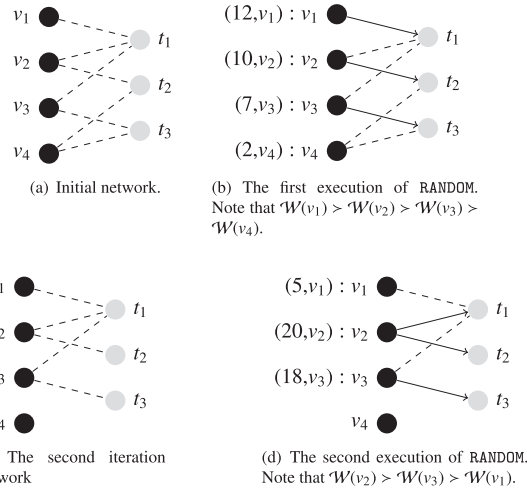
To implement the *ith* iteration of RANDOM$^+$ for $i \geq 1$, every reader v that finds itself non-redundant in the previous iteration $i-1$ draws a new random number $\alpha_i(v)$ from a uniform distribution and creates the pair $(\alpha_i(v), \mathbf{id}(v))$. A tag t is owned by the reader v with maximum $(\alpha_i(v), \mathbf{id}(v))$, where $v \in N_R(t)$. Note that a tag may have a new owner in each iteration of RANDOM$^+$. The last owner of a tag t is its actual owner.

The pseudocodes of RANDOM and RANDOM$^+$ are given in Algorithms 2 and 3. The write(v,t) procedure (line 3) indicates that reader v writes in the memory of a tag t. The *for loop* in lines (2–3) is a write phase of a write/read round. The read(v,t) procedure (line 5) indicates that reader v reads the memory of a tag t. The *for loop* in lines (4–7) represents the read phase of the write/read round. The waiting period spent by readers after a write phase is omitted from the algorithm description to simplify the al-

[6] $m' \geq m$ because some readers in $(v_{(1)}, \ldots, v_{(m')})$ are not necessarily included in $\mathcal{C}$.

Algorithm 3 Algorithm RANDOM$^+$ (ψ) at reader v.

1: **for** i : 1toψ **do**
2: Run algorithm RANDOM with $\mathcal{W}(v) = (\alpha_i(v), \mathbf{id}(v))$
3: If v is redundant, then v terminates the algorithm.
4: Reader v is non-redundant if it is a non-redundant reader in all ψ iterations.



(a) Initial network.

(b) The first execution of RANDOM. Note that $\mathcal{W}(v_1) \succ \mathcal{W}(v_2) \succ \mathcal{W}(v_3) \succ \mathcal{W}(v_4)$.

(c) The second iteration network

(d) The second execution of RANDOM. Note that $\mathcal{W}(v_2) \succ \mathcal{W}(v_3) \succ \mathcal{W}(v_1)$.

Fig. 3. Example of RANDOM$^+$ with $\psi = 2$. The pairs beside the readers are the weights of the readers. Assume that $\mathbf{id}(v_4) \succ \mathbf{id}(v_3) \succ \mathbf{id}(v_2) \succ \mathbf{id}(v_1)$. The non-redundant readers in the first iteration are v_1, v_2 and v_3. These non-redundant readers execute the second iterations with new weights. The readers that are non-redundant in the second iteration are v_2 and v_3. The algorithm terminates with v_2 and v_3 as non-redundant readers.

gorithm description. An illustrative example of RANDOM is given in Fig. 3.

4.3. MAX–MIN: a variation of RANDOM$^+$

A different version of RANDOM$^+$, called MAX–MIN, consists of two iterations only. This algorithm generates fewer non-redundant readers compared to RANDOM$^+$ with two iterations on average. This algorithm requires that the weight of a reader is computed only once. In the first iteration, each tag is owned by the reader v with the *maximum* weight $\mathcal{W}(v) = (\alpha(v), \mathbf{id}(v))$. The second iteration is executed by the non-redundant readers that survived the first iteration. In the second iteration, each tag is owned by the reader with the *minimum* weight $\mathcal{W}(v) = (\alpha(v), \mathbf{id}(v))$. $\alpha(v)$ remains the same in both iterations.

The intuition behind MAX–MIN is the following. We use algorithm SEQ and the reader weights of RANDOM. Let the non-redundant readers selected in sequence by SEQ be $v_{(1)}, v_{(2)}, \ldots, v_{(k)}$. That is, for any $1 \leq j < i \leq k$, $\mathcal{W}(v_{(i)}) \succ \mathcal{W}(v_{(j)})$. Given this sequence, each reader $v_{(j)}$ covers at least one tag that is not covered by any reader $v_{(i)}$, but the opposite is not true. Thus, it is possible that the non-redundant readers $v_{(j)}, \ldots, v_{(k)}$ cover all the tags already covered by $v_{(1)}, \ldots, v_{(j-1)}$. As a result, following the opposite order of readers (i.e., , $v_{(k)}, \ldots, v_{(2)}, v_{(1)}$), which is followed by the second round of MAX–MIN, may generate fewer non-redundant readers than if a random order of readers is followed (i.e., another round of RANDOM as it is case with RANDOM$^+$ with two iterations).

Example:. We give an illustrative example of MAX–MIN being executed over the network given in Fig. 3(a). The output of the first iteration is the same as that in Fig. 3(b). In the second iteration, the readers keep their weights of the first iteration. That is, the weights of the non-redundant readers in the second iteration are $(12, v_1)$, $(10, v_2)$ and $(7, v_3)$. Reader v_3 owns tags t_1 and t_3 since its weight, $(7, v_3)$, is the minimum weight written in these tags. Reader v_2 owns t_2.

Table 2
Terminologies introduced in Section 5.

Term	Definition
$\mathcal{T}^a$	The set of active tags
$\mathcal{T}^a(v)$	The set of active neighbor tags of reader v
$\mathcal{R}^a(v)$	The set of active neighbor readers of reader v

5. GDE: Greedy decentralized elimination

The execution of multiple shuffling procedures by RANDOM$^+$ reduces the number of non-redundant readers generated by RANDOM using randomization. On the other hand, GREEDY always generates non-redundant readers sets of expected size that are less or equal to what RANDOM generates. This is because GREEDY inserts into the cover $\mathcal{C}$ the reader that covers the maximum of not-yet covered tags in every iteration, and thus it outperforms RANDOM on average. Algorithm GREEDY orders the readers according to the number of their *active neighbor tags*, defined as the number of neighbor tags that are not owned yet. This order may change every time a new reader is included into the cover $\mathcal{C}$, or basically in every iteration of GREEDY. The continuous change of the readers order guarantees that the size of the cover set $\mathcal{C}$ generated by GREEDY is less or equal to what RANDOM generates on average (see the conditional probabilities method [19,20] for more details).

The main issue of GREEDY is that it requires a centralized node. Furthermore, the naive decentralized implementation of GREEDY suffers from a high communication cost, since it requires that a maximum reader (i.e., leader election) is repeatedly found until a cover is formed. The objective of algorithm GDE is to overcome this issue by introducing an efficient decentralized implementation of GREEDY.

5.1. Algorithm description

We start introducing the details of GDE by giving the following definitions. The new mathematical terms are summarized in Table 2.

Definition 2. A reader *deactivates* itself if it terminates the execution of the algorithm. Readers that do not deactivate themselves are called *active readers*.

Definition 3. A tag is *deactivated* if it terminates the execution of the algorithm. The set of *active tags* is denoted by $\mathcal{T}^a$. The set of active neighbor tags of a reader v is denoted by $\mathcal{T}^a(v)$.

Definition 4. At an iteration i of GDE, a pair of active readers v and u are called *active neighbor readers* if they share the coverage of at least one active tag t at iteration i. The set of *active neighbor readers* of reader v is denoted by $\mathcal{R}^a(v)$. That is, $\mathcal{R}^a(v) = \{v' | \{\mathcal{T}^a(v) \cap \mathcal{T}^a(v')\} \neq \emptyset\}$.

Algorithm GDE runs in iterations. At least one reader and one tag are deactivated in each iteration. Each iteration is executed by the set of active readers and active tags at that iteration. A tag is deactivated only after being owned by a neighbor reader. The algorithm terminates when all readers and tags are deactivated.

In each iteration, every active reader v writes its weight $\mathcal{W}(v)$ in the memory $\mathcal{M}(t)$ for each active neighbor tag $t \in \mathcal{T}^a(v)$. The weight $\mathcal{W}(v)$ of a reader v is set to $(|\mathcal{T}^a(v)|, \mathbf{id}(v))$ in each iteration v is active in.

Algorithm 4 Algorithm GDE at reader v.

```
1: T^a(v) ← N_T(v);
2: while reader v is active do
3:    W(v) ← (|T^a(v)|, id(v))
4:    for each t ∈ T^a(v) do
5:       write(v,t,W(v))
6:    for each t ∈ T^a(v) do
7:       read(v, t)
8:    if v has the maximum W(v) in each t ∈ T^a(v) (i.e. local maximum) then
9:       v owns and deactivates each t ∈ T^a(v)
10:   update T^a(v) by eliminating all deactivated tags
11:   if T^a(v) = ∅ then
12:      v is deactivated
13: if v owns no tag then
14:   v is redundant and deactivated
```

Definition 5 (Local maximum)**.** A reader v is called a *local maximum* at iteration i if it is the reader with maximum $\mathcal{W}(v)$ among all its active neighbor readers at the same iteration. That is, v is a local maximum if $\mathcal{W}(v) \succ \mathcal{W}(v')$ for each $v' \in \mathcal{R}^a(v)$ in iteration i.

A reader that recognizes that it is a local maximum reader owns all its active neighbor tags $\mathcal{T}^a(v)$. A tag is deactivated by its owner once it is owned. A reader v' that has no active tags deactivates itself. This may occur either because v' owned and deactivated all its active neighbor tags $\mathcal{T}^a(v')$, or because all the neighbor tags $N_T(v')$ of v' are owned by other readers. As a result, a tag is owned by exactly one reader. This is because the tags are owned by local maximum readers and are deactivated as soon as they are owned. Note that according to the definition of local maximum readers, it is not possible for a pair of local maximum readers to own the same tag during the execution of the algorithm.

The implementation of the rules given above requires two write/read rounds per iteration. Each active reader v writes its weight $\mathcal{W}(v)$ in $\mathcal{M}(t)$ for every active neighbor tag t in $\mathcal{T}^a(v)$. Then, an active reader v checks if it has the maximum weight $\mathcal{W}(v)$ at each tag $t \in \mathcal{T}^a(v)$. If this is the case, then v is a local maximum reader. A local maximum reader owns and deactivates its active neighbor tags by writing a special flag in their memories. The pseudocode of algorithm GDE, executed at reader v, is given in Algorithm 4. An example illustrating how GDE works is given in Fig. 4.

5.2. Theoretical analysis

Theorems 5.2 and 5.3, given below, prove the correctness of GDE. Theorem 5.2 proves that the non-redundant readers generated by GDE are exactly the same as those generated by GREEDY. Theorem 5.3 proves that GDE terminates in at most $|\mathcal{R}|$ iterations, each of which consists of two write/read rounds. Before proving Theorem 5.2, we prove the following Lemma.

Lemma 5.1. *The local maximum readers in any reader network do not share any neighboring tags. That is, if $\mathcal{R}_{lmax}$ is the set of local maximum readers in a given network, then $N_T(v) \cap N_T(u) = \emptyset$ for every pair v and u in $\mathcal{R}_{lmax}$. This means that a pair of local maximum readers cannot also be* coverage neighbor *readers.*

Proof. By contradiction, if there is a tag t shared between two local maximum reader v and u, then by the uniqueness of the reader weights only one of v and u can be the maximum reader for t (let it be v). As a result, the other reader u is not a maximum of t and hence is not a local maximum reader. □

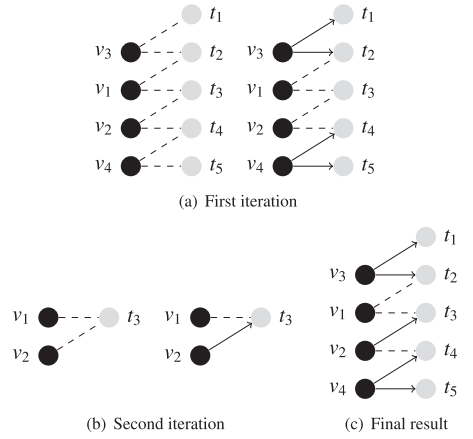


Fig. 4. Example of GDE. Initially the network consists of four readers and four tags. The set $\mathcal{R} = \{v_4, v_3, v_2, v_1\}$ where $\mathbf{id}(v_4) \succ \mathbf{id}(v_3) \succ \mathbf{id}(v_2) \succ \mathbf{id}(v_1)$. In the first iteration, the set of local maximum readers is $\{v_4, v_3\}$. Reader v_4 deactivates all its neighbor tags, which are t_4 and t_5. Reader v_3 also deactivates all its neighbor tags, which are t_1 and t_2. At this point, the readers v_4 and v_3 find that all their neighbor tags are deactivated, and hence terminate the execution of the algorithm. Reader v_1 and v_2 find that one of their neighbor tags, namely t_3, is not yet deactivated. Therefore, both readers move to the second iteration of GDE. In the second iteration, the network consists of v_1 and v_2 and the only active tag is t_3 as shown in 4(b). Reader v_2 is the only local maximum reader since $\mathbf{id}(v_2) \succ \mathbf{id}(v_1)$ and both have only one active neighbor tag (that is, $\mathcal{T}^a(v_2) = \mathcal{T}^a(v_1) = \{t_3\}$). Therefore, v_2 owns and deactivates t_3. The algorithm terminates since all readers are deactivated. The set of non-redundant readers is $\{v_4, v_3, v_2\}$, as every one of these readers owned at least one neighbor tag.

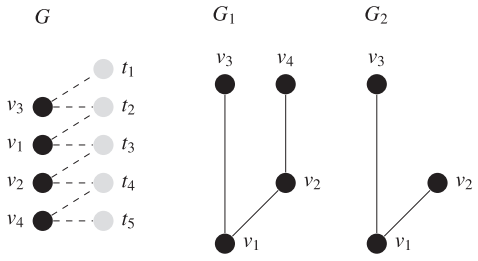


Fig. 5. Example for the directed graph used in the proof of Theorem 5.2. G_1 is formed from the readers $\{v_4, v_3, v_2, v_1\}$ and the tags $\{t_5, t_4, t_3, t_2, t_1\}$ in G. Every coverage neighbor reader in G shares an edge in G_1. G_2 is formed from the removal of v_4, since it is the first to be picked by GREEDY, and its neighbor tags t_5 and t_4.

Theorem 5.2. *Let $\mathcal{C}_s$, $\mathcal{C}_d$ be the set of non-redundant readers found by GREEDY and GDE respectively, then $\mathcal{C}_s = \mathcal{C}_d$.*

Proof. Let us create a graph $G_1 = (\mathcal{R}, \mathcal{E}_{cov}[\mathcal{R}, \mathcal{T}])$ where $\mathcal{R}$ is the set of readers and $\mathcal{E}_{cov}[\mathcal{R}, \mathcal{T}]$ are the coverage relationships between the readers such that $(u, v) \in \mathcal{E}_{cov}[\mathcal{R}, \mathcal{T}]$ if both readers share the coverage of at least one tag in $\mathcal{T}$ (i.e., coverage neighbor readers). We orient the edges of G_1 such that an edge (u, v) indicates that $\mathcal{W}(u) \succ \mathcal{W}(v)$ (see Fig. 5 for an example). In the oriented graph G_1, the local maximum readers are the *sources* and are denoted by $\mathcal{R}_{lmax}(G_1)$. The first reader to be selected by GREEDY, denoted $v_{(1)}$, is necessarily in $\mathcal{R}_{lmax}(G_1)$. We create then a new oriented graph G_2 induced from the set of readers $\{\mathcal{R}/\{v_{(1)}\}\}$ and the set of tags $\{\mathcal{T}/\{N_T(v_{(1)})\}\}$. According to Lemma 5.1, the local maximum readers in G_1 except $v_{(1)}$ have to be local maximum readers in G_2. As a result, the second reader to be picked by GREEDY, denoted $v_{(2)}$, is 1) in $\{\mathcal{R}_{lmax}(G_1)/\{v_{(1)}\}\}$ (since their weights will not change according to Lemma 5.1), or 2) in a directed path from $v_{(1)}$ to a sink in the graph G_1. For the second case, this happens because if $v_{(2)}$ was in a directed path starting from reader $w \neq v_{(2)}$ to a sink node, then w is necessarily a local maximum reader (i.e., $\mathcal{W}(w) \succ \mathcal{W}(v_{(2)})$) and hence w should be picked first by GREEDY instead of $v_{(2)}$. This shows that there is an independence between

the maximum readers $\mathcal{R}_{lmax}(G_1)$. Thus, all of them can be selected by GREEDY in the same iteration and this is what GDE does. □

Theorem 5.3. *Algorithm GDE terminates correctly in at most* $2|\mathcal{R}|$ *write/read rounds.*

Proof. There must be at least one local maximum reader in each iteration of GDE, which is essentially the maximum reader among all active readers. This is guaranteed by the uniqueness of the weights. Therefore, GDE requires at most $|\mathcal{R}|$ iterations. Each deactivated reader v deactivates all its active neighbors tags. Each iteration consists of two write/read rounds as defined in the algorithm description.

The upper bound $|\mathcal{R}|$ is shown to be tight by building the following scenario. For simplicity, let $\mathcal{R} = \{v_1, \ldots, v_m\}$ and let $\mathbf{id}(v_i) \succ \mathbf{id}(v_{(i-1)})$ for $1 < i \leq m$. We define the following line graph $\mathcal{L}(m)$. Each reader v_i covers a set of tags T_i that are not covered by any other reader, and $|T_i| = i$. The readers are placed on a line, such that each pair of readers v_i and v_{i+1} share the coverage of a tag $t_{(i,i+1)}$ for $1 < i < m$. This is depicted in Fig. 6. Note that v_m and v_{m-1} cover the maximum number of tags, which is equal to m. Since $\mathbf{id}(v_m) \succ \mathbf{id}(v_{m-1})$, then v_m is the only local maximum at this configuration. The elimination of v_m generates the graph $\mathcal{L}(m-1)$. The same procedure is repeated $m = |\mathcal{R}|$ times. Every reader v_i is a local maximum in iteration i. □

6. LIMITED-GDE: Limiting the number of iterations of GDE

The main issue in GDE is that it may require a large number of write/read rounds. Algorithm LIMITED-GDE reduces the number of iterations of GDE by limiting the number of its iterations to a constant integer $\psi \geq 1$. The constant ψ is a design parameter used to balance the number of iterations and the number of non-redundant readers.

The algorithm exploits similarities between GDE and RRE. Algorithm LIMITED-GDE executes GDE for $\psi - 1$ iterations at most. If GDE is found to terminate in one of the $\psi - 1$ iterations, then LIMITED-GDE terminates. Otherwise, every active tag t at iteration ψ is owned by its active neighbor reader v with the maximum weight $\mathcal{W}(v)$ among all active neighbors readers of t, where $\mathcal{W}(v) = (|\mathcal{T}^a(v)|, \mathbf{id}(v))$. Note that this is similar to algorithm RRE, since every tag is owned by the reader v that wrote in $\mathcal{M}(t)$ and has the maximum weight $\mathcal{W}(v)$.

Example:. Consider the reader network given in Fig. 4. Let ψ be set to 2. GDE does not terminate in $\psi - 1$ iterations. At the ψ th iteration, the set of active readers are $\{v_1, v_2\}$, while the only active tag is t_3. Readers v_1 and v_2 write their weights in t_3. Reader v_2 owns tag t_3 since its weight is the largest according to tag t_3. The non-redundant readers are $\{v_4, v_3, v_2\}$, which is the same result found by executing GDE. However, only $2 \times (\psi - 1) + 1 = 3$ write/read rounds were executed, whereas GDE executed 4 write/read rounds. The difference between the two algorithms is clearer in larger reader networks.

Theorem 6.1. *Algorithm LIMITED-GDE terminates correctly in at most* $2(\psi - 1) + 1$ *write/read rounds.*

We show in the following that RRE, LIMITED-GDE, and GDE all belong to the same family of algorithms. LIMITED-GDE is the generalization of these algorithms. The weight of a reader is decided by the number of the tags it covers. Knowing the relationship between these algorithms, we can design other similar algorithms but with different definitions of readers weights. Theorem 6.2 below shows the relationship between:

1. The set of non-redundant readers generated by GDE (denoted by $\mathcal{C}_g$),
2. The set of non-redundant readers generated by LIMITED-GDE after i iterations (denoted by $\mathcal{C}_l(i)$ for $1 \leq i \leq \psi$), and
3. The set of non-redundant readers generated by RRE (denoted by $\mathcal{C}_r$).

Theorem 6.2. *For an integer* $\psi > 1$, $|\mathcal{C}_g| \leq |\mathcal{C}_l(\psi)| \leq |\mathcal{C}_l(\psi - 1)| \leq \cdots \leq |\mathcal{C}_l(1)| = |\mathcal{C}_r|$.

Proof. Let the set of active tags at iteration i be $\mathcal{T}_i^a$. Let $\mathcal{R}_{max}(i)$ be the set of readers that are maximum in at least one of their neighbor tags at iteration i. Let $\mathcal{R}_{lmax}(i)$ be the set of local maximum readers at iteration i. Then:

$$\mathcal{C}_l(1) = \mathcal{R}_{max}(1) = \mathcal{C}_r$$
$$\mathcal{C}_l(i)_{i>1} = \mathcal{R}_{max}(i) \cup \{\mathcal{R}_{lmax}(i-1) \cup \cdots \cup \mathcal{R}_{lmax}(0)\}$$

Note that $\mathcal{C}_g = \bigcup_{1 \leq i \leq z} \mathcal{R}_{lmax}(i)$, where z is the number of iterations required by GDE to terminate. Also note that $\mathcal{R}_{lmax}(i) = \mathcal{R}_{max}(i) = \emptyset$ for all $i > z$ since all tags and readers are deactivated after z iterations. From the definition of LIMITED-GDE, $\mathcal{C}_l(z) = \mathcal{C}_g$, and the same applies for all $\mathcal{C}_l(i)$ such that $i \geq z$.

Lastly, note that $\mathcal{R}_{lmax}(i) \subseteq \mathcal{R}_{max}(i)$. This leads to $\mathcal{R}_{max}(i) \subseteq \mathcal{R}_{max}(i-1)$ since, in each iteration i, the readers of $\mathcal{R}_{lmax}(i-1)$ are already in $\mathcal{C}_l(i)$ and $\mathcal{R}_{lmax}(i-1) \subseteq \mathcal{R}_{max}(i-1)$. Therefore:

$$\mathcal{C}_l(z+1) \subseteq \mathcal{C}_l(z) \subseteq \mathcal{C}_l(z-1) \cdots \subseteq \mathcal{C}_l(2) \subseteq \mathcal{C}_l(1)$$

which completes the proof. □

7. Simulation results

This section studies the performance of the reader-tag RFC algorithms introduced in this paper using simulation experiments. We compare our algorithms, namely RANDOM, RANDOM$^+$, MAX-MIN, GDE and LIMITED-GDE, against algorithms RRE [6], DRRE [7], and NTE [16]. LEO [8] is excluded from these comparisons since its description in [8] lacks some important implementation details (e.g., the readers write/read scheduling algorithm). GREEDY [6] is also excluded since it forms the exact same set of non-redundant readers that is formed by GDE (see Theorem 5.2).

The performance of these algorithms is studied using the following performance metrics:

1. The number of non-redundant readers,
2. The number of write operations.

The selection of the best combinations of the number of readers and tags is a difficult task since applications of RFC algorithms can fit in small-scale indoor or large-scale outdoor environments. The positioning of the readers and tags forms another challenging task for the same reason. Thus, RFC algorithms are studied under different network topologies; each of which simulates certain scenarios where reader networks may be used. This would enrich the understanding of the studied algorithms behavior. These topologies are called:

1. *Uniform geometric topologies*; defined in Section 7.1.1,
2. *Arbitrary topologies* (or, non-geometric topologies); defined in Section 7.1.2.

The superiority of GDE is clear in all the experiments of this chapter. RANDOM outperforms RRE and DRRE in most experiments, whereas MAX-MIN has an even better performance as it outperforms RANDOM (2) and RANDOM (3) (i.e., RANDOM$^+$ with 2 and 3 iterations respectively) in most experiments. These results, however, occur in geometric topologies. The experiments of this chapter show that RANDOM, MAX-MIN, and LIMITED-GDE (1) (that is, LIMITED-GDE with $\psi - 1 = 1$) are shown to have a balance between the number of non-redundant readers, number of write/read rounds and the number of read and write operations.

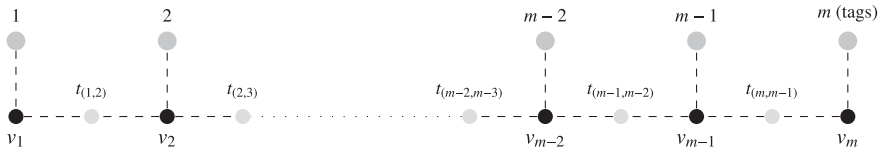


Fig. 6. The line graph $\mathcal{L}(m)$. The readers are represented as black circles, the tags are represented as gray circles. The larger and darker gray circles are combinations of tags located very closely to each other.

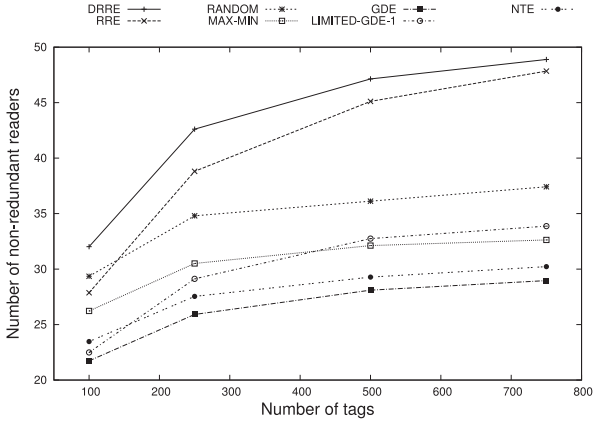


Fig. 7. Impact of the number of tags on the number of non-redundant readers in uniform geometric topologies with 100 readers.

In the following, each experiment consists of 50 trials. The figures in this section show the averages of these trials. More detailed simulation results can be found in [21]. This includes the numerical results tables, the impact of different types of topologies, and the impact of the number of write/read rounds on LIMITED-GDE and RANDOM$^+$.

7.1. Number of non-redundant readers

7.1.1. Uniform geometric topologies

This section studies the performance of the reader-tag RFC algorithms under *uniform geometric topologies*. These topologies are constructed as follows; a set of readers and tags are spread uniformly randomly in a rectangular plane with an area of 200 m × 200 m. A reader covers a tag if the Euclidean distance between them is at most 5 m, which is the *interrogation range* of readers (i.e., r_i). The experiments consider only the networks that guarantee that every tag is covered by at least one reader.

The number of tags impact:. At first, the impact of the number of tags on the number of non-redundant readers is studied. The following experiments fix the number of readers to 100 readers, while the number of tags is in {100, 250, 500, 750}. The results of these experiments are shown in Fig. 7. GDE outperforms all the other algorithms, even the centralized algorithm NTE which uses more complex rules. Nevertheless, NTE outperforms the algorithms except GDE. This is an expected superiority since all the other algorithms are decentralized, and hence, use less resources compared to centralized algorithms.

Fig. 7 shows that RRE outperforms DRRE despite the fact that DRRE executes two write/read rounds whereas RRE executes only one write/read round. To elaborate more on this point, note that the weight $\mathcal{W}(v)$ of a reader v in RRE is $(|N_T(v)|, \mathbf{id}(v))$, where $N_T(v)$ is the number of neighbor tags of reader v, whereas the weight $\mathcal{W}(v)$ of a reader v in DRRE is $(|N_{R_s}(v)|, \mathbf{id}(v))$, where $N_{R_s}(v)$ is the number of coverage neighbor readers of v. To know their coverage neighbor readers, the readers must execute a write/read round in which each reader appends its identifier to all its neighbor tags memory contents. After that, each reader v reads the memory of all its neighbor tags and computes $N_{R_s}(v)$. As a result, a DRRE reader makes a decision of whether it is redundant or not at the end of the second write/read round. This issue, however, is not found in RRE.

Using SEQ to understand reader-tag RFC algorithms:. Fig. 7 shows that RANDOM forms fewer non-redundant readers compared to RRE in the uniform geometric topologies. The outperformance is more significant as the number of tags increases. For instance, RRE generates 25% more redundant readers than RANDOM in networks with 750 tags.[7]. This is an improvement of about 10 readers on average. This result is related to the increase in the average tags counts as the number of tags increases (that is, as $|\mathcal{T}|$ increases, the average size of $N_T(v)$ for each $v \in \mathcal{R}$ increases as well). This result is explained informally in the following using the generalized sequential algorithm SEQ of Section 4.1.

Recall first that SEQ is a generalization of DRRE, RRE, and RANDOM. That it, the set of non-redundant readers $\mathcal{C}$ formed by the sequential versions of DRRE, RRE, and RANDOM are exactly the same as that of the decentralized versions. According to its description, SEQ passes the readers in a descending order according to their weights, where a weight of a reader v is $\mathcal{W}(v)$.[8] The sorted list of readers is denoted by $\{v_{(1)}, \ldots, v_{(m)}\}$, where $v_{(i)}$ indicates that $v_{(i)}$ is the i^{th} ranked reader by the algorithm (that is, $\mathcal{W}(v_{(i)}) \succ \mathcal{W}(v_{(j)})$ if $i < j$). A reader $v_{(i)}$ is added to the solution $\mathcal{C}$ only if it covers at least one not-yet owned tag. The sorted set of readers that are included in the solution $\mathcal{C}$ is denoted by $\{v_{c(1)}, \ldots, v_{c(k)}\}$, where k is size of $\mathcal{C}$ (that is, $v_{c(i)}$ is the i^{th} reader to be included in $\mathcal{C}$ by SEQ).

According to the definitions given above, it is necessary that $v_{(1)} = v_{c(1)}$. Thus, RRE must include $v_{(1)}$ in its solution $\mathcal{C}$, where reader $v_{(1)}$ is the reader that covers the maximum number of tags (since $|N_T(v_{(1)})|$ is the maximum among all other readers in $\mathcal{R}$). However, the main issue of RRE is caused by the fact that its sequential version sorts the readers only once. Thus, a reader $v_{c(i)}$, for $i > 1$, is not necessarily the reader that covers the maximum number of tags in iteration i. In fact, the inclusion of $v_{c(i)}$ in $\mathcal{C}$ at iteration i may lead to unnecessary increase in the size of $\mathcal{C}$. As an example, a possible scenario is when $v_{(2)}$ owns a small number of tags although $|N_T(v_{(2)})|$ is large. This happens if $|\{N_T(v_{(1)}) \cap N_T(v_{(2)})\}| = \beta$ where $0 < \beta \ll |N_T(v_{(2)})|$. Given that $\beta > 0$, $v_{(2)}$ must be included in $\mathcal{C}$ according to SEQ (that is, $v_{c(2)} = v_{(2)}$) although $v_{(2)}$ would own only a small subset of its neighbor tags. The tags that would be owned by $v_{(2)}$ can be owned by other readers given that they are few in quantity. If there are many readers having the same characteristic of $v_{(2)}$ (i.e., own a small set of tags that could have been owned by other readers),

[7] This is calculated as $1 - |\mathcal{C}_{\text{RRE}}| \setminus |\mathcal{C}_{\text{RANDOM}}|$, where $|\mathcal{C}_{\text{RRE}}|$ is the number of non-redundant readers generated by RRE in this set of experiments. $|\mathcal{C}_{\text{RANDOM}}|$ is defined similarly

[8] Note that the definition of the reader weights is different in each algorithm. For instance, in RRE the weight $\mathcal{W}(v)$ is set to $(|N_T(v)|, \mathbf{id}(v))$, whereas it is set to $(\alpha(v), \mathbf{id}(v))$ in RANDOM.

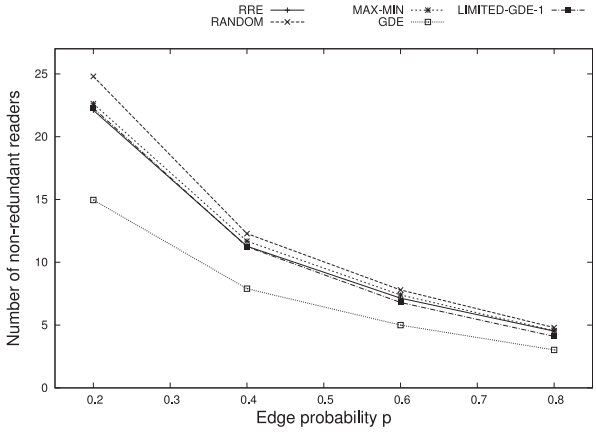


Fig. 8. Impact of the edge probability on the number of non-redundant readers in arbitrary topologies.

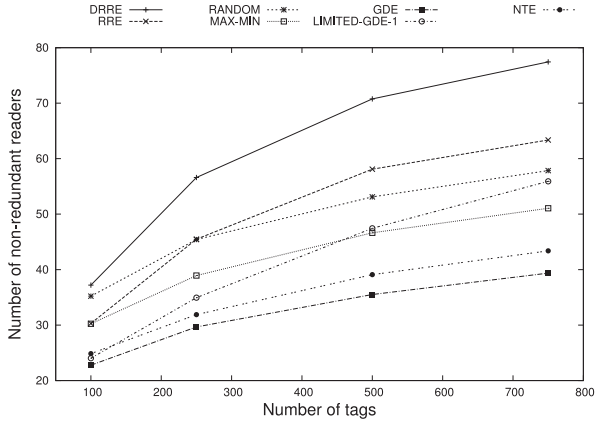


Fig. 9. Impact of the number of tags on the number of non-redundant readers in uniform geometric topologies with probabilistic edges ($p = 0.9$, 150 readers).

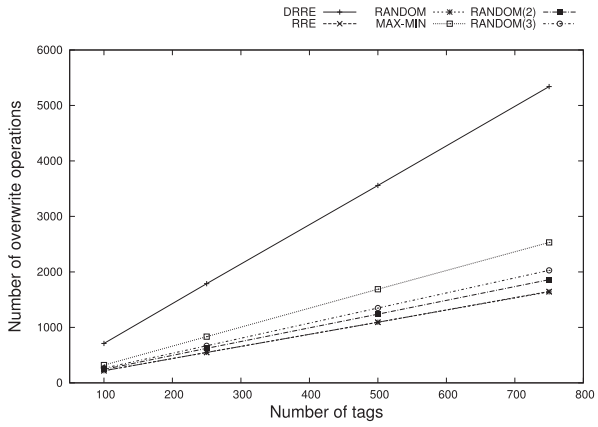


Fig. 10. Impact of the number of tags on the total number of overwrite operations in uniform geometric topologies.

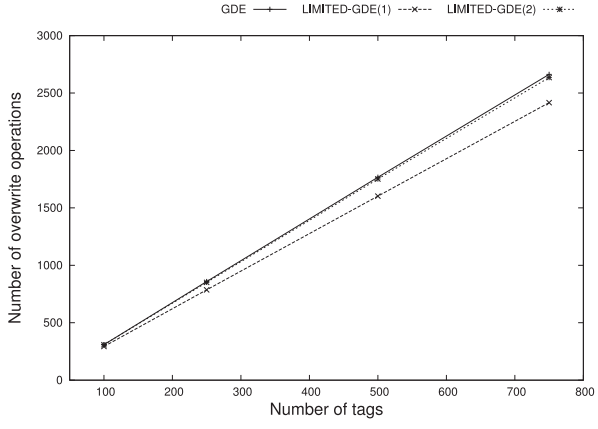


Fig. 11. Impact of the number of tags on the total number of overwrite operations in uniform geometric topologies.

then the number of non-redundant readers increases unnecessarily. In the studied geometric topologies, the readers are located in proximity of each other. Thus, the number of tags shared between the readers increases. This increases the number of readers which have the same characteristic of $v_{(2)}$ in the example above. This high level of intersection between readers becomes clearer as the average tag count increases in geometric topologies. This explains the weaknesses of RRE.

GREEDY, and hence GDE, solves the issue of RRE (described above) by resorting the set of readers every time a new reader is included in $\mathcal{C}$. That is, the reader that covers the maximum number of not-yet owned tags is included in $\mathcal{C}$ in each iteration. RANDOM, on the other hand, solves this issue by random selection of readers. This random strategy explains the superiority of RANDOM over RRE in this type of geometric topologies.

Fig. 7 shows that algorithm MAX-MIN outperforms LIMITED-GDE (1). The out-performance can be explained using the same argument used to explain the superiority of RANDOM over RRE. This is because MAX-MIN can be considered as a two-iteration version of RANDOM (see Section 4.3). Similarly, LIMITED-GDE (1) can be considered as a two-iteration version of RRE (see Theorem 6.2).

7.1.2. Arbitrary topologies

The results observed above are related to the geometric nature of the studied topologies. To enforce the previous arguments, the following experiments study the performance of the algorithms in non-geometric topologies. These topologies are called *arbitrary topologies*. An arbitrary topology is a random bipartite graph $G = (\mathcal{R}, \mathcal{T}, E_s)$, where $\mathcal{R}$ is the set of readers and $\mathcal{T}$ is the set of tags. An edge $(v, t) \in E_s$ represents a coverage relationship between a reader and a tag. The probability that an edge exists in E_s is fixed to a constant p, where $0 \leq p \leq 1$. Each of the topologies studied in the following experiments consists of 100 readers and 750 tags. The edge probability is in {0.2, 0.4, 0.6, 0.8}.

The impact of the edge probability on the number of non-redundant readers is shown in Fig. 8. Most of the results in these experiments are repetitions of previous results. However, note that RRE outperforms RANDOM in these experiments. This is contrary to previous experiments. This confirms further the argument used previously to explain the superiority of RANDOM over RRE in geometric topologies.

A mixture of arbitrary and uniform geometric topologies:. Previous experiments show that the behavior of RRE and RANDOM depends on whether the network is geometric or arbitrary (i.e., non-geometric). Thus, it shall be natural to study the behavior of these algorithms in topologies that have characteristics of geometric and arbitrary topologies. These topologies are uniform geometric topologies in which an edge between a reader v and a tag t can be removed with a probability $1 - p$ for $0 \leq p \leq 1$. That is, each reader connects to a tag within its interrogation range with probability p. This construction method can be used to emulate the inaccuracy of the tag discovery procedure, and hence it adds more practicality to uniform geometric topologies and arbitrary topologies. The performance of RFC algorithms is studied under this type of topologies. The number of readers is set to 150 readers, while p is set to 0.9.

Table 3
Summary of the advantages and disadvantages of the algorithms presented in this paper.

Algorithm	Advantages	Disadvantages
RANDOM	Serves as a benchmark to RFID coverage algorithms that use the reader-tag communication model. It also outperforms major algorithms in practical scenarios even if it should be considered as a benchmark. Requires only write/read round. Reader weight $\mathcal{W}(v)$ is easy to compute.	Can generate a large number of non-redundant readers.
RANDOM$^+$	Improves the performance of RANDOM by simply running multiple RANDOM write/read rounds.	Can generate a large number of non-redundant readers.
MAX-MIN	Improves the performance of RANDOM by simply running two write/read rounds. Does not require to compute the reader weight $\mathcal{W}(v)$ more than once.	Can generate a large number of non-redundant readers.
GDE	Generates the exact set of non-redundant readers generated by GREEDY but in a decentralized fashion. Generates a very low number of non-redundant readers; the best in the literature so far.	May require a large number of write/read rounds to terminate
LIMITED-GDE	Reduces the maximum number of write/read rounds to a constant. Generates a low number of non-redundant readers even with a small ψ	Every iteration requires two write/read rounds except the last. This may be expensive in some scenarios. RANDOM$^+$ and MAX-MIN are more suitable in such cases.

The impact of the number of tags on the number of non-redundant readers is shown in Fig. 9. All algorithms are observed to generates a larger number of non-redundant readers if compared against the case where uniform geometric topologies are used (see Fig. 7). This is because the number of non-redundant readers in the networks of Fig. 9 is smaller compared to the networks of Fig. 7. The results show that the out-performance of RANDOM over RRE becomes clearer as the number of tags increases; whereas the performance of both algorithms is approximately the same in networks with smaller number of tags. This result suggests, as the density of the network increases, the effect of the uniform removing of edges becomes neglected.

7.2. Number of write operations

This section compares the RFC algorithms with respect to the number of write operations they execute. Two important issues regarding the write operations must be elaborated before analyzing the results of this section:

1. For a reader v to write a sequence of bits B in the memory of a neighbor tag t, the reader 1) reads the memory content $\mathcal{M}(t)$ of t and locally stores it in M_b, where M_b is in the memory of v, 2) appends to B to M_b (that is, create M_a where $M_a = \{M_b \cup B\}$), and 3) *overwrites* M_a in the memory of tag t. Therefore, a write operation is also called an *overwrite* operation in this section.
2. Most of the studied algorithms do not necessitate that a reader writes a value in each write/read round it is active in. For instance, the readers in RRE and RANDOM are interested only in the maximum weight already written in the memories of their neighbor tags. Therefore, a reader v overwrites its weight $\mathcal{W}(v)$ in the memory $\mathcal{M}(t)$ of a neighbor tag t only if it finds that $\mathcal{W}(v)$ is larger than what is already written in t. The same concept applies to GDE and LIMITED-GDE. Write operations are slightly different in DRRE. The first iteration of DRRE requires that a reader overwrites all its active neighbor tags in each write round. This is because each reader must write its identifier in the memory of all its neighbor tags. These written identifiers are then used by each reader in order to find its neighbor readers. This causes an increase in the number of write operations in DRRE.

The total number of overwrite operations executed by the studied RFC algorithms are shown in Figs. 10 and 11 (separated into two figures for clarity). The total number of overwrite operations is computed similarly. Note that the number of read and overwrite operations depends mainly on the number of the reader-tag cover relationships in the studied networks, and the number of write/read rounds executed by the algorithm. Note that MAX-MIN executes significantly more overwrites compared to RANDOM (2). This is despite that MAX-MIN and RANDOM (2) execute the same number of read operations. This is a direct result of using the inverse order of readers when executing the second iteration of MAX-MIN. This also explains the smaller number of non-redundant readers generated by MAX-MIN compared to RANDOM (2). Note as well that the rate of decrease in the number of overwrites in LIMITED-GDE decreases as we increase the number of iterations. This is because there are fewer non-redundant readers in each new iteration.

8. Summary and Conclusions

A summary of the algorithms introduced in this paper is given in Table 3. We introduced in this paper two sets of algorithms that solve the RFID coverage problem using writeable tags. The first set consists of randomized algorithms, called RANDOM and RANDOM$^+$. It also consists of a variant of these algorithms, called MAX-MIN. The second set of algorithms consists of deterministic algorithms, called GDE and LIMITED-GDE. Our algorithms were shown to outperform major existing algorithms in different scenarios. There are still more variations of this problem (e.g., adding load balancing constraints, or others). A promising direction is fault tolerance. That is, how to solve the RFID coverage problem if it is assumed that the links between readers and tags may fail (e.g., a reader is

not able to write in a tag in its proximity). Another direction is the problem of optimizing write/read rounds. Lastly, we see that the reader-tag RFID coverage problem still requires future research for two main reasons; the novelty of the reader-tag communication model, and the various and promising applications of the problem.

References

[1] C. Swedberg, Chip-size passive RFID tag promise long range, RFID J (2009). http://www.rfidjournal.com/article/view/4585
[2] EPCglobal, Tag Data Standard, version 1.6, Tech. Rep., EPCglobal, 2011.
[3] C. Aggarwal, J. Han, Springer, 2013.Ch.11
[4] B. Wang, Coverage problems in sensor networks: A survey, ACM Comput Surv 43 (4) (2011) 1–53.
[5] B. Cărbunar, A. Grama, J. Vitek, O. Cărbunar, Redundancy and coverage detection in sensor networks, ACM Trans Sens Netw. 2 (1) (2006) 94–128.
[6] B. Carbunar, M.K. Ramanathan, M. Koyutürk, S. Jagannathan, A. Grama, Efficient tag detection in RFID systems, J. Parallel Distrib Comput. 69 (2) (2009) 180–196.
[7] K.-M. Yu, C.W. Yu, Z.-Y. Lin, A density-based algorithm for redundant reader elimination in a RFID network, in: The second international conference on future generation communication and networking (FGCN'08), 1, IEEE, 2008, pp. 89–92.
[8] C.-H. Hsu, Y. Chen, C.-T. Yang, A layered optimization approach for redundant reader elimination in wireless RFID networks, in: The second IEEE Asia-Pacific service computing conference, IEEE, 2007, pp. 138–145.
[9] N. Irfan, M. Yagoub, Efficient algorithm for redundant reader elimination in wireless RFID networks, Int j comput sci issues 3 (11) (2010) 1–8.
[10] Q. Dong, A. Shukla, V. Shrivastava, D. Agrawal, S. Banerjee, K. Kar, Load balancing in large-scale RFID systems, Comput Netw 52 (2008) 1782–1796.
[11] V.G. Dhas, R. Muthukaruppan, K. Balakrishnan, R. Ganesan, Optimal solution for RFID load balancing, in: Recent trends in networks and communications, Springer, 2010, pp. 41–49.
[12] A. Jedda, H.T. Mouftah, Decentralized RFID coverage algorithms with applications for the reader collision avoidance problem, IEEE Trans Emerg Top Comput (99) (2015).
[13] S. Lu, S. Yu, A fuzzy k-coverage approach for RFID network planning using plant growth simulation algorithm, J Netw Comput Appl 39 (2014) 280–291.
[14] C.E. Leiserson, R.L. Rivest, C. Stein, T.H. Cormen, Introduction to algorithms, The MIT press, 2001.
[15] H. Yihua, L. Shilei, A middleware-based approach for redundant reader elimination, in: The fifth international conference on new trends in information science and service Science (NISS), 1, IEEE, 2011, pp. 209–214.
[16] K. Ali, W. Alsalih, H. Hassanein, Using neighbor and tag estimations for redundant reader eliminations in RFID networks, in: Wireless communications and networking conference (WCNC), IEEE, 2011, pp. 832–837.
[17] W.A.A. Alsalihy, N.A. Askar, Enhancement of redundant reader elimination by using hybrid algorithm in RFID systems, Wireless Pers Commun (2013) 1–18.
[18] , Tag Data Standard, version 1.6, Tech. Rep., EPCglobal, 2011.
[19] R. Motwani, P. Raghavan, Randomized algorithms, Cambridge university press, 1995.
[20] N. Young, Randomized rounding without solving the linear program, in: The sixth annual ACM-SIAM symposium on Discrete algorithms, Society for industrial and applied mathematics, 1995, pp. 170–178.
[21] A. Jedda, Distributed algorithms for networks formation in a scalable internet of things, University of Ottawa, 2014 Ph.D. thesis.

Ahmed Jedda received the M.Sc. and Ph.D. degrees in computer science from the university of Ottawa, Canada. He worked in the WiSense Laboratory during his Ph.D. studies. His research interests include distributed network formation algorithms, the Internet of Things, Bluetooth networks, RFID reader networks, structured Peer-To-Peer (P2P) networks, and data streams processing.

Mazen G. Khair received his B.Sc. degree from the Department of Electrical and Computer Engineering of Al-Mustansiriya University, Iraq, in 1997 and M.Sc. from the School of Information Technology and Engineering of the University of Ottawa, Canada, in 2004. Between March 2002 and March 2004, he was an intern at the Communication Research Center (CRC) of Canada. He enrolled in the Ph.D. program at the University of Ottawa in 2004, and was a research and teaching assistant there between 2002 and 2008. He was a member of the Center of Research in Photonics at the University of Ottawa between April 2004 and February 2006. During his Ph.D. studies, he was a software engineer at Nortel networks between March 2006 and May 2008. Currently, he is a PostDoc at the University of Ottawa, Canada.

Hussein T. Mouftah received his B.Sc. and M.Sc. from Alexandria University, Egypt, in 1969 and 1972 respectively, and his Ph.D. from Laval University, Quebec, Canada in 1975. He joined the School of Information Technology and Engineering (now School of Electrical Engineering and Computer Science) of the University of Ottawa in 2002 as a Tier 1 Canada Research Chair Professor, where he became a Distinguished University Professor in 2006. He has been with the ECE Dept. at Queen's University (1979–2002), where he was prior to his departure a Full Professor and the Department Associate Head. He has six years of industrial experience mainly at Bell Northern Research of Ottawa (then known as Nortel Networks). He served as Editor-in-Chief of the IEEE Communications Magazine (1995–97) and IEEE ComSoc Director of Magazines (1998–99), Chair of the Awards Committee (2002–03), Director of Education (2006–07), and Member of the Board of Governors (1997–99 and 2006–07). He has been a Distinguished Speaker of the IEEE Communications Society (2000–2007). He is the author or coauthor of 8 books, 60 book chapters and more than 1200 technical papers, 12 patents and 140 industrial reports. He is the joint holder of 14 Best Paper and/or Outstanding Paper Awards. He has received numerous prestigious awards, such as the 2007 Royal Society of Canada Thomas W. Eadie Medal, the 2007–2008 University of Ottawa Award for Excellence in Research, the 2008 ORION Leadership Award of Merit, the 2006 IEEE Canada McNaughton Gold Medal, the 2006 EIC Julian Smith Medal, the 2004 IEEE ComSoc Edwin Howard Armstrong Achievement Award, the 2004 George S. Glinski Award for Excellence in Research of the U of O Faculty of Engineering, the 1989 Engineering Medal for Research and Development of the Association of Professional Engineers of Ontario (PEO), and the Ontario Distinguished Researcher Award of the Ontario Innovation Trust. Dr. Mouftah is a Fellow of the IEEE (1990), the Canadian Academy of Engineering (2003), the Engineering Institute of Canada (2005) and the Royal Society of Canada RSC Academy of Science (2008).