Multi-Seed Group Labeling in RFID Systems

Jihong Yu, Jiangchuan Liu, Rongrong Zhang, Lin Chen, Wei Gong, Shurong Zhang

***Abstract*—Ever-increasing research efforts have been dedicated to radio frequency identification (RFID) systems, such as finding top-k, elephant groups, and missing-tag detection. While group labeling, which is how to tell tags their associated group data, is the common prerequisite in many RFID applications, its efficiency is not well optimized due to the transmission of useless data with only one seed used. In this paper, we introduce a unified protocol called GLMS which employs multiple seeds to construct a composite indicator vector (CIV), reducing the useless transmission. Technically, to address *Seed Assignment Problem* (SAP) arising during building CIV, we develop an approximation algorithm (AA) with a competitive ratio 0.632 by globally searching for the seed contributing to the most useful slot. We then further design two simplified algorithms through local searching, namely c-search-I and its enhanced version c-search-II, reducing the complexity by one order of magnitude while achieving comparable performance. We conduct extensive simulations to demonstrate the superiority of our approaches.**

***Index Terms*—RFID, group labeling, seed assignment.**

I. Introduction

Recent years have witnessed an unprecedented development of the radio frequency identification (RFID) technology in various applications ranging from inventory control [2] and supply chain management [28] [15], to object tracking [18] and localization [5]. An RFID system typically consists of RFID readers and tags wherein a reader can query tags and collect information from tags, and a tag works in mode of backscatter communications [23], which captures energy in RF signal from its nearby reader and use the energy to send message to this reader. With the development of RFID technology, new generations of tags, referred to as computational tags, are armed with abilities of sensing and computing, and become programmable, e.g., WISP tag [1], UMASS Moo tag [27].

In these applications, categorizing the objects (tags) to be monitored into groups is a common practice for efficient management, especially when the system scales (e.g., libraries, supermarkets). A bootstrapping functionality to enable group-wise object management is to inform each object its group data (e.g. group ID, other related group information), which is named *group labeling*. For example:

- Over-the-air reprogramming on computational RFID tags [21] [22]. These tags work in the same region on a variety of sensing tasks, e.g., temperature, humidity monitoring, and intrusion detection. We regard the tags carrying out the same mission as belonging to the same group. In such scenario, it is necessary to maintain and upgrade firmware of tags wirelessly. Since the firmware for tags in different groups is usually different, the system administrator must reprogram categorized tags correctly. That is to say, data for one group should not be received by tags in the other groups.
- Group ID-enabled applications. When the administrator needs to frequently check the status of the expiry-date-sensitive objects, grouping the objects (tags) with the similar expiry date is necessary, wherein group IDs play a important role. Specifically, if the tags with the similar expiry date share the same group ID, the reader can send the required data together with the group ID once to all group members, which not only sharply reduces the communication cost in comparison with the traditional unicast transmission, but also is prerequisite of diverse queries in RFID systems, such as tag estimation [3] [11], top-k query [13] [19] and missing tag detection [25].

While due to the nature of RFID, a tag has neither information of the other tags nor its group, it thus does not know which data is only for its group. In this context, the group labeling is called for to correctly tell each tag the data for its group and facilitate the tag management illustrated above.

The RFID group labeling problem is to *inform all tags of their associated group data in an RFID system correctly and time-efficiently*. Despite its importance, this cornerstone service is largely under-investigated. The work [16] can solve this problem by polling each tag. BIC [26] can label tags by exploiting the singleton slots. While they spend too much time sending either many tag IDs or same group data repetitively, as only one tag is labeled per slot. The single-seed protocol CCG [10] leverages the slots which multiple tags of the same group are mapped to label multiple tags with one slot. It, however, wastes much time on the transmission of empty slots and slots mapped by multiple tags from different groups, and is inefficient for the low useful slot probability in a single indicator vector. For example, consider 10^3 tags are evenly partitioned into $4, 8, 10$ groups, the probability that a slot cannot be used to label tags in CCG exceeds 0.6, which leaves huge space for improvement.

This work is supported in part by Beijing Institute of Technology Research Fund Program for Young Scholars and Young Elite Scientist Sponsorship Program of China and Chongqing Key Laboratory of Mobile Communications Technology, and was supported in part by a Canada NSERC Discovery Grant and an NSERC E.W.R. Steacie Memorial Fellowship. Part of the work of R. Zhang, S. Zhang and L. Chen is supported by the NSF of China (no. 61801064, no. 61502330), and the CNRS PEPS project MIRFID. Corresponding author: Jihong Yu.

J. Yu is with School of Information and Electronics, Beijing Institute of Technology, Beijing, China (jihong.yu@bit.edu.cn).

J. Liu are with School of Computing Science, Simon Fraser University, Burnaby, British Columbia, Canada (jcliu@sfu.ca).

R. Zhang is with Information Engineering College, Capital Normal University, Beijing, China (zhangrr@cnu.edu.cn).

L. Chen is with Laboratoire de Recherche en Informatique, Université Paris-Sud and Institut Universitaire de France, Orsay, France (chen@lri.fr).

W. Gong is with School of Computer Science and Technology, University of Science and Technology, Hefei, China, and with School of Computing Science, Simon Fraser University, Burnaby, Canada (weigong@ustc.edu.cn).

S. Zhang is with College of Mathematics, Taiyuan University of Technology, Taiyuan, China (zhangshurong@tyut.edu.cn).

This paper presents a multi-seed-based protocol enabling multiple mappings from tags to slots so that the reader can pick up the most informative slots among all mappings for the data transmission and the efficiency is thus improved. The key challenge lies in how to find these slots while achieving seed assignment with low complexity. The superiority and novelty of our method compared with the existing ones is four-fold:

1) Empty slots and those mapped by multiple tags from different groups under one seed which are wasted in [10], can be used to label tags with another seed in our method.
2) The impact of multiple mappings on tag collisions of different groups is weaken. With different seeds a tag mapped to multiple slots actually receives its group data only in one slot and will keep silent, reducing collision probability of different groups in the subsequent slots.
3) Collision slots with tags from same groups instead of only singleton slots or empty slots in the existing work [4] [14] are exploited in our method, improving time efficiency. Moreover, a k-good slot that can label k tags of the same group, can become k^+-good where k and k^+ are constant and $k^+ > k$, significantly reducing the labeling delay.
4) This paper is the first work formally proving NP-hardness of the formulated problem arising from the application of multiple seeds and designing the approximation algorithms to for the group labeling problem, which makes the mathematical nature of our work completely different from the existing ones and more challenging.

Our contributions are articulated as follows: 1) We use a multi-seed approach to achieve efficient group labeling wherein we find NP-hardness of the *Seed Assignment Problem* (SAP) arising from the employment of multiple seeds to the group labeling. This result reveals the underlying difficulty of the group labeling problem, which has not been found before. 2) Due to the NP-hardness of the problem, we first design an approximate seed assignment algorithm, with a competitive ratio 0.632, which selects the slot with the most tags from the same group each time among all slots and assigns the corresponding seed to this slot. Then by leveraging the characteristic that a tag only receives its associated group data in one slot, we develop another two simplified algorithms, namely c**-search-I** and c**-search-II**. By exploiting to a greater extent that the slots are originally useless but will become useful, they achieve the comparable performance with less complexity. 3) We develop a unified group labeling protocol, named GLMS, to consolidate each of AA, c-search-**I**, c-search-**II** with concrete communication mechanism for the reader and tags. We also investigate the optimal parameter configuration.

Our multi-seed protocol generalizes the existing single-seed protocols with remarkably better performance. Our test results show that GLMS achieves a gain of up to 34.2% in terms of the group labeling time.

II. Problem Formulation and Motivation

We study an RFID system of one or multiple readers and a number of tags, wherein the tags are partitioned into multiple groups and the readers are connected via high-speed channels with a back-end server of powerful computing capability.

TABLE I
Main Notations

Symbols	Descriptions
k-good	Useful slot with k tags
N	The number of tags in the system
G	The number of groups
g, d_g	Group index, data for group g
N_g	The number of tags of group g
f, l	Frame size, the number of seeds
s_i	The i-th seed
C_{ij}	The set of tags mapped to j-th slot under s_i
m	The number of labeled tags in the current round
z	The number of chosen useful slots in the current round
u	Time efficiency
N'	Unlabeled tags in current round
N'_g	Unlabeled tags of group g in the current round
G'	The number of groups with unlabeled tags
$\overline{f}$	Upper bound of f
$\overline{l}$	Upper bound of l

We regard the server and the reader(s) as a single entity called *the reader* for simplicity [14] [7]. Generally, the tags have user-defined memory to achieve the writing and storage of the user-defined data [8]. Moreover, we assume that the reader has the IDs of all tags in the system, commonly in designing application-oriented protocols, e.g., missing tag event detection [14] [29] and information collection [16] [4]. To streamline the presentation, we first consider the single-reader case and discuss the multi-reader case later.

Consider a set $X=\{x_1, x_2, \cdots, x_N\}$ of N tags whose IDs are recorded in the reader divided into G disjoint groups. Suppose the size of group g ($1 \leq g \leq G$) is N_g and we have $\sum_{g=1}^{G} N_g = N$. We denote by d_g the data for group g ($1 \leq g \leq G$). In this paper, we are interested in addressing the following problem: *The group labeling problem is to devise a protocol to send each group data correctly to all its members (tags) within the minimum time.* By correctly, we mean that the data for one group should not be received by tags of the other groups. The performance metric is the communication cost between the reader and the tags. Table I summaries main notations used in the paper.

A. Single-Seed vs. Multi-Seed

The communication between the reader and tags follows the frame slotted Aloha protocol [6]: the reader initiates communication first by broadcasting commands containing the parameters, such as frame size f, l random seed(s) s_i with $i \leq l$. In the existing single-seed protocols where $l=1$, each tag uses its ID and the received seed to generate **one** pseudo random number via hash function $H(ID, s_1)$ and then maps itself to the slot $(H(ID, s_1) \mod f)$ in the frame. On the contrary, in our multi-seed protocol where $l \geq 1$, each tag holds **multiple** pseudo random numbers with l different seeds and is mapped to l slots in the frame and the most useful slot will be chosen by the reader to send data as introduced shortly.

In this paper, we make the following definitions on slot states: 1. *Empty slot*: Consider an arbitrary slot, if no tag is mapped to this slot; 2. *Heterogeneous slot*: if multiple tags from different groups are mapped to this slot. 3. *Useless slot*: if this slot is either empty or heterogeneous. If the reader sends data in such a slot, either no tag receives data or tags from one

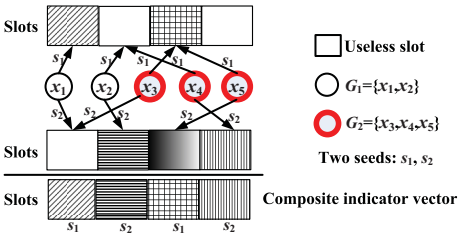


Fig. 1. Exemplifying the motivation: the shaded rectangles typify useful slots.

group receive data of another group, which should be avoided in the group labeling problem; 4. *Useful slot*: if one or multiple tags from the same group are mapped to this slot. In such a slot, the reader can send data to one or multiple tags from the same group. 5. *Reparable slot*: A slot is reparable if it becomes useful from a heterogeneous slot as the protocol runs, which will happen when tag(s) blent with the others from another group stays silent after being assigned useful slots.

B. Motivation

As an indicator vector constructed from a single mapping generates limited useful slots, much time is wasted on the transmission in the useless slots. If multiple seeds are used to generate multiple mappings, the reader can pick up the most informative slots from them to build a composite indicator vector (CIV), reducing the number of the useless slots. Intuitively, assume a slot in a single indicator vector is useful with the probability of 0.5, then with l seeds used to map the tags this probability is $1-(1-0.5)^l$, which quickly approaches 100% with the increase of l.

In addition to increase the number of useful slots, using multiple seeds can also contribute to more labeled tags. Let k-good define a useful slot with k tags. A slot may be k-good under one seed but k^+-good under other seeds where $k^+>k$, which can be interpreted from the following toy example.

Example 1. Consider an RFID system with two tag groups $G_1=\{x_1, x_2\}$ and $G_2=\{x_3, x_4, x_5\}$ and suppose a frame of four slots and two seeds s_1, s_2. From Fig. 1 where the shaded rectangles stand for the useful slots, we find just partial slots useful after either mapping, but a CIV of all slots being useful can be built by selecting the most informative slots from two mappings. Specifically, designating s_1 for the first and third slots, and s_2 for the second and fourth slots, we can build a CIV indicating the seed assignment for each slot so that all slots to be executed become from useless ones (e.g., the 2nd slot under s_1) to useful ones (e.g., the 2nd slot under s_2) and from 1-good one (e.g., the 3rd slot under s_2) to 2-good one (e.g., the 3rd slot under s_1).

Motivated by the above observation, we design a series of seed assignment algorithms to build the CIV, and develop a unified group labeling protocol, named GLMS, to consolidate each algorithm with the concrete communication mechanism for the reader and tags, respectively. Note that the designed seed assignment algorithms are used in the first phase of the group labeling protocol GLMS. In the following, we first introduce the group labeling protocol and elaborate how to build the CIV, subsequently.

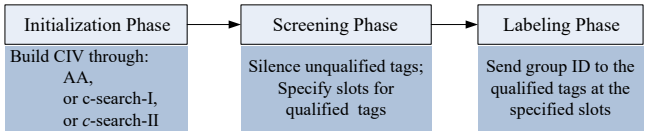


Fig. 2. The process of GLMS: Initialization Phase, Screening Phase, and Labeling Phase in sequence.

III. Group Labeling Protocol with Multiple Seeds (GLMS)

The execution of the protocol GLMS consists of multiple rounds, each having three phases referred to as *initialization phase*, *screening phase* and *labeling phase*, respectively. The reader first uses one of the seed assignment algorithms, namely AA, c-search-I, and c-search-II, to be introduced in Sec. IV to build a CIV that determines a unique tag-seed-slot relationship. In the screening phase, the reader sends the CIV to inform each active tag of whether and when it is scheduled to receive its associated group data. In the labeling phase, the reader transmits group data in the designated slots to the eligible tags. If a tag receives its associated group data, it will keep silent in the subsequent rounds. The process of GLMS and the core function of each phase are illustrated in Fig. 2.

Protocol Description. Consider an arbitrary round in the execution of the protocol GLMS. Let N', N'_g denote the number of the remaining overall unlabeled tags and that of unlabeled tags of group g in the beginning of this round, respectively. And denote by G' the number of the groups with unlabeled tags. If it is the first round, it holds that $N = N'$ and $G = G'$. The l seeds denoted as s_i, $1 \le i \le l$, are used in this round to generate the CIV of f slots. Our multi-seed protocol GLMS is shown in Algorithm 1 and Algorithm 2.

1) Initialization Phase: Given l seeds and the frame size f, the CIV can be compounded from l mappings, each involving a different seed. How the values of f and l are chosen will be analysed in Sec. V on the parameter optimisation. Specifically, in the i-th mapping, we employ seed s_i to map each active tag to one of f slots in the frame. With all l seeds used, the reader records l vectors, each consisting of f cells storing tags mapped to the corresponding slots. Using one of the seed assignment algorithms introduced in Sec. IV, the reader can designate one seed for each slot in the CIV maximizing the time efficiency in this round.

More specifically, based on the seed assignment, the reader builds a CIV of f slots each of which corresponds to a slot in the frame at the same position and stores the index of the assigned seed. If designating seed s_i for a slot j, the reader stores i that is the index of s_i in the j-th slot of the CIV. If a slot is still useless after l mappings, the reader sets its value in the CIV to zero. Consequently, the positions of non-zero value in the CIV stands for the useful slots of the frame. As there are l seeds, we need $\lceil \log_2(l+1) \rceil$ bits to record one seed's index, that is to say, the length of the CIV is $f \cdot \lceil \log_2(l+1) \rceil$.

Note that if a tag is mapped to a useful slot as specified in the CIV, we refer to this slot as *the useful slot for this tag*.

2) Screening Phase: The reader broadcasts a message containing the built CIV, the frame size f and l seeds $s_1, s_2, \cdots, s_l$. Upon receiving the message, each tag can extract two pieces of information from the CIV: One is whether

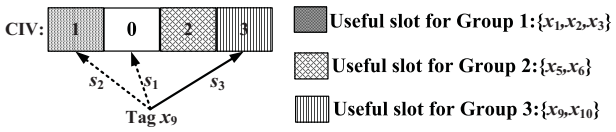


Fig. 3. Interpreting indicator vector: s_1, s_2, s_3 are the used seeds.

Algorithm 1: GLMS for the reader

// Phase one - the CIV construction
Generate l seeds s_1, s_2,..., s_l randomly
Map the unlabeled tags into l slots and generate $f \times l$ cells each recording a set of the tags mapped to the corresponding slot
Build CIV via AA or $c-$search-I or $c-$search-II; record the number of non-zero slots in the CIV z
// Phase two - CIV transmission
Issue a frame start command, transmit CIV and the corresponding frame size f and l seeds
// Phase three - tags labeling
for $i = 1$ *to* z **do**
    Issue slot-start command
    Broadcast the corresponding group ID to the tags mapped to the i-th homogeneous slot
end
Update the set of the unlabeled tags and initiate the next round

the tag is eligible to receive its group data in this round. Specifically, each tag can employ the received l seeds to select l slots in the frame and knows the corresponding l positions it is mapped to in the CIV. Based on the rule of generating the CIV, if a tag is mapped to the j-th position in the CIV under seed s_i and the value in that position is i, then the tag regards slot j as the useful slot for it. In case that the conditions can be satisfied under multiple seeds, the tag only selects the slot with the smallest value of j. While if a tag fails under all seeds, it does not participate in any activity until next round.

The other one is which slot a qualified tag should actually wait for its group data. Because the CIV may contain zero elements which stand for the useless slots, the reader needs to remove the corresponding slots before starting the frame to transmit group data for saving time. The key here is that the tag must know which slots are removed. To that end, we use the ordering approach [16]. Assume slot j is the useful one for the tag, the tag first checks every position before the position j in the CIV. If there exist $\hat{j}$ non-zero elements, the tag will select $(\hat{j}+1)$-th slot to receive its group data and $\hat{j}<j$.

Let us see an example shown in Fig. 3. Consider an arbitrary tag x_9. With seeds s_1, s_2, s_3, x_9 is mapped to the 2nd, 1st and 4th slots. After checking the corresponding positions in the CIV, x_9 finds only the 4th element equal to 3 that is the index of s_3, so it regards slot 4 as its useful slot. Furthermore, as there exist two non-zero elements before the 4th position in the CIV, x_9 will wait for its group data at slot 3 in the labeling phase. Therefore, only the three useful slots will be executed in the labeling phase instead of four in the original frame.

3) Labeling Phase: After the qualification test in the screening phase, only the eligible tags partake in this phase. As knowing all tag IDs and the CIV, the reader knows the order of the slots actually selected by the eligible tags. Assume there are z non-zero positions in the CIV, the reader initiates a labeling frame of z slots and sends the corresponding group data at each slot to the eligible tag(s) for which this slot is useful. As the tag(s) in each slot comes from the same group, they can be labeled simultaneously. On the other hand, each tag learns from the CIV at which slot the reader will transmit its group data and can thus receive the data at that slot.

Algorithm 2: GLMS for tags

Receive the CIV and the corresponding frame size f and l random seeds
Compute l mapped slot number $sn[i] = H(f, ID, s_i)$
Initialize the current slot number $csn \leftarrow 1$ and current random seed index $ci \leftarrow 0$
while *TRUE* **do**
    Wait-for-slot-start().
    $j \leftarrow$ the number of zeros in the first csn positions in CIV
    $ci \leftarrow CIV[csn + j]$
    if $(csn + j) == sn[ci]$ **then**
        Store the received Group ID.
    end
    $csn \leftarrow csn + 1$
end

For instance, recall the example in Fig. 3, the reader actually initiates a frame containing three useful slots in Fig. 3. It can label tags x_1, x_2, x_3 by sending ID of group 1 in the slot 1, and label tags x_5, x_6 and x_9, x_{10} in the slots 2 and 3, respectively.

After the current round, the reader moves to the next round, which is identical except that the labeled tags will keep silent. That is, only the unlabeled tags attend the next round. The above process repeats round after round until all tags receive their associated group data.

In what follows, we start formally presenting the seed assignment algorithms used to build the CIV.

IV. Seed Assignment Algorithms

The key of our multi-seed method lies in the seed assignment arising in building the CIV. Specifically, given l seeds s_i $(1\leq i\leq l)$ and the frame size f, the reader needs to designate one seed for each slot in the CIV and inform each tag of the seed assignment through sending the CIV. Therefore, if the CIV is built the tags mapped to each slot are deterministic.

More specifically, recall that the CIV of f slots is compounded from l mappings, there are $l \times f$ cells in total each of which records a set of the tags mapped to the corresponding slot, as shown in Fig. 4. C_{ij} stands for the set of the tags mapped to slot j under seed s_i for $1\leq i\leq l$ and $1\leq j\leq f$, and $1\leq I_j\leq l$ denotes the index of the seed finally assigned for slot j in the CIV, and C_j is the set of tags that will be mapped to slot j under seed s_{I_j} following the built CIV. Note that since l seeds are used and zero represents useless slots, we need $\lceil \log(l+1) \rceil$ bits to stand for each seed index I_j. Moreover, it may happen that $I_j = I_{j'}$ for $j \neq j'$ because a seed may be assigned to multiple slots in the CIV.

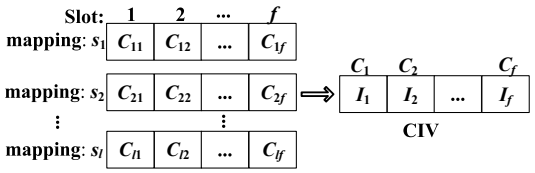


Fig. 4. Exemplifying the seed assignment problem: C_{ij} is the set of the tags mapped to slot j under seed s_i; I_j denotes the index of the seed assigned for slot j in the CIV; C_j is the set of the tags mapped to slot j under seed s_{I_j}.

As a tag can be mapped to l positions under l different seeds, slots from multiple mappings may share the same tags, that is, $C_{ij} \cap C_{i'j'} \neq \varnothing$ for $i' \neq i$ and $j' \neq j$. Define a set comprising tags from the same group as pure set which is equivalent to a useful slot. Define the time efficiency u as the number of tags labeled per unit time. Recall that if a seed is designated for a slot, then the tags mapped to this slot under this seed are deterministic. In this sense, we should carefully assign seeds such that the time efficiency u can be maximized.

Given a seed assignment, let z be the number of yielded useful slots and let $m = |\cup_j C_j|$ be the size of the union of the tags mapped to the useful slots. Let t_0 and t_g denote the time for the reader to transmit one bit and data for group g, respectively. Without loss of generality, we assume data size for each group is identical. With (2), we formally define the following seed assignment problem.

Problem 1 (Seed assignment problem)**.** *Given $l \times f$ sets of the tags C_{ij} for $1 \leq i \leq l$ and $1 \leq j \leq f$, and define S as the collection of the seeds assigned to each slot in the CIV, the seed assignment problem is to seek S satisfying*

$$S = \underset{s_{I_j}}{\operatorname{argmax}} \frac{|\cup_j C_j|}{t_g(a+z)},$$

where $a = f\lceil \log_2(l+1) \rceil t_0/t_g$. That is to say, given the seeds and the frame size, the reader seeks an optimum collection S of the seeds which will maximize the time efficiency u.

Problem 1 performs combinatorially, which is usually NP-hard. The challenge here lies in how to prove its NP-hardness. In the following, we formally state the NP-hard observation and its proof.

Theorem 1. *Problem 1 is NP-hard.*

Proof. For clarity, we just outline the proof here and the complete proof is provided in Appendix A. To study the hardness of Problem 1, we prove it polynomially reducible from the Maximum coverage problem [20] which is a classic NP-hard problem. Given h sets and an integer $k \leq h$ with which we need to solve the Maximum coverage problem, the polynomial reduction comprises three steps: First, we replicate each set k times and obtain $h \times k$ sets. Second, we introduce k dummy sets to guarantee that each slot in the CIV is assigned only one seed. Third, we prove that u reaches its maximum only when k sets are chosen in Problem 1. □

Due to the NP-hardness of SAP, in what follows, we design a series of algorithms to approach the optimal time efficiency. Specifically, we first design an approximation algorithm (AA) and develop two simplified algorithms with the less complexity but good performance on the top of AA.

Algorithm 3: Approximation algorithm for Problem 1

Input : s_i, f
Output: u_{max}, tags in picked slots C, seed assignment S

Initialisation:
$C, S \leftarrow \varnothing; R, z, u_{max} \leftarrow 0; H \leftarrow (C_{ij})_{l \times f}$
while $j_1 \leq f$ **do**
  // Search the most useful slot
  for $j = 1$ *to* f **do**
    for $i = 1$ *to* l **do**
      if C_{ij} *is useful and* $|C_{ij}| > R$ **then**
        $R \leftarrow |C_{ij}|, I \leftarrow i, J \leftarrow j$
      end
    end
  end
  // Select the seed contributing to the most useful slot
  if $\frac{|C \cup C_{IJ}|}{t_g(a+z+1)} \geq u_{max}$ **then**
    $S \leftarrow S \cup (s_I, J)$ */* Assign seed s_I to slot J */*
    $C \leftarrow C \cup C_{IJ}$, and $z \leftarrow z + 1$
    $u_{max} \leftarrow \frac{|C \cup C_{ij}|}{t_g(a+z)}$
  else
    Stop
  end
  // Clear the slots at J-th column in Fig. 4 and deduct the tags in the picked slot from the remaining slots
  for $j = 1$ *to* f **do**
    for $i = 1$ *to* l **do**
      if $j == J$ **then**
        $H \leftarrow H/C_{ij}, C_{ij} \leftarrow \varnothing$
      else
        $C_{ij} \leftarrow C_{ij} - C_{IJ}$
      end
    end
  end
  if $H == \varnothing$ **then**
    Stop
  end
end
Return u_{max}, C, S

A. Approximation Algorithm

1) Motivation: Recall the Problem 1 that seeks the seed assignment to maximize time efficiency u, we can achieve this objective from two directions. On the one hand, we want to use fewer useful slots, i.e., minimizing z, while maximizing the number of the tags m involved in these used useful slots. Observing the waste of heterogeneous slots (c.f. Sec. II-B) in the prior work, we, on the other hand, hope to design an algorithm that is able to exploit the heterogeneous slots that can become useful as the algorithm runs.

2) Overview: Define the most useful slot as the useful slot with the most tags from the same group. The core idea of AA lies in that each time the reader selects the seed contributing to the most useful slot to maximize the time efficiency u. Note

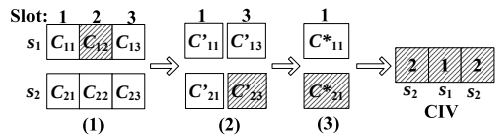


Fig. 5. AA: the streak represents the unselected most useful slot.

that there is a unique seed-tag-slot mapping, that is, given any two of them, we can fix the third one. Since a set of the tags (c.f. Fig. 4) is indexed by the used seed and the mapped slot, once a most useful slot is found the reader assigns the corresponding seed to this slot and knows the tags mapped to this slot, which are referred to as covered tags here.

Moreover, to enable the utility of heterogeneous slots, the reader first deducts the covered tags from the remaining nonempty slots including both heterogeneous and useful slots, and then checks their states and picks the most useful one among them. The rationale behind this is that each covered tag will stay silent after its corresponding most useful slot so that actually it will not be blent with tags in the subsequent slots under all mappings, which enables the conversion of a subsequent heterogeneous slot into a useful one. Note that we refer to such a heterogeneous slot as *reparable slot*.

3) Algorithm Description: Formally, we illustrate the AA in Algorithm 3 with the input of l seeds and the frame size f. It is easy to check that the computation complexity of AA is $O(l \cdot f^2)$. The main procedures of AA are summarized below.

- Each time the reader
  - picks the most useful slot which the most uncovered tags are mapped to and brings the most gain in time efficiency u. (Line 4-12 in Algorithm 3)
  - records the subscripts of the chosen slot standing for which seed will be assigned to this slot. (Line 13)
  - records the tags in the chosen slot and marks them as covered, and removes them from the remaining slots. Since only one seed should be assigned to each slot in the CIV, the slots under the other mappings but in the same column (c.f. Fig. 4) as the chosen most useful slot would be emptied. (Line 19-27)
- The algorithm stops if there is no useful slot or no useful slot contributing to the greater time efficiency.
- The algorithm outputs the seed allocation for each slot in the CIV and a collection of the covered tags, with which the time efficiency u is maximized under the given input.

After executing Algorithm 3, the reader builds a CIV and knows which tags can be labeled in which slots. Specifically, if a set C_{ij} in the useful slot is chosen, then the reader designates seed s_i for the slot j and sets the value of the slot j in the CIV to i. In case that all sets in the column j in Fig. 4 are not chosen, the reader sets the slot j's value to zero in the CIV.

Next, we illustrate AA in Fig. 5 with 2 seeds and a frame of 3 slots. First, the reader finds C_{12} the most useful, then it assigns s_1 to slot 2 in the CIV and empties C_{i2} while removing the tags in the intersections between C_{12} and the others, yielding C'_{ij}. Repeating the operations, the reader finds C'_{23} the most useful via searching from the columns 1 and 3, and then C^*_{21} from the columns 1 in sequence. Finally, the reader builds the CIV as shown in Fig. 5. To evaluate algorithm performance, we derive the competitive ratio of the algorithm.

Algorithm 4: c-search-**I** for Problem 1

Input : s_i, f, c
Output: u_{max}, tags in picked slots C, seed assignment S

1 **Initialisation:**
  $C, S \leftarrow \varnothing; R, z, u_{max} \leftarrow 0; H \leftarrow (C_{ij})_{l \times f}$
2 **while** $j_1 \leq f$ **do**
3 Choose c columns out of unselected ones randomly
4 *// Search the most useful slot from the c columns: define $j_{j'}$ as the j'-th chosen column*
5 **for** $j' = 1$ *to* c **do**
6 **for** $i = 1$ *to* l **do**
7 $C_{ij_{j'}} \leftarrow C_{ij_{j'}} - C_{IJ}, H \leftarrow H/C_{iJ}$
8 **if** $C_{ij_{j'}}$ *is useful and* $|C_{ij_{j'}}| > R$ **then**
9 $R \leftarrow |C_{ij_{j'}}|, I \leftarrow i, J \leftarrow j_{j'}$
10 **end**
11 **end**
12 **end**
13 Conduct the operations as lines $12 - 14$ in Alg. 3
14 **end**
15 Return u_{max}, C, S

Lemma 1 (Competitive ratio of Algorithm 3)**.** *Let u_{opt} denote the optimal time efficiency of Problem 1, it holds for the time efficiency u_{max} of Algorithm 3 that $u_{max} \geq 0.632 u_{opt}$.*

Proof. The proof is detailed in Appendix B. □

B. Simplified Algorithms

For better scalability to the system scale, we here present two simplified algorithms, namely c-search-**I** and its improved version: c-search-**II**, to reduce the complexity of AA while achieving the comparable performance.

*1) c-search-**I**:* The key difference of c-search-**I** from AA consists in locally searching the most useful slot among the c columns in Fig. 4 chosen ***randomly*** each time instead of global searching among all f columns in AA. At first glance, this simplified operation would degrade the performance significantly, but besides the less complexity, another advantage by this is curing more heterogeneous slots, which benefits to the increase in time efficiency. Look at an example with the frame size f and $c \leq f$. Assume that the first most useful slot in AA occurs at one of the mappings in f-th column of Fig. 4, then none of heterogeneous slots can become useful. This is because a tag mapped to a heterogeneous slot can be eliminated from this slot only when this heterogeneous slot is later than the most useful slot for this tag. While in c-search-**I**, if we find the first most useful slot in $f/2$-th column by locally searching among c randomly chosen columns, then we can exploit the subsequent reparable slots.

We list c-search-**I** in Algorithm 4 with a new input c and summarize the main procedures as below: Each time the reader

- chooses c columns from unselected ones randomly, containing $c \cdot l$ slots.

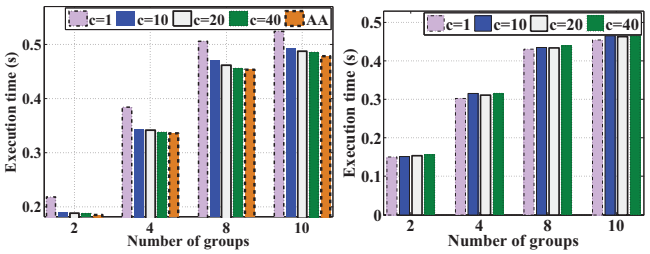


Fig. 6. Impact of c in c-search-**I**.

Fig. 7. Impact of c in c-search-**II**.

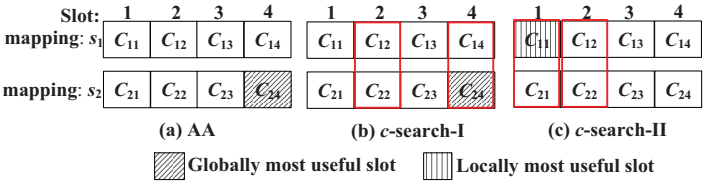


Fig. 8. Difference among three algorithms: s_1 and s_2 are two seeds.

Algorithm 5: c-search-**II** at $c = 1$ for Problem 1

while $j_1 \leq f$ **do**
  // Search the most useful slot from the j_1-th column
  for $i = 1$ *to* l **do**
    $C_{i,j_1} \leftarrow C_{i,j_1} - C_{IJ}, H \leftarrow H/C_{iJ}$
    if C_{i,j_1} *is useless* **then**
      $C_{i,j_1} \leftarrow \varnothing$
    else if $|C_{i,j_1}| > R$ **then**
      $R \leftarrow |C_{i,j_1}|, I \leftarrow i, J \leftarrow j_1$
    end
  end
  The remaining steps are the same as c-search-**I**
end

- removes the covered tags from these chosen slots.
- picks the most useful slot among the slimmed-down $c \cdot l$ slots, which achieves the most gain in time efficiency u.
- records the subscripts of the chosen slot standing for which seed will be assigned to which slot.
- records the tags in the most useful slot picked and marks them as covered.

Next, we illustrate the influence of c on the performance. Example 2. In the experiment, we partition 1000 tags evenly into G=2, 4, 8, 10 groups, and vary c from 1 to 40. Fig. 6 shows that the time overhead at c=40 is the least, which is very close to AA. For the tradeoff between the complexity and performance, we will set c=40 in the simulation in Sec. VI.

*2) c-search-**II**:* As described above, c-search-**I** achieves the comparable performance with the less complexity, but it may fail to exploit the reparable heterogeneous slots furthest. For example, if the first most useful slot in c-search-**I** arises in $f/2$-th column among c randomly chosen columns, then we cannot exploit the potential reparable slots in the first $(\frac{f}{2}-1)$ columns. To address the issue in c-search-**I**, we propose an improved algorithm, named c-search-**II**, pursuing less complexity but better performance than c-search-**I**.

The main difference from c-search-**I** is that c-search-**II** chooses c columns among the unselected columns ***in the ascending order of the column number*** instead of randomly. For instance, assume c=10, we choose the columns 1—10 as the candidates (c.f. Fig. 4). In the case that columns 1, 3 and 4 have been chosen previously, we will select columns 2 and 5—13. Next, we would like to take an example to explain the main differences among AA, c-search-**I** and c-search-**II**.

Example 3. We show the first round operation of the three algorithms in Fig. 8 where we suppose c=2 in two simplified algorithms. Specifically, AA finds C_{24} the most useful slot

Algorithm 6: Seeking the optimal f and l

Input : N', G', $step$, t_1 and t_{gid}
Output: u^*, f^*, l^*
Initialisation: $\overline{f} = \infty, \overline{l} = \infty, u^* = 0, Q = 0$
while $f \leq \overline{f}$ *and* $l \leq \overline{l}$ **do**
  Execute Algorithms 3 or 4 or c-search-**II**
  $u = u_{max}$ returned from the executed algorithm
  $Q = Q + 1$, find f_{q^*}, l_{q^*} with $\underset{1 \leq q \leq Q}{\operatorname{argmax}}\, u(f_q, l_q)$
  $f^* = f_{q^*}$, $l^* = l_{q^*}$, $u^* = u(f_{q^*}, l_{q^*})$
  Update $\overline{f}$ with (3) and update $\overline{l}$ with (4)
  $f = f + step$, $l = l + 1$
  if $f > \overline{f}$ **then**
    if $l \leq \overline{l}$ **then**
      $f = 1 : step : \overline{f}$
    else
      Stop
    end
  else if $l > \overline{l}$ **then**
    $l = 1 : \overline{l}$
  end
end
Return optimum efficiency u^* and the optimum (f^*, l^*)

by globally searching among 2∗4 cells, while c-search-**I** first selects two columns randomly (assume that columns 2 and 4 are chosen), and searches for the most useful slot among 2∗2 cells. Differently, c-search-**II** chooses the first two columns and then searches among the corresponding 2∗2 cells. As C_{11} is found the most useful in c-search-**II**, the reparable slots in the columns 2—4 can be exploited later.

In this paper, *we will set c to 1 in c-search-**II*** and state the seed assignment process in Algorithm 5. The rationale behind the setting is that with c=1 we can employ the potential reparable slots to the greatest extent, namely those in the columns 2—f. Besides, the yielded complexity is $O(l \cdot f)$ which is less than $O(c \cdot l \cdot f)$ in c-search-**I** and $O(l \cdot f^2)$ in AA, which are listed in Table II. Under the settings as in Example 2, we show in Fig. 7 that c-search-**II** achieves good performance at c=1. We will further evaluate the performance of c-search-**II** at c=1 in Sec. VI.

V. Parameter Configuration

In this section, we investigate how to tune the used parameters in the protocol to maximize the time efficiency which is defined as the ratio of the labeled tag population size to the execution time in each round. The reason for optimizing time

TABLE II
ALGORITHM COMPLEXITY WITH l SEEDS AND THE FRAME SIZE f.

Algorithm	AA	c-search-**I**	c-search-**II**
Complexity	$O(l \cdot f^2)$	$O(c \cdot l \cdot f)$	$O(l \cdot f)$

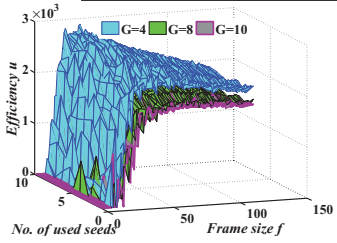


Fig. 9. c-search-**I**: u vs. f.

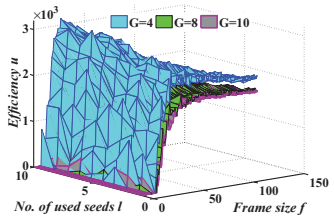


Fig. 10. c-search-**II**: u vs. f.

efficiency lies in that the higher time efficiency means that the more tags will be labeled per unit time.

The execution time of the current round, defined as T, comprises the time to transmit the CIV and group data. Denote by z the frame size in the labeling phase, for f slots are executed in the screening phase, T can be calculated as

$$T = \lceil \log_2(l+1) \rceil \cdot f \cdot t_0 + z \cdot t_g, \tag{1}$$

where t_0 and t_g denote the time for the reader to transmit one bit and group data, respectively.

Let m be the number of tags labeled in the considered round, then the time efficiency in this round, denoted by u, is

$$u = \frac{m}{T} = \frac{m}{\lceil \log_2(l+1) \rceil \cdot f \cdot t_0 + z \cdot t_g}. \tag{2}$$

Given (2) on u, we next need to find such a pair of f and l that u achieves the maximum. Note that we use u and $u(f,l)$ interchangeably in the rest of the paper. As m and z and their relationship in the protocol cannot be formulated, it is necessary to search the optimal parameter pair of f and l. For this purpose, we propose a dynamic searching algorithm.

Before introducing the searching algorithm, we first establish an upper bound for f and l, denoted by $\overline{f}$ and $\overline{l}$ respectively, in the following lemma.

Lemma 2. *For $\forall f > \overline{f}$ and/or $l > \overline{l}$, it holds that $\hat{u}(f,l) < \hat{u}(\overline{f},l)$ and $\hat{u}(f,l) < \hat{u}(f,\overline{l})$ where $\hat{u}(f,l) = \frac{N'}{\lceil \log_2(l+1) \rceil \cdot f \cdot t_0 + G' \cdot t_g}$.*

Proof. The proof is provided in Appendix C. □

Having derived the upper-bounds of f and l, we get the searching region $[1,\overline{f}] \times [1,\overline{l}]$. To speed up the searching process, we propose a dynamic searching algorithm updating the value of $\overline{f}$ and $\overline{l}$ for the $(Q+1)$-th search from the observations of the Q leading searches. Let f_q, l_q with $1 \le q \le Q$ denote each pair of f and l in the first Q searches, we can find the optimal pair (f_{q^*}, l_{q^*}) contributing to the greatest u in the first Q searches. Given f and l, executing any of AA, c-search-**I** and c-search-**II** will return u. With observations above, we update $\overline{f}$ and $\overline{l}$ by solving the following equations:

$$\text{Update } \overline{f}\text{: } u(f_{q^*}, l_{q^*}) = \hat{u}(\overline{f}, l_{q^*}), \tag{3}$$

$$\text{Update } \overline{l}\text{ : } u(f_{q^*}, l_{q^*}) = \hat{u}(f_{q^*}, \overline{l}). \tag{4}$$

Formally, IV the searching process is illustrated in Algorithm 6. With the input of the number N' of the unlabeled tags, the number G' of groups with unlabeled tags as well as the step size for f, t_1 and t_g, Algorithm 6 will output the optimal pair (f^*,l^*) and the maximum time efficiency u^*.

Considering the memory of commercial tags ranges from 32 bits to 8192 bits [8], one cannot use an arbitrary number of seeds, so we denote by l_{act} the maximum seeds a tag can store in its memory. Consequently, we need to update $\overline{l}$ in Algorithm 6 by choosing the minimum one between l_{act} and the solution of (4). Note that we set l_{act} to 10 in the simulation.

Moreover, we investigate how the frame size f influences the time efficiency u via the experiment where l_{act}=10 and N=10^3 tags are evenly partitioned into G=4, 8, 10 groups. Specifically, we snapshot the first round of GLMS with c-search-**I** and c-search-**II**. Fig. 9 and 10 show that the time efficiency u can be regarded as convex approximately with respective to f. It is thus feasible to employ the gradient method to speed up the searching for the optimum f^*.

Discussion on Multi-reader case. In large-scale RFID systems deployed in a large area, multiple readers are required to ensure the full coverage for a larger number of tags. To work with multiple readers, we leverage the same approach as [9] [17] [25] that the back-end server synchronizes and schedules all readers such that a multi-reader RFID system operates as the single-reader one. Specifically, the back-end server calculates all the parameters and constructs the CIV involved in the group labeling protocol and sends them to all readers such that the readers broadcast the same parameters and CIV to the tags.

Explanation on NP-hardness. When l_{act}=1 or the optimum l^*=1, our protocol is degraded to the single-seed protocol which does not need to assign seeds and is not NP-hard. The NP-hard seed assignment problem arises from the employment of multiple seeds. Albeit NP-hardness brings new challenges, we design a series of algorithms running in polynomial time to approximate the optimum and confirm their performance theoretically and experimentally. Moreover, the computation is done in the back-end server which is usually of the high computational capacity.

Potential implementation. Consider the implementation of the proposed protocol, programmable tags, such as those based on WISP hardware, and a USRP-based Software-Defined RFID reader are needed. In order to achieve hashing functionality, hash values are pre-stored in each tag, which is supported by WISP 4, WISP 5, and MSP430. In the scheme implementation, two commands need to be added: 1) TRANSIV that is used to transmit the CIV; 2) QUERPAR that contains the parameters used in the protocol and starts the slot.

Specifically, the reader first sends TRANSIV commend to broadcast the CIV, and then sends QUERPAR commend. Consider an arbitrary slot j. When a tag receives this commend, it starts computing the number by selecting the $\lfloor \log f \rfloor$-bit string starting from the i-th bit in the pre-stored hash value like in [4], where i is the seed value of the j-th position in the CIV. If the number equals to the current slot number, then the tag waits and receives the data sent from the reader.

VI. PERFORMANCE EVALUATION

A. Simulation Settings

We evaluate the performance of proposed approaches in comparison with the state-of-the-art solution CCG [10]. We

TABLE III
EXECUTION TIME UNDER DIVERSE N, G, l_{act}: STUDYING THE IMPACT OF l_{act}

Protocol		**Vary the number of groups G and l_{act}: (G, l_{act})**								
		(4,10)	(4,15)	(4,20)	(8,10)	(8,15)	(8,20)	(10,10)	(10,15)	(10,20)
c-search-**I**	$N = 100$	0.025	0.025	0.025	0.039	0.037	0.036	0.042	0.041	0.039
	$N = 1000$	0.333	0.311	0.31	0.444	0.426	0.424	0.473	0.456	0.449
	$N = 2000$	0.680	0.636	0.623	0.935	0.877	0.871	0.982	0.928	0.917
	$N = 5000$	1.700	1.647	1.622	2.260	2.240	2.204	2.385	2.287	2.282
c-search-**II**	$N = 100$	0.024	0.024	0.024	0.037	0.035	0.035	0.041	0.038	0.037
	$N = 1000$	0.32	0.285	0.282	0.431	0.407	0.391	0.455	0.425	0.421
	$N = 2000$	0.629	0.572	0.562	0.890	0.813	0.801	0.946	0.880	0.860
	$N = 5000$	1.581	1.459	1.433	2.209	2.001	1.978	2.331	2.127	2.204

conduct the experiments under both symmetric and asymmetric scenarios with various number of groups and group sizes. By symmetric/asymmetric, we mean that tag population size in each group is identical/different. We use the communication parameters specified in the EPCglobal C1G2 standard [6]. Specifically, the data rate from the reader to tags is 26.7 kbps, meaning it takes $37.45\ \mu s$ for the reader to transmit one bit, so we have t_1=$37.45\ \mu s$. We take group ID of $\lceil \log_2 G \rceil$ bits as group data, so we have t_g=$37.45 * \lceil \log_2 G \rceil$. Besides, we consider the time interval of $302\ \mu s$ between any two consecutive communications between the reader and tags in the computation of the execution time.

Due to the complexity of AA, we will focus on evaluating the GLMS running the simplified algorithms, namely GLMS with c-search-**I** and GLMS with c-search-**II**, but we can measure the performance of AA from Fig. 6 in the RFID system of 1000 tags. As discussed in Sec. IV-B1, IV-B2 and V, we set c=40 for c-search-**I**, and set c=1 and l_{act}=10 for c-search-**II**. Albeit using l_{act}=10, we also evaluate its impact on the performance. For simplification, we will use c-search-**I** and c-search-**II** in the figures in the below to stand for GLMS with c-search-**I** and c-search-**II**, respectively.

B. Simulation Results

The performance metric is the communication cost in terms of execution time. We first show the influence of l_{act} with diverse number of groups G and tags N in the system, and simulate symmetric scenarios with G and the group size varied and proceed to its asymmetric counterparts, subsequently.

1) Performance evaluation under different l_{act}: Here, we conduct experiments to investigate the impact of l_{act} on GLMS with c-search-**I** and GLMS with c-search-**II**. To that end, we simulate scenarios with $N = 100, 1000, 2000, 5000$ tags in the system where the tags are evenly partitioned into $G = 4, 8, 10$ groups, respectively. And the value of l_{act} are set to $10, 15, 20$. The simulation results are listed in Table III.

As shown in Table III, the increase in the value of l_{act} reduces the execution time under all settings. Specifically, the performance difference between $l_{act} = 10$ and $l_{act} = 20$ is bigger than that between $l_{act} = 15$ and $l_{act} = 20$ which is less than 3%. More specifically, we observe from the results that the most significant performance difference is about 11% arising between $l_{act} = 10$ and l_{act}=20 for GLMS with c-search-**II** when G=4 and N=2000. Considering the constraint on memory capacity of commercial tags as discussed in Sec. V and the tradeoff between the computational complexity and the execution time, we set l_{act}=10 in the subsequent simulations.

2) Performance comparison under symmetric scenario: This scenario consists of two cases: one is varying the number of the groups and the other is varying the group size.

Case 1. Here we set the total number of the tags N=12000 and G=2:2:10 with the identical group size. From the results shown in Fig. 11(a), we can observe that GLMS with c-search-**II** and GLMS with c-search-**I** perform better than CCG, with the performance gain of up to 26.8% and 15.9%, respectively. This is because we employ multiple seeds to reduce the transmission of useless slots and c-search-**II** can furthest exploit the heterogeneous slots that will become useful. Besides, increasing the number of groups renders more execution time, as more groups reduce the useful slot probability.

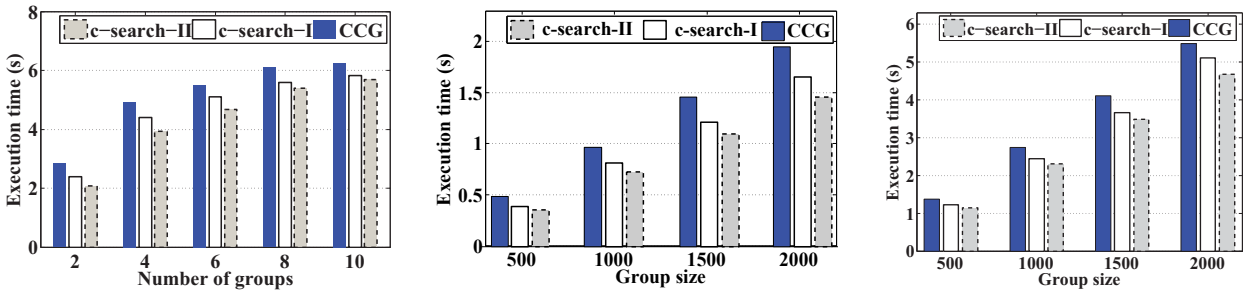
Case 2. Here we set G=3, 6 while varying the group size from 500 to 2000, and show the results in Fig. 11(b) and 11(c), respectively. As shown in the pictures, GLMS with c-search-**I** and GLMS with c-search-**II** can still finish the group labeling task within the less time than CCG. Especially, with c-search-**II**, GLMS can save time, under all group size settings, at least 22.5% when G=3, and at least 14.8% when $G = 6$.

3) Performance comparison under asymmetric scenario: This scenario consists of there cases: the first two cases are the asymmetric counterparts of the symmetric scenarios, i.e., varying the number of the groups and the group size, respectively, and we increase the asymmetry in the third case.

Case 1. In this case, we choose each group size randomly from $[100, 2000]$ while varying G from 2 to 10, and depict the results in Fig. 12(a). It can be drawn from Fig. 12(a) that c-search-**II** achieves the best time efficiency and c-search-**I** performs better than CCG, which results from the ability of our approaches of exploiting more useful slots. Specifically, c-search-**II** and c-search-**I** reduce the time up to 34.2% and 24.3%, respectively, in comparison with CCG.

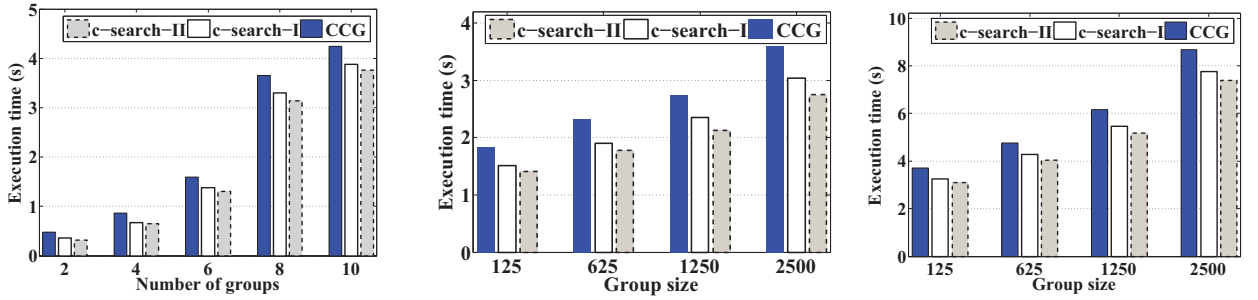
Case 2. In this case, we set the number of the groups to G=3, 6, and choose the group size randomly from $[a, 5000]$ with a=125, 625, 1250, 2500. Fig. 12(b) and 12(c) depict the simulation results, from which we observe that c-search-**II** performs best and c-search-**I** is also better than CCG. Specifically, c-search-**II** and c-search-**I** reduce the time cost up to 23.5% and 18.3% when G=3, and up to 17.2% and 12.1% when G=6, respectively, in comparison with CCG.

Case 3. In this case, we also set G=3, 6, but we synthesize the following four subcases by choosing the group size from different ranges: Subcase 1: G=3, we choose the group size randomly for the first group from $[100, 500]$, and from $[2000, 3000]$ for the others. Subcase 2: G=6, we choose the group size randomly for the three groups from $[100, 500]$, and



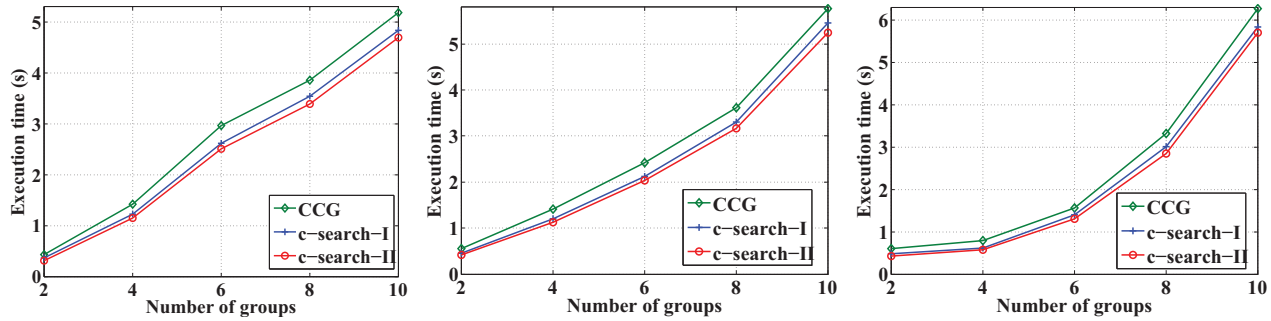
(a) Case 1: $N = 12000$ and various groups G (b) Case 2: $G = 3$, varying group size N_g (c) Case 2: $G = 6$, varying group size N_g

Fig. 11. Performance comparison in symmetric scenario with the various number of groups and group sizes: smaller execution time means better performance.



(a) Case 1: various groups G (b) Case 2: $G = 3$, varying group size N_g (c) Case 2: $G = 6$, varying group size N_g

Fig. 12. Performance comparison in asymmetric scenario with the various number of groups and group sizes: smaller execution time means better performance.



(a) Case 1: $G = 2 : 2 : 10$ and $\delta = 200$ (b) Case 2: $G = 2 : 2 : 10$ and $\delta = 400$ (c) Case 3: $G = 2 : 2 : 10$ and $\delta = 800$

Fig. 13. Performance comparison in asymmetric scenario with the normally distributed group size: smaller execution time means better performance.

TABLE IV
PERFORMANCE EVALUATION IN CASE 3

Protocol	Subcase 1	Subcase 2	Subcase 3	Subcase 4
CCG	1.1971	2.9932	1.429	3.1914
c-search-**I**	0.9567	2.5905	1.1704	2.7825
c-search-**II**	0.8719	2.3832	1.0496	2.6052

from $[2000, 3000]$ for the others. Subcase 3: G=3, we choose group size randomly for the three groups from $[100, 500]$, $[1000, 2000]$, and $[2000, 3000]$, respectively. Subcase 4: G=6, we choose the group size randomly for the first two groups from $[100, 500]$, from $[2000, 3000]$ for the last two groups, and $[1000, 2000]$ for the others, respectively. As shown in Table IV, c-search-**II** and c-search-**I** always outperform CCG. Specifically, CCG spends up to 27.6% and 20.1% time more than ours, respectively, for the transmission of useless slots.

4) Performance comparison under asymmetric scenario with other distributions: Normal distribution: We consider three cases, each of which has the same number of the groups but has the different group sizes. Specifically, we set G=2:2:10 in all cases, and each group size follows the normal distribution $N(1000, \delta^2)$ with the standard deviation δ varied from 200 in Case 1 to 400 in Case 2 and to 800 in Case 3. As shown in Fig. 13, GLMS with c-search-**II** is the fastest with the less complexity than c-search-**I**, and saves time of up to $27\%, 23\%, 28\%$ in the three cases, respectively, compared with CCG. Zipfian distribution: Each group size is sampled from $[1, 1000]$ following the Zipfian distribution $Z(1000, 1, G)$ with the number of groups G set to $\{10, 20, 50, 100\}$. The performance gain of c-search-**II** over CCG is 31%, 27%, 20%, and 8%, respectively.

VII. RELATED WORK

Group labeling is a common functionality for many RFID applications. This section presents the prior works on group labeling and the existing multi-seed/hash RFID protocols.

The feasible solutions to the group labeling problem. One straightforward solution is to use the basic polling protocol (BP) [16] where each tag is polled with its group data by the reader one by one. And BIC [26] can label each tag with its group data by informing each tag of the singleton slot when the tag should wait for its group data. These methods only employ singleton slots such that only one tag can be labeled

per slot, as a result, they spend too much time either sending many tag IDs or group data and are thus time-consuming.

To improve time efficiency, the authors in [10] devise three protocols, namely EPG, FIG and CCG. In EPG, the reader first polls all tags in the same group and sends the group data once. EPG is better than BP for less transmission of group data, however, it still wastes time sending many tag IDs. In FIG, the reader builds a Bloom filter for each group from its tags to filter out tags of the other groups. Although outperforming EPG, FIG suffers from the false positives of Bloom filter and has to deactivate the wrong tags by polling, which increases the time cost. To address this problem, CCG allows the reader send different group data to tags of multiple groups in one round. The reader sends a single indicator vector to inform tags of each slot state such that only the tags in the useful slots will receive their respective group data. Instead of using one seed in CCG, this paper employs multiple seeds to build a composite indicator vector to further improve the time efficiency.

Multi-seed/hash based protocols in RFID systems. The multi-seed/hash methods are used to address the information collection and tag monitoring tasks in RFID systems. Chen *et al* [4] employ multiple hashes to enable the fast information collection. Then, the multi-seed/hash method is used in monitoring the missing tag event and unknown tag event. Specifically, Luo *et al* [14] introduce the multi-seed method to detect missing tags in an RFID system. The works [12] [24] [25] address the missing tag detection and identification with multiple hashes. Recently, Gong *et al* [7] combine the Bloom filter with multi-seed method in order to detect the unknown tags fast and reliably. *The main novelty of our work is exploiting collision slots instead of only singleton or empty slots in these works. Moreover, we address a different group labeling problem, making the theoretical analysis completely new. We would like to emphasize that this paper is the first work proving NP-hardness of SAP arising from the application of multiple seeds and designing the approximation algorithms, which makes our work more challenging.*

VIII. Conclusion

This paper studied how to achieve efficient group labeling. To this target, we proposed a new multi-seed group labeling protocol GLMS. We found the NP-hard seed assignment problem arising from the employment of multiple seeds. To address this problem, we first introduced an approximation algorithm with the proved competitive ratio and then designed two simplified algorithms with the less complexity and comparable performance. The simulation results demonstrate the superiority of the proposed approaches.

References

[1] WISP platform. Available: http://wisp.wikispaces.com/.

[2] Barcoding Inc. How RFID works for inventory control in the warehouse. Available: http://www.barcoding.com/rfid/inventory-control.shtml.

[3] M. Chen, J. Liu, S. Chen, and Q. Xiao. Efficient anonymous category-level joint tag estimation. In *IEEE ICNP*, pages 1–10, 2016.

[4] S. Chen, M. Zhang, and B. Xiao. Efficient information collection protocols for sensor-augmented RFID networks. In *IEEE INFOCOM*, pages 3101–3109, 2011.

[5] C. Duan, L. Yang, and Y. Liu. Accurate spatial calibration of rfid antennas via spinning tags. In *IEEE ICDCS*, pages 519–528, 2016.

[6] EPCglobal Inc. Radio-frequency identity protocols class-1 generation-2 UHF RFID protocol for communications at 860 mhz - 960 mhz version 1.0.9, 2005. Available: http://www.gs1.org.

[7] W. Gong, J. Liu, and Z. Yang. Fast and reliable unknown tag detection in large-scale RFID systems. In *ACM MobiHoc*, pages 141–150, 2016.

[8] IMPINJ. RFID tag chips. Available: http://www.impinj.com/products/.

[9] M. Kodialam, T. Nandagopal, and W. C. Lau. Anonymous tracking using RFID tags. In *IEEE INFOCOM*, pages 1217–1225, 2007.

[10] J. Liu, B. Xiao, S. Chen, F. Zhu, and L. Chen. Fast RFID grouping protocols. In *IEEE INFOCOM*, pages 1948–1956, 2015.

[11] X. Liu, K. Li, A. X. Liu, S. Guo, M. Shahzad, A. L. Wang, and J. Wu. Multi-category RFID estimation. *IEEE/ACM TON*, 2016.

[12] X. Liu *et al.* A multiple hashing approach to complete identification of missing RFID tags. *IEEE TCOM*, 62(3):1046–1057, 2014.

[13] X. Liu *et al.* Top-k queries for multi-category RFID systems. In *IEEE INFOCOM*, pages 1–9, 2016.

[14] W. Luo, S. Chen, T. Li, and Y. Qiao. Probabilistic missing-tag detection and energy-time tradeoff in large-scale RFID systems. In *ACM MobiHoc*, pages 95–104, 2012.

[15] S. Qi *et al*. Double-edged sword: Incentivized verifiable product path query for rfid-enabled supply chain. *IEEE ICDCS*, 2017.

[16] Y. Qiao, S. Chen, T. Li, and S. Chen. Energy-efficient polling protocols in rfid systems. In *ACM MobiHoc*, page 25, 2011.

[17] M. Shahzad and A. X. Liu. Expecting the unexpected: Fast and reliable detection of missing RFID tags in the wild. In *IEEE INFOCOM*, pages 1939–1947, 2015.

[18] L. Shangguan *et al*. Otrack: Order tracking for luggage in mobile rfid systems. In *IEEE INFOCOM*, pages 3066–3074, 2013.

[19] B. Sheng, C. C. Tan, Q. Li, and W. Mao. Finding popular categories for RFID tags. In *ACM MobiHoc*, pages 159–168, 2008.

[20] V. V. Vazirani. *Approximation Algorithms*. Springer-Verlag Berlin Heidelberg, 2003.

[21] D. Wu, M. J. Hussain, S. Li, and L. Lu. R^2: Over-the-air reprogramming on computational rfids. In *IEEE RFID*, pages 1–8, 2016.

[22] D. Wu, L. Lu, M. J. Hussain, S. Li, M. Li, and F. Zhang. R^3: Reliable over-the-air reprogramming on computational rfids. *ACM Transactions on Embedded Computing Systems*, 17(1):9, 2017.

[23] X. Gao, P. Wang, D. Niyato, K. Yang, and J. An. Auction-based time scheduling for backscatter-aided rf-powered cognitive radio networks. *IEEE TWC*, 18(3):1684–1697, 2019.

[24] J. Yu, L. Chen, R. Zhang, and K. Wang. Finding needles in a haystack: Missing tag detection in large rfid systems. *IEEE TCOM*, 65(5):2036–2047, 2017.

[25] J. Yu, L. Chen, R. Zhang, and K. Wang. On missing tag detection in multiple-group multiple-region rfid systems. *IEEE TMC*, 16(5):1371–1381, 2017.

[26] H. Yue et al. A time-efficient information collection protocol for large-scale rfid systems. In *IEEE INFOCOM*, pages 2158–2166, 2012.

[27] H. Zhang et al. Moo: A batteryless computational rfid and sensing platform. Available: http://spqr.cs.umass.edu/moo/.

[28] K. Zhao *et al.* Emod: Efficient motion detection of device-free objects using passive rfid tags. In *IEEE ICNP*, pages 291–301, 2015.

[29] Y. Zheng and M. Li. P-mti: Physical-layer missing tag identification via compressive sensing. *IEEE/ACM TON*, 23(4):1356–1366, 2015.

Jihong Yu received the B.E degree in communication engineering and M.E degree in communication and information systems from Chongqing University of Posts and Telecommunications, Chongqing, China, in 2010 and 2013, respectively, and the Ph.D. degree in computer science at the University of Paris-Sud, Orsay, France, in 2016. He was a research fellow in the School of Computing Science, Simon Fraser University, Canada. He is currently a professor in the School of Information and Electronics at Beijing Institute of Technology. His research interests include RFID, backscatter networks, and Internet of things.

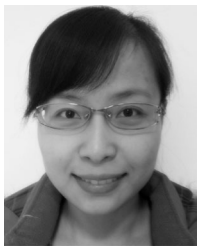
Jiangchuan Liu (S'01-M'03-SM'08-F'17) received B.Eng. (Cum Laude) from Tsinghua University, Beijing, China, in 1999, and Ph.D. from The Hong Kong University of Science and Technology in 2003. He is currently a Full Professor in the School of Computing Science at Simon Fraser University, British Columbia, Canada. He is an IEEE Fellow and an NSERC E.W.R. Steacie Memorial Fellow and a Fellow of the Canadian Academy of Engineering.

He is a Steering Committee Member of IEEE Transactions on Mobile Computing, and Associate Editor of IEEE/ACM Transactions on Networking, IEEE Transactions on Big Data, and IEEE Transactions on Multimedia. He is a co-recipient of the Test of Time Paper Award of IEEE INFOCOM (2015), ACM TOMCCAP Nicolas D. Georganas Best Paper Award (2013), and ACM Multimedia Best Paper Award (2012).

Rongrong Zhang received the B.E and M.E degree in communication and information systems from Chongqing University of Posts and Telecommunications, Chongqing, China, in 2010 and 2013, respectively, and Ph.D. degree in Computer Science at the University of Paris Descartes, France, in 2017. She was a research fellow in the school of electrical engineering and computer science, university of Ottawa, Ontario, Canada, and is an associate professor at Capital Normal University, Beijing, China. Her research interests focus on Wireless Area Body Networks for e-health applications.

Lin Chen (S'07-M'10) received his B.E. degree in Radio Engineering from Southeast University, China in 2002 and the Engineer Diploma from Telecom ParisTech, Paris in 2005. He also holds a M.S. degree of Networking from the University of Paris 6. He currently works as associate professor in the department of computer science of the University of Paris-Sud. He serves as Chair of IEEE Special Interest Group on Green and Sustainable Networking and Computing with Cognition and Cooperation, IEEE Technical Committee on Green Communications and Computing. His main research interests include modeling and control for wireless networks, distributed algorithm design and game theory.

Wei Gong (M'14) received the BS degree from Department of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China, in 2003 and the M.S. and Ph.D. degrees in School of Software and Department of Computer Science and Technology from Tsinghua University, Beijing, China, in 2007 and 2012, respectively. He was a research fellow in the School of Computing Science, Simon Fraser University, Canada and is a professor in the School of Computer Science and Technology at University of Science and Technology. His research interests include backscatter networks, mobile computing, and Internet of things.



Shurong Zhang received the PhD degree in mathematics from Shanxi University, China, in 2013. She is currently a lecturer in the College of Mathematics, Taiyuan University of Technology, China. Her research interests include graph theory and interconnection networks, computational complexity, and algorithm design and analysis.