# Optimization Problems in Throwbox-Assisted Delay Tolerant Networks: Which Throwboxes to Activate? How Many Active Ones I Need?

Fan Li, *Member, IEEE*, Zhiyuan Yin, Shaojie Tang, *Member, IEEE*, Yu Cheng, *Senior Member, IEEE*, and Yu Wang, *Senior Member, IEEE*

**Abstract**—One of the solutions to improve mobile Delay Tolerant Network (DTN) performance is to place additional stationary nodes, called throwboxes, to create a greater number of contact opportunities. In this paper, we study a key optimization problem in a time-evolving throwbox-assisted DTN: throwbox selection, to answer the questions such as "how many active throwboxes do I need?" and "which throwboxes should be activated?" We formally define two throwbox optimization problems: *min-throwbox problem* and *k-throwbox problem* for time-evolving DTNs modeled by weighted space-time graphs. We show that *min-throwbox problem* is NP-hard and propose a set of greedy algorithms which can efficiently provide quality solutions for both challenging problems.

**Index Terms**—Throwbox optimization, topology design, delay tolerant networks

## 1 Introduction

THE intermittent connectivities in Delay Tolerant Networks (DTNs) result in the lack of instantaneous end-to-end paths, large transmission delay and unstable network topology. Recent advances in DTN routing [1], [2], [3], [4] have overcome limitations in connectivity by relying on intermittent contacts between mobile nodes to deliver packets. However, lack of rich contact opportunities in many DTN applications (especially with sparse deployments) still causes poor delivery ratio and long delay of DTN routing.

One of the solutions to improve mobile DTN performance is to place additional stationary nodes, called ThrowBoxes (TBs), to create a greater number of contact opportunities [5], [6], [7], [8], [9], [10]. Throwboxes are small, battery-powered, and inexpensive devices equipped with wireless interfaces and storage. They are usually stationary, and can relay data between mobile nodes in a store-and-forward way. When two nodes pass by the same location at different time, the throwbox acts as a relay, creating a new contact opportunity. Throwboxes can operate without communication with other throwboxes. Simulations and real deployments [6], [7], [8], [9], [10] have demonstrated that introducing small number of active throwboxes can indeed improve the network performances and overall throughputs. Section 2 reviews related works on throwbox-assisted DTNs.

In this paper, we assume that a set of throwboxes is already deployed to assist the DTN. Each throwbox can be turned on or off adaptively to the dynamics of the DTN. One of the key design problems in such throwbox-assisted DTNs is throwbox selection. Given a set of locations of throwboxes, we need to answer questions like "how many active throwboxes do I need?" and "which throwboxes I should activate?". This is more critical if the budget of active throwboxes is limited. General relay placement in static wireless networks [11], [12] has been well studied. However, in DTNs, the nodes are mobile and the network topology evolves over time. These features bring new challenges and make existing relay placement algorithms not suitable in DTNs. To our best knowledge, there is not much study on throwbox deployment or selection except for [5], which studies a joint throwbox deployment and routing optimization problem. However, their focus is only on the long term average capacity. Instead, here we study how to select active throwboxes in a time-evolving and predictable DTN so that the network reliability is guaranteed or maximized.

We first model a time-evolving and predictable DTN as a weighted space-time graph which includes both spacial and temporal information about the dynamic network. We assume that (1) the network topology (contacts between nodes) could be known a priori or can be predicted from historical tracing data;[1] and (2) there is a finite set of locations for deployed throwboxes. In Section 3, we then formally define two throwbox optimization problems: (1) *min-throwbox problem*—to guarantee certain level of reliability, how many active throwboxes are needed and which one should be activated? and (2) *k-throwbox problem*—which $k$ throwboxes should be activated so that the network reliability is maximized over time? We discuss the hardness of both problems. Then, we propose a set of greedy algorithms which can efficiently provide quality solutions for these two optimization problems in Section 4. One of the proposed algorithms can particularly guarantee an $(1+\ln(r(\mathcal{G})/r(opt)))$ approximation for min-throwbox problem and an $(1-1/e)$ approximation for k-throwbox problem. Here $r(\mathcal{G})$ is the reliability of the network with all throwboxes activated and $opt$ is the optimal solution. Finally, in Section 5, we conduct extensive simulations over random time-evolving DTNs and real life DTN traces [15] to demonstrate the efficiency of the proposed methods.

## 2 Related Works

Throwbox-assisted DTNs are first proposed by [5] where a joint throwbox deployment and routing optimization problem is studied and a greedy algorithm, which relies on network flow technique to solve multiple linear programming problems, is proposed. However, their study only focuses on the *average capacity*, i.e., the maximum data rate that can be sent between two nodes in *long term*. Different from them, we consider the detailed topology evolving over time (not just average contact capacity) and aim to optimize the overall routing reliability in the network. In [6], energy efficiency inside each throwbox for throwbox-assisted DTNs is considered. An energy-efficient architecture is proposed and a real testbed is built over such architecture. An approximate heuristic is given for solving the NP-hard problem at a throwbox of meeting an average power constraint while maximizing the number of bytes forwarded. However, their energy optimization is only performed within each individual throwbox. There are also other studies on analytical models of delay distribution [7], [8] and relay strategies [9], [10] for throwbox-assisted DTNs, which do not consider throwboxes deployment or selection.

Various relay placement problems in static wireless networks have been well-studied, such as the static relay placement [11], [12] or the mobile relay planning [16], [17] in static wireless sensor networks. However, the networks studied in this paper are

- *F. Li and Z. Yin are with the School of Computer Science, Beijing Institute of Technology, Beijing 100081, China. E-mail: fli@bit.edu.cn.*
- *S. Tang is with the Jindal School of Management, University of Texas at Dallas, Richardson, TX 75080. E-mail: shaojie.tang@utdallas.edu.*
- *Y. Cheng is with the Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, IL 60616. E-mail: cheng@iit.edu.*
- *Y. Wang is with the Department of Computer Science, University of North Carolina at Charlotte, Charlotte, NC 28223. E-mail: yu.wang@uncc.edu.*

*Manuscript received 9 July 2014; revised 23 May 2015; accepted 25 June 2015. Date of publication 30 June 2015; date of current version 13 Apr. 2016.*
*Recommended for acceptance by H. M. Ammari.*
*For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.*
*Digital Object Identifier no. 10.1109/TC.2015.2451621*

1. Note that the deployed throwboxes can help collecting historical data to model the topology evolving over time. In addition, many DTN networks do have clear temporal patterns of evolving topology, such as space DTNs, DTNs formed by public buses or students who share fixed class schedules. Many such examples are given in [13], [14].

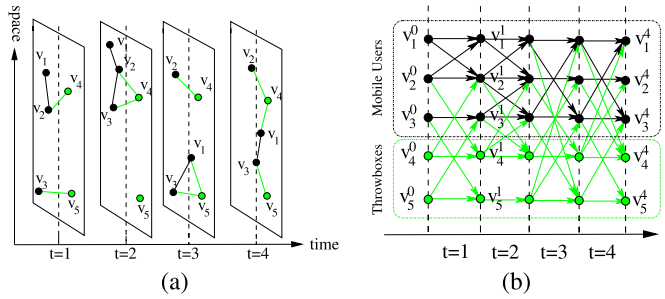



Fig. 1. (a) Example of a time-evolving throwbox-assisted DTN (two green nodes are throwboxes). (b) The corresponding space-time graph.

time-evolving DTNs where all devices are mobile and the network topology evolves over time.

In our previous work [13], [14], we have studied *topology control (TC)* problem for time-evolving DTNs, which aims to build a sparse space-time graph while guaranteeing the connectivity or reliability requirement over time. Even though those studies share the underlying space-time graph model with this paper, they are completely different problems. In TC problem, arbitrary links between two nodes at any time slots can be activated or not. In our throwbox optimization problem, if a throwbox is activated, all its spacial and temporal links over any time slots are activated. Therefore, the problems, NP-hardness proofs, and solutions are completely different.

## 3 Throwbox Optimization in Predictable DTNs

In this section, we first introduce a weighted space-time graph model and associated assumptions, and then formally define two throwbox optimization problems.

### 3.1 Models and Assumptions

In this paper, we adopt the *space-time graph* [18], [19] to model the time-evolving DTNs, since it can capture the evolving characteristics in both spacial and temporal spaces. Assume that $V_{user} = \{v_1, \ldots, v_n\}$ and $V_{throwbox} = \{v_{n+1}, \ldots, v_{n+m}\}$ be the set of all individual users (wireless devices) and the set of all deployed throwboxes in the network over a period of time $T$. Here, time is divided into discrete and equal time slots, e.g., $\{1, \ldots, T\}$. Let $V = V_{user} + V_{throwbox}$ be the whole nodes set. Since the positions of individual nodes and the topology co-evolve over time and we assume this information is known, then a sequence of static graphs can be defined over $V$ to model the interactions among nodes in the time-evolving DTN. Fig. 1a illustrates such an example with three mobile users (in black) and two potential throwboxes (in green). Some of the snapshots may not be connected at all even with all throwboxes (e.g., the first and third snapshot in Fig. 1a). This makes routing tasks over them challenging.

We can then convert this sequence of static graphs into a space-time graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, which is a directed graph defined in both spacial and temporal spaces. Fig. 1b illustrates the corresponding space-time graph of the same network. In $\mathcal{G}$, $T+1$ layers of nodes are defined and each layer has $n+m$ nodes, thus the whole vertex set $\mathcal{V} = \{v_j^t | j = 1, \ldots, n+m \text{ and } t = 0, \ldots, T\}$. Two kinds of links (spacial links and temporal links) are added between consecutive layers in the edge set $\mathcal{E}$. A temporal link $\overrightarrow{v_j^{t-1} v_j^t}$ (those horizontal links in Fig. 1b) connects the same node $v_j$ across consecutive $(t-1)$th and $t$th layers, which represents that the node can carry the message in the $t$th time slot. A spacial link $\overrightarrow{v_j^{t-1} v_k^t}$ represents a forwarding opportunity from one node $v_j$ to its neighbor $v_k$ in the $t$th time slot (i.e., $v_j$ and $v_k$ are within each other's transmission range in time slot $t$). In the figure, black links are communication links among mobile users, while green links are communication links with potential throwboxes. We assume that all throwboxes have the capacity to buffer any packet for any long time period, thus there exists temporal links of throwboxes. By defining the space-time graph $\mathcal{G}$, any communication operation in the time-evolving network can be simulated on this directed graph.

A space-time graph $\mathcal{G}$ is *connected* over time period of $T$ if and only if there exists at least one directed path between each pair of nodes $(v_i^0, v_j^T)$ ($i$ and $j$ are in $[1, n]$). Hereafter, we assume that the underlying space-time graph $\mathcal{G}$ is always connected. This guarantees that the packet can be delivered between any two nodes in the network over the period of $T$. Note that a connected space-time graph does not require connectivity in each snapshot.

To consider the reliability of lossy wireless links or inaccurate link predictions, we also define a *reliablility probability* $r(e)$ for each link $e \in \mathcal{E}$, which represents the probability of a successful data transmission over link $e$. Here, we assume that the reliablility probability of each link can be obtained through link estimation techniques at the link and physical layers [20] or mobility prediction techniques [21]. Given the reliability of each link, we can then define the reliability of a path $P$ or a structure $\mathcal{G}$. Hereafter, we consider the reliability for single-copy DTN routing where only one copy of each message is propagated in the network. Thus, the resulting propagation path of a message is basically a single space-time path in $\mathcal{G}$. Given a path $P(u, v)$ connecting nodes $u$ and $v$, the reliability of $P(u, v)$ is the product of reliability of all links in that path. For a given topology $\mathcal{G}$ (a space-time graph), we can define the *most reliable path* $P_r^{\mathcal{G}}(u, v)$ as the path from $u$ to $v$ in $\mathcal{G}$ with the highest reliability. Let $r^{\mathcal{G}}(u, v) = \Pi_{e \in P_r^{\mathcal{G}}(u,v)} r(e)$ be the reliability of path $P_r^{\mathcal{G}}(u, v)$. Then the *reliability* of the topology $\mathcal{G}$ is defined as follows

$$r(\mathcal{G}) = \min_{1 \le i,j \le n} r^{\mathcal{G}}(v_i^0, v_j^T). \tag{1}$$

Here, the reliability of the topology is the minimum path reliability among all source-destination pairs. Another alternative way to define the reliability is taking the summation of path reliabilities instead of the minimum. Notice that when $\mathcal{G}$ is given, it is easy to calculate $r(\mathcal{G})$ by using any shortest path algorithms (such as the Dijkstra's algorithm).

### 3.2 Throwbox Optimization Problems

With the helps from active throwboxes there will be more forwarding opportunities among devices, thus keeping more active throwboxes usually increases the reliability of the network. However, maintaining active throwboxes has certain cost [6]. Therefore, we may want to carefully decide how many active throwboxes we need and which throwboxes should be activated. We now formally define the *throwbox optimization problems* over the weighted space-time graph model, which have two versions: *min-throwbox problem* and *k-throwbox problem*. In both definitions, $n$ mobile users and their contact patterns are given as the input space-time graph, and the selection is only made over throwboxes.

**Definition 1.** *Given a weighted space-time graph $\mathcal{G}$ (with $n$ mobile users and $m$ potential throwboxes) and a threshold $\gamma \le r(\mathcal{G})$, the aim of* min-throwbox problem *is to find the minimum set of active throwboxes such that the space-time graph $\mathcal{H}$ formed by these throwboxes and all mobile users (which is a subgraph of $\mathcal{G}$, e.g. in Fig. 2 with one active throwbox) has a reliability larger than or equal to $\gamma$.*

The min-throwbox problem is to use minimum number of active throwboxes while guaranteeing certain level of reliability. Its solution gives both the number of active throwboxes needed and who are them.

**Definition 2.** *Given a weighted space-time graph $\mathcal{G}$ (with $n$ mobile users and $m$ potential throwboxes) and a constant $k$ ($k \le m$), the aim of*

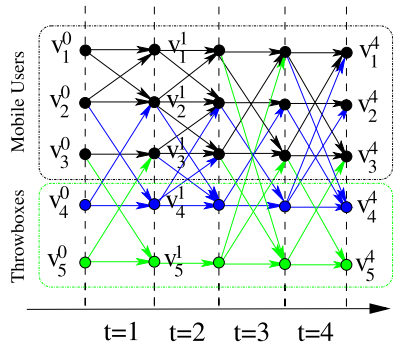



Fig. 2. Select one active throwbox (marked as blue).

k-throwbox problem *is to find $k$ active throwboxes such that the space-time graph $\mathcal{H}$ formed by these throwboxes and all mobile users has the maximum reliability.*

The k-throwbox problem aims to limit the number of usage of active throwboxes to $k$ while achieving the best reliability. The network operator may have fixed budget to activate $k$ throwboxes. The solution explores which $k$ throwboxes should be activated.

Note that both newly defined throwbox optimization problems are different from traditional relay node placement problems [11], [12], since the network is not static but evolves over time. They are also different from the topology control problems [13], [14], which aim to build a sparse subgraph while guaranteeing the connectivity or reliability. In TC, arbitrary links from any node at any time can be activated.

#### 3.2.1 Hardness

We first prove the NP-hardness of the min-throwbox problem.

**Theorem 1.** *The* min-throwbox problem *is NP-hard.*

**Proof.** We first show how to reduce the set cover problem to our min-throwbox problem. Given an instance of set cover problem where $m$ subsets $S_1, S_2, \ldots, S_m$ are defined over a set of $n$ elements $e_1, e_2, \ldots, e_n$, as shown in Fig. 3, we can construct an instance of the min-throwbox problem as follows. We treat all elements as mobile users and all subsets as throwboxes and construct a space-time graph with only two time slots , as shown in Fig. 3. For each element $e_i$, we add one temporal link in the first time slot, and add $n-1$ spacial links to all other elements in the second time slot. For each subset $S_j$, we add two temporal links and some spacial links in both time slots. If an element $e_i \in S_j$, we add two spacial links $\overrightarrow{e_i^0 S_j^1}$ and $\overrightarrow{S_j^1 e_i^2}$. All links have the same reliability of 1. The reliability requirement $\gamma$ of the min-throwbox problem is also set as 1. By this overall construction, it is easy to verify that a solution of the constructed min-throwbox problem is a solution of the original set cover problem. Since the construction can be done in polynomial time and the set cover problem is NP-hard, thus the min-throwbox problem is also NP-hard. □

The NP-hardness of the k-throwbox problem is not very straightforward. However, if we define the reliability of the topology as the summation of path reliability in the space-time graph, then a similar construction in Fig. 3 can be used to reduce the k set cover problem to our k-throwbox problem.

**Theorem 2.** *The* k-throwbox problem *is NP-hard, if the topology reliability is defined as summation of path reliability.*

## 4 Throwbox Selection Algorithms

Since both throwbox optimization problems are computationally hard, in this section, we propose a set of different heuristics to carefully select active throwbox placements fulfilling the reliability requirement or maximizing it. Once again, we assume that the space-time graph $\mathcal{G} = (\mathcal{E}, \mathcal{V})$, including $n$ mobile users and $m$ potential throwboxes over time period of $T$, is given as the input. Let $N$ and $M$ denote the total number of nodes and links in graph $\mathcal{G}$ (i.e., $|\mathcal{V}|$ and $|\mathcal{E}|$), respectively. Notice that $N = (n+m)(T+1)$ and $M = O((n+m)^2 T)$.

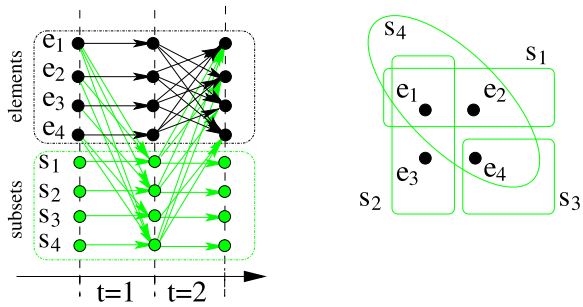



Fig. 3. Illustrations for NP-hard proof of the min-throwbox problem.

### 4.1 General Greedy Approaches

Finding the optimal solutions for throwbox optimization problem by exploring all possible combinations of throwbox selection is very challenging and time consuming, thus, our greedy approaches simply make a single throwbox choice in each round by adding or removing one active throwbox from the network. The procedure will guarantee to terminate after at most $m$ rounds, which is much more efficient than exponential brute force algorithm. The same approaches work for both $k$-throwbox problem and min-throwbox problem, and the only difference is the termination condition. One is when $k$ active throwboxes are selected, while the other is when the reliability requirement is achieved or void. The detailed general methods are given in Algorithm 1 and Algorithm 2.

**Algorithm 1.** Greedy-Adding Throwboxes (GrdAddTBs)

**Input:** the original space-time graph $\mathcal{G}$ (including potential throwbox set $V_{throwbox}$), a constant $k$ (or a threshold $\gamma$).
**Output:** the selected throwbox set $V_{selected-throwbox}$ and the corresponding new space-time graph $\mathcal{H}$.

$\mathcal{H} \leftarrow \mathcal{G} - \{V_{throwbox}\}$ and $V_{selected-throwbox} = \emptyset$
**while** $|V_{selected-throwbox}| < k$ (or $r(\mathcal{H}) < \gamma$) **do**
  Greedily select a throwbox $v_i$ from all unselected throwboxes $V_{throwbox} - V_{selected-throwbox}$, i.e., $v_i = GreedySelect(V_{throwbox} - V_{selected-throwbox}, \mathcal{H})$
  $\mathcal{H} \leftarrow \mathcal{H} + \{v_i\}$
  $V_{selected-throwbox} \leftarrow V_{selected-throwbox} + \{v_i\}$
**return** $V_{selected-throwbox}$ and $\mathcal{H}$

**Algorithm 2.** Greedy-Deleting Throwboxes (GrdDelTBs)

**Input:** the original space-time graph $\mathcal{G}$ (including potential throwbox set $V_{throwbox}$), a constant $k$ (or a threshold $\gamma$).
**Output:** the selected throwbox set $V_{selected-throwbox}$ and the corresponding new space-time graph $\mathcal{H}$.

$\mathcal{H} \leftarrow \mathcal{G}$ and $V_{selected-throwbox} = V_{throwbox}$
**while** $|V_{selected-throwbox}| > k$ (or $r(\mathcal{H}) > \gamma$) **do**
  Greedily select a throwbox $v_i$ from $V_{selected-throwbox}$, i.e., $v_i = GreedySelect(V_{selected-throwbox}, \mathcal{H})$
  $\mathcal{H} \leftarrow \mathcal{H} - \{v_i\}$
  $V_{selected-throwbox} \leftarrow V_{selected-throwbox} - \{v_i\}$
**return** $V_{selected-throwbox}$ and $\mathcal{H}$ (or $V_{selected-throwbox} + \{v_i\}$ and $\mathcal{H} + \{v_i\}$ )

The first algorithm (GrdAddTBs) starts with a space-time graph $\mathcal{H}$ only including mobile users $(v_1, v_2, \ldots, v_n)$. Then it greedily adds in active throwboxes until either $k$ throwboxes are activated (for k-throwbox problem) or the reliability of $\mathcal{H}$ reaches the required threshold (for min-throwbox problem). The second algorithm (GrdDelTBs) starts with the original space-time graph $\mathcal{G}$ with all throwboxes activated, and gradually deletes active throwboxes until only $k$ active throwboxes are left or the reliability constraint breaks. In both algorithms, during the process, our method greedily selects one single active throwbox in each round based on certain criteria (as shown in Line 3 in both algorithms, and we will introduce different criteria in the next section). Hereafter, we generalize such greedy selection of a single throwbox $v_i$ from a set of throwboxes $V_x$ based on current space-time graph $\mathcal{H}$ as a function $GreedySelect(V_x, \mathcal{H})$ with $v_i$ as its output. Let us denote the time complexity of $GreedySelect()$ as X.

Both GrdAddTBs and GrdDelTBs can obviously satisfy the number of throwboxes requirement (or reliability requirement) of $\mathcal{H}$. For the $k$-throwbox problem, the time complexities of GrdAddTBs and GrdDelTBs are $O(kX)$ and $O((m-k)X)$, respectively. For the min-throwbox problem, the time complexities of GrdAddTBs and GrdDelTBs are $O(k \max\{X, Y\})$ and $O((m-k) \max\{X, Y\})$, respectively. Here, $Y$ denotes the time complexity of checking the reliability constraint, which is $O(n(M + N \log N))$ if Dijkstra's algorithm is used.

### 4.2 How to Pick the Best Throwbox

Now we are ready to describe two different criteria for the $GreedySelect()$: based on *node degrees* or *reliability changes*, to select the best throwbox in each round to be added in or removed from the network.

#### 4.2.1 Based on Node Degrees

Each throwbox may bring new contact and forwarding opportunities to the mobile users in the network. One way to measure such improvement over connectivity of a throwbox $v_i$ is its total node degree added to the network, i.e., $d(v_i) = \sum_{t=1}^{T}(d(v_i^t))$, where $d(v_i^t)$ denotes the number of links from/to $v_i^t$ to/from other mobile users at time slot $t+1$ or $t$. In each greedy iteration, we simply add the throwbox with largest $d(v_i)$ (or remove the throwbox with smallest $d(v_i)$). The intuition behind it is trying to use the throwboxes with better connectivities (larger node degree over time) to improve the reliability among mobile users. The time complexity of $GreedySelect$ based on node degrees is $O(mDT)$ where $D$ is the maximum node degree of a throwbox at time $t$ or $t+1$. Clearly $D$ is bounded by $2n$. Thus, the time complexity is $O(mnT)$.

#### 4.2.2 Based on Reliability Changes

More directly, we can use the reliability changes due to adding or removing throwbox, i.e., $r(v_i) = r(\mathcal{H} + \{v_i\}) - r(\mathcal{H})$ for GrdAddTBs or $r(v_i) = r(\mathcal{H}) - r(\mathcal{H} - \{v_i\})$ for GrdDelTBs. In each greedy iteration, we simply add the throwbox with largest reliability improvement $r(v_i)$ (or remove the throwbox with the smallest deduction $r(v_i)$). Obviously, this metric is more direct to our optimization goal or constraint than node degrees. The time complexity of this method is around $O(mn(M + N \log N))$ if $m$ rounds of $n$ times of Dijkstra's algorithm are used. In term of complexity, in the worst case, this could be much larger than those based on node degrees.

Hereafter, we use postfixes -D and -R to represent which greedy criterion is used by the general approach. For example, GrdAddTBs-D or GrdAddTBs-R denotes the greedy algorithm which uses node degree metric or reliability change metric to select a throwbox to be added in each round.

### 4.3 Performance Guarantee of GrdAddTBs-R

It is always nice to have performance guarantee for some simple greedy heuristics. However, it is not always an easy case to prove any approximation ratio. Fortunately, we can prove the following lemma (Lemma 1) that the reliability function $r(\mathcal{H})$ is non-negative, monotone, and submodular. Consider an arbitrary function $f(A)$ that maps subsets of a finite ground set $U$ to non-negative real numbers. We say that $f$ is submodular if it satisfies a natural diminishing returns property: the marginal gain from adding an element to a set $A$ is at least as high as the marginal gain from adding the same element to a superset of $A$. Formally, a submodular function satisfies

$$f(A \cup \{v\}) - f(A) \geq f(B \cup \{v\}) - f(B)$$

for all elements $v$ and all pairs of sets $A \subseteq B$. Next, we prove that the reliability function $r(\mathcal{H})$ is submodular.

**Lemma 1.** *The reliability function $r(\mathcal{H})$ is non-negative, monotone, and submodular.*

**Proof.** Non-negative property is obvious. Let $\mathcal{H}'$ and $\mathcal{H}''$ be two subgraphs of $\mathcal{G}$ which include selected sets $A$ and $B$ of throwboxes, respectively. Assume that $A \subseteq B$, i.e., $\mathcal{H}' \subseteq \mathcal{H}''$ and $\mathcal{H}''$ uses additional throwboxes other than $\mathcal{H}'$. Since all subgraphs use $n$ mobile users, hereafter, we only use the throwboxes set in the reliability function. Thus, $r(A) = r(\mathcal{H}')$ and $r(B) = r(\mathcal{H}'')$, then the equation $r(A \cup \{v\}) - r(A)$ denotes the reliability increase due to add a new throwbox $v$ to $A$. Obviously, it is non-negative value, since adding one more throwbox can increase the reliability. This implies that reliability function $r$ is a monotone function. Now consider $r(B \cup \{v\}) - r(B)$, which is the reliability increase due to adding the throwbox $v$ to $B$. Any improvement of reliability is associated with an original path in $\mathcal{H}''$. If the improved path does not use any throwbox which is in $B$ but not in $A$, then the same level of improvement should also occur for $A$ (i.e., $r(A \cup \{v\}) - r(A) = r(B \cup \{v\}) - r(B)$). If the improved path does use some throwboxes which are not in $A$, then the improvement over $B$ is much less than $A$ (i.e., $r(A \cup \{v\}) - r(A) > r(B \cup \{v\}) - r(B)$) since $r(B) \geq r(A)$. Therefore, overall $r(A \cup \{v\}) - r(A) \geq r(B \cup \{v\}) - r(B)$ and $r$ is submodular. □

Submodular functions have a number of nice properties. One of them is a result from [22], [23], summarized as the following theorem.

**Theorem 3.** *For a non-negative, monotone submodular function $f$, let $S$ be a set of size $k$ obtained by selecting elements one at a time, each time choosing an element that provides the largest marginal increase in the function value. Let $S^*$ be a set that maximizes the value of $f$ over all $k$-element sets. Then $f(S) \geq (1 - 1/e) \cdot f(S^*)$; in other words, $S$ provides a $(1 - 1/e)$-approximation.*

Notice that our $k$-throwbox problem is exactly the type of problem described in Theorem 3. Theorem 3 and Lemma 1 together implies:

**Theorem 4.** *GrdAddTBs-R (Algorithm 1 with greedy metric based on reliability changes) guarantees a $(1 - 1/e)$ approximation for the $k$-throwbox problem.*

On the other hand, to prove the approximation ratio for the min-throwbox problem is relatively harder. In the min-throwbox problem, we aim to find a minimum set of throwboxes which can achieve certain reliability threshold. Such a problem is more like the minimum submodular cover problem [24], whose goal is to find a covering set of minimum cost. Our reliability threshold can be treated as the covering requirement. However, for the minimum

submodular cover problem, the greedy algorithm cannot give a constant approximation ratio. In [24], Wolsey has shown that greedy algorithm can give a $H(\beta)$-approximation for the minimum submodular cover problem where $\beta$ is the maximum value of the submodular function $f$ over all singletons and $H(\beta)$ is the $\beta$th harmonic number. However, his proof requires that $f$ is integer valued. Notice our reliability function is not necessarily an integer valued. Fortunately, recently Wan et al. [25] have proved a more general result and extended the approximation proof to fractional submodular function. From Theorem 3.1 of [25], we can directly have the following theorem.

**Theorem 5.** *GrdAddTBs-R (Algorithm 1 with greedy metric based on reliability changes) guarantees a* $(1 + \ln(r(\mathcal{G})/r(opt)))$ *approximation for the min-throwbox problem, where* opt *is the optimal solution of throwbox selection.*

Notice that in our case $\rho = 1$ (a parameter in Theorem 3.1 of [25] reflects the curvature of the submodular cost), since the cost (the total number of selected throwbox) is linear. In addition, the minimum submodular cover problem studied in [25] asks for full coverage (i.e., $r(\mathcal{H}) \geq r(\mathcal{G})$) and requires $r(\mathcal{H}_\emptyset) = 0$ (here $\mathcal{H}_\emptyset = \mathcal{G} - \{V_{throwbox}\}$ means a $\mathcal{H}$ with only $n$ mobile users and no throwbox selected). However, defining a new submodular function $r'(\mathcal{S}) = \min\{r(\mathcal{S}), \gamma\} - r(\mathcal{H}_\emptyset)$ can solve the problems.

### 4.4 Another Greedy Solution

So far all greedy algorithms only add/remove a single throwbox in each round, we now introduce a method which adds multiple throwboxes in each round. This algorithm is only for the min-throwbox problem. Note that to achieve the reliability constraint $\gamma$, basically we need a reliable path between each pair $(v_i^0, v_j^T)$ for $i, j = 1, \ldots, n$. Let $Z$ represent the set of pairs of $(v_i^0, v_j^T)$ for $1 \leq i, j \leq n$ whose reliabilities in current space-time graph $\mathcal{H}$ are still smaller than $\gamma$. In each round, the new greedy algorithm tries to add some throwboxes along one single reliable path to improve the reliability of paths between at least one pair of nodes in $Z$. Thus, after $n^2$ rounds, all pairs of nodes in the original $Z$ are guaranteed to be connected by reliable paths in $\mathcal{H}$. The greedy criteria of which path to pick among all possible reliable paths is simply the one with the least number of unselected throwboxes. By doing so, the algorithm hopefully only uses the minimum number of throwboxes at the end. Algorithm 3 gives the detailed algorithm. We use GrdAddTBs-P to denote this greedy algorithm. To obtain the reliable path with the least number of unselected throwboxes alone is not an easy task. It is basically a variation of the *restricted shortest path* problem [26], a NP-hard problem. Thus, we use an existing heuristic, Backward-Forward method (BFM) [27], which is one of the most efficient methods among all existing methods (its execution time is about three times that of Dijkstra's algorithm). The overall computational complexity of GrdAddTBs-P is roughly $O(n^3(M + N \log N))$, since in each round $O(n)$ times of Dijkstra's algorithm are running on the space-time graph and there are $n^2$ rounds.

## 5 Simulations

To evaluate our proposed algorithms for throwbox optimization problems, we have conducted extensive simulations over randomly generated time-evolving networks and real DTNs extracted from realistic contact traces [15]. We implement and test the following algorithms:

- *GrdAdd(Del)TBs-D:* greedy algorithms adding/deleting throwboxes based on node degrees.
- *GrdAdd(Del)TBs-R:* greedy algorithms adding/deleting throwboxes based on reliability changes.
- *GrdAdd(Del)TBs-Ra:* greedy algorithms randomly adding/deleting throwboxes.
- *GrdAddTBs-P:* greedy algorithm adding throwboxes based on reliable path with least throwboxes.
- *OPT:* the optimal solution for throwbox optimization problem obtained by brute force method.

**Algorithm 3.** Greedy-Adding Throwboxes based on Reliable Path with Least Throwboxes (GrdAddTBs-P)

**Input:** the original space-time graph $\mathcal{G}$ (including potential throwbox set $V_{throwbox}$) and a reliability threshold $\gamma$.
**Output:** the selected throwbox set $V_{selected-throwbox}$ and the corresponding new space-time graph $\mathcal{H}$.

$\mathcal{H} \leftarrow \mathcal{G} - \{V_{throwbox}\}$ and $V_{selected-throwbox} = \emptyset$
**while** $r(\mathcal{H}) < \gamma$ **do**
  $Z = \{(v_i^0, v_j^T) | r^{\mathcal{H}}(v_i^0, v_j^T) < \gamma \text{ and } i, j \in [1, n]\}$
  **for all** pairs $(v_i^0, v_j^T) \in Z$ **do**
    Find a reliable path between $v_i^0$ and $v_j^T$ in $\mathcal{G}$ (i.e., its reliability $\geq \gamma$), which contains the least number of unselected throwboxes. Denote it as $P_{Least-TBs}(v_i^0, v_j^T)$
  Pick the path using the least number of unselected throwboxes among all $P_{Least-TBs}(v_i^0, v_j^T)$ for all $(v_i^0, v_j^T) \in Z$. Assume it is $P_{Least-TBs}(v_k^0, v_l^T)$.
  **for all** throwbox $v_p \in P_{Least-TBs}(v_k^0, v_l^T)$ and $v_p \notin V_{selected-throwbox}$ **do**
    $\mathcal{H} \leftarrow \mathcal{H} + \{v_p\}$
    $V_{selected-throwbox} \leftarrow V_{selected-throwbox} + \{v_p\}$
**return** $V_{selected-throwbox}$ and $\mathcal{H}$

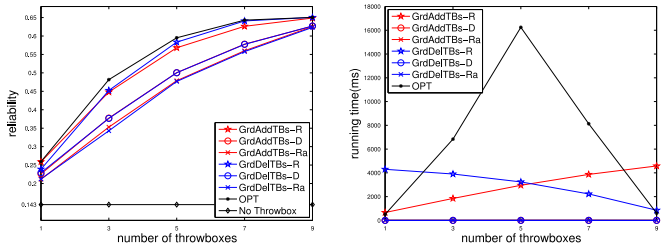
Here for reference purposes we include two randomized algorithms (GrdAdd(Del)TBs-Ra) where a randomly selected throwbox is added or deleted in each round. The performance metrics are the reliability of resulting network (i.e., $r(\mathcal{H})$) for $k$-throwbox problem and the number of selected throwboxes for min-throwbox problem. In addition, we also measure the actual running time of each method.

### 5.1 Simulations on Random Time-Evolving Networks

We first test our algorithms on randomly generated networks. We generate a sequence of static random graphs with $n + m$ nodes ($n$ mobile users and $m$ potential throwboxes) over $T = 10$ time slots. For each static snapshot, the link between two mobile users or one mobile user and one throwbox is randomly inserted based on a probability $p$. Clearly, the larger value of $p$ is, the denser the network is. For each link $e$ we then randomly generate its reliability $r(e)$ in a range $[r_{min}, r_{max}]$. We test different settings of these parameters in our simulations, and the discoveries and conclusions are consistent. Due to space limit, we only report the results for the following setting. We set $p = 0.11$, $r_{min} = 0.3$, and $r_{max} = 0.6$ for links between a pair of mobile users; and set $p = 0.22, r_{min} = 0.6$, and $r_{max} = 1.0$ for links between a mobile user and a throwbox. Obviously, throwboxes are usually more reliable than normal mobile devices. Finally, we generate the weighted space-time graph based on the sequence of static graphs. For each setting, we generate 100 random time-evolving networks and report average performances of our proposed algorithms.

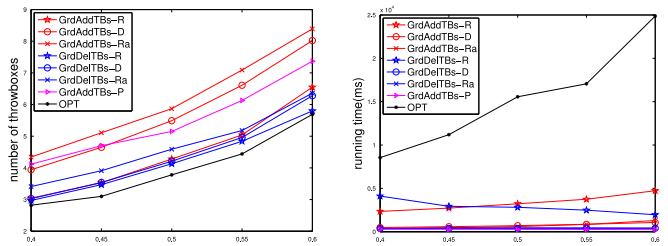
#### 5.1.1 $k$-Throwbox Problem

We first test our proposed algorithms for $k$-throwbox problem, where $k$ throwboxes need to be selected and the goal is to maximize the reliability of the network (i.e., $r(\mathcal{H})$). Here, we first run our algorithms on random networks with $n$ mobile users and $m$ throwboxes (i.e., $n = 20$ and $m = 10$), and let $k$ range from 1 to 9.



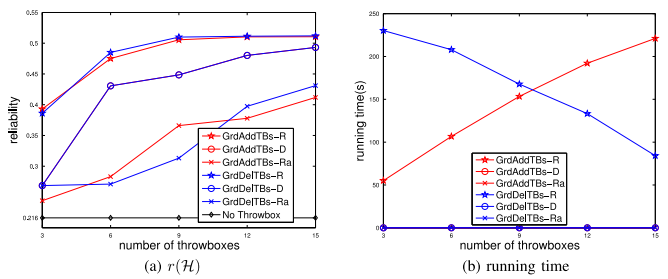

(a) $r(\mathcal{H})$ (b) running time

Fig. 4. Results on random nets ($n = 20$, $m = 10$) for $k$-throwbox problem.




(a) number of throwboxes (b) running time

Fig. 5. Results on random nets ($n/m = 20/10$) for min-throwbox problem.



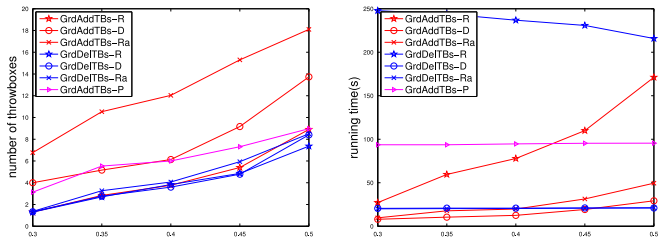

(a) $r(\mathcal{H})$ (b) running time

Fig. 6. Simulation results for $k$-throwbox problem on networks from Infocom 2006 trace data [15] ($n = 40$ and $m = 20$ static throwboxes).

For these small networks, we are able to find the optimal solution $OPT$ with brute force algorithm. Fig. 4a shows the reliabilities reached by each algorithm with different number of throwboxes. It is clear that with more throwboxes a higher reliability can be achieved. The straight blue line at the bottom shows the reliability without any throwboxes. Fig. 4b also plots the running time of each algorithm. Via these two figures, we can find: (1) Brute force algorithm can find the optimal solution with maximum reliability but the running time is the largest among all methods; (2) Both random algorithms (GrdAddTBs-Ra and GrdDelTBs-Ra) preform poorly in term of achieved reliability; (3) GrdAddTBs-R and GrdDelTBs-R can achieve the best reliability among all proposed methods and almost match the $OPT$, which confirms our theoretical analysis on approximation ratio; (4) GrdAddTBs-D and GrdDelTBs-D achieve the same reliability since they are based on the same degree order. Although they cannot achieve the same level of reliability with those based on reliability changes, their running times are much less than those of GrdAddTBs-R and GrdDelTBs-R. Thus there is a tradeoff between network reliability and time complexity. We also test the performance of proposed algorithms in larger random networks to discover the scalability of our algorithms. We can draw the similar conclusions. Overall, GrdAddTBs-R and GrdDelTBs-R can achieve the highest reliability.

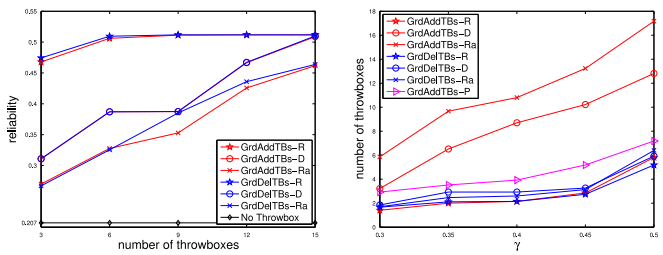
#### 5.1.2 Min-Throwbox Problem

For the min-throwbox problem, we test all algorithms (including GrdAddTBs-P) over the same sets of random networks, with the reliability constraint $\gamma$ ranging from $0.40$ to $0.60$. Fig. 5




(a) number of throwboxes (b) running time

Fig. 7. Simulation results for min-throwbox problem on networks from Infocom 2006 trace data [15] ($n = 40$ and $m = 20$ static throwboxes).




(a) $k$-throwbox (b) min-throwbox

Fig. 8. Simulation results $r(\mathcal{H})$ on networks from Infocom 2006 trace data [15] ($n = 40$ and $m = 20$ mobile throwboxes).

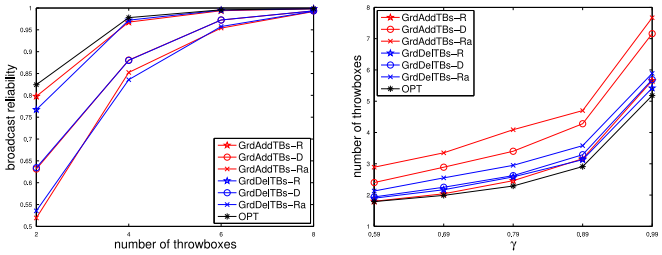
shows the detailed results. It is clear that higher reliability requirements lead to more throwboxes involved. This also confirms that more throwboxes can introduce more contact opportunities in the network. Again $OPT$ needs the smallest number of throwboxes, however its running time is the largest and increases exponentially. Both GrdDelTBs-R and GrdAddTBs-R perform very well (requiring small number of throwboxes) as expected. What is interesting is that GrdDelTBs-D also uses quite small number of throwboxes. Generally, greedy deleting throwboxes scheme uses less throwboxes than greedy adding throwboxes scheme. GrdAddTBs-P is slightly better than GrdAddTBs-D.

### 5.2 Simulations on Real DTN Tracing Data

Taking advantages of public wireless tracing data, we also test our algorithms over a realistic contact traces: the Infocom 2006 trace data [15]. This data set includes Bluetooth sightings by groups of users (i.e., 78 participants) carrying iMotes for four days during Infocom 2006 conference in Barcelona, Spain. In addition, 20 stationary iMotes were deployed throughout the hotel, with more powerful batteries and extended radio ranges. For this set of simulation, we randomly choose 40 mobile users from the 78 mobile iMotes, and treat 20 stationary iMotes as 20 potential static throwboxes. We generate 30 random time-evolving networks, and report average performances of our proposed algorithms. The reliabilities of links are randomly generated as we did for random networks. Figs. 6 and 7 show the results for $k$-throwbox problem and min-throwbox problem, respectively. All the conclusions are consistent with those from random network experiments and confirm our theoretical analysis. Methods based on reliability changes (GrdDelTBs-R and GrdAddTBs-R) perform very well in solving both optimization problems. In addition, GrdDelTBs-D also works well for min-throwbox problem and uses less running time.

### 5.3 Mobile Throwboxes

So far we only consider static throwboxes. Static throwboxes may help with increasing contact opportunities at certain time, however, they might be idle in other time slots. If throwboxes can move, they can change to better places during their idle time slots. Thus, it is possible to introduce mobile throwboxes into DTN to further




(a) $k$-throwbox (b) min-throwbox

Fig. 9. Results on random nets ($n/m = 20/10$) with broadcast reliability.

increase the contact opportunities. Our model and proposed methods can be directly applied to mobile throwboxes, since the space-time graph only describes the contact relationship among mobile users and throwboxes. As long as the future contact can be predicted, no change is needed to handle mobile throwboxes. Fig. 8 shows a set of results from experiments over Infocom traces [15], where 20 mobile devices are chosen as throwboxes instead. Results confirm that mobile throwboxes can also significantly improve the reliability and our proposed methods work well in such scenario too.

### 5.4 Broadcast Reliability

Previously, we assume a single-copy DTN routing protocol is used. If multiple copies are allowed during the propagation, the reliability will increase [28], [29], [30]. The simplest multi-copies routing is flooding routing or epidemic routing [28], where a node with a message will relay it to every node it encounters. For flooding-based routing, we can define a new type of reliability, *broadcast reliability*, as follows. For a given source-destination pair $u, v$ in $\mathcal{H}$, the $r^{\mathcal{H}}(u, v)$ is the probability that a packet sent from node $u$ over the topology $\mathcal{H}$ reaches node $v$ under flooding-based DTN routing. To efficiently calculate the pair-wise broadcast reliability is not an easy job. Actually, it is known that the computation of such reliability over general graphs is a NP-hard problem [31]. Fortunately, the space-time graph in our model is a very special directed acyclic graph where all paths from the sources to the destinations are $T$ hops and there are no loops. This property allows us to compute the reliability $r^{\mathcal{H}}(u, v)$ efficiently by using Dynamic Programming (DP) [14], with the same time complexity as that of Dijkstra's algorithm. Thus, all of our proposed algorithms except for GrdAddTBs-P can then work with broadcast reliability. The only difference is using the DP algorithm to check the reliability of a topology. Fig. 9 shows results over random time-evolving networks based on broadcast reliability. No significant difference is found for these simulations compared with those with unicast reliability. The only fact is that for the same network, broadcast reliability is much higher than its unicast reliability.

## 6 Conclusion

Recent studies have shown the enhancement of DTN performances with throwboxes. This paper investigates a key problem, throwbox selection, in a time-evolving throwbox-assisted DTN modeled by a weighted space-time graph. Two throwbox optimization problems are formally introduced: min-throwbox problem and k-throwbox problem. We formally analyze the hardness of both problems and propose a set of greedy algorithms which can efficiently provide quality solutions. We show the efficiency of the proposed methods through extensive simulations over both random time-evolving DTNs and real life DTN traces. One limitation of the proposed problems and solutions is the assumption on predictability of future contacts. For some of large-scale real DTNs, it is difficult to obtain accurate prediction on future contacts among devices. One possible solution is leveraging the social pattern among devices which can be discovered via mining of historical records [32].

## Acknowledgments

The work is partially supported by the National Natural Science Foundation of China under Grant Nos. 61370192, 61432015 and 61428203, and the US National Science Foundation under Grant Nos. CNS-1319915 and CNS-1343355. Yu Wang is the corresponding author.

## References

[1] C. Liu and J. Wu, "Scalable routing in delay tolerant networks," in *Proc. 8th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2007, pp. 51–60.
[2] P. Hui, J. Crowcroft, and E. Yonek, "Bubble rap: Social-based forwarding in delay tolerant networks," in *Proc. 9th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2008, pp. 241–250.
[3] V. Erranmilli, M. Crovella, A. Chaintreau, and C. Diot, "Delegation forwarding," in *Proc. 9th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2008, pp. 251–260.
[4] Y. Zhu, B. Xu, X. Shi, and Y. Wang, "A survey of social-based routing in delay tolerant networks: positive and negative social effects," *IEEE Commun. Survey Tuts.*, vol. 15, no. 1, pp. 387–401, Jan.-Mar. 2013.
[5] W. Zhao, Y. Chen, M. Ammar, M. Corner, B. Levine, and E. Zegura, "Capacity enhancement using throwboxes in DTNs," in *Proc. IEEE Int. Conf. Mobile Ad hoc Sens. Syst. Conf.*, 2006, pp. 31–40.
[6] N. Banerjee, M. D. Corner, and B. N. Levine, "Design and field experimentation of an energy-efficient architecture for DTN throwboxes," *IEEE/ACM Trans. Netw.*, vol. 18, no. 2, pp. 554–567, Apr. 2010.
[7] M. Ibrahim, A. Al Hanbali, and P. Nain, "Delay and resource analysis in MANETs in presence of throwboxes," *Perform. Eval.*, vol. 64, no. 9, pp. 933–947, 2007.
[8] B. Gu and X. Hong, "Latency analysis for thrown box based message dissemination," in *Proc. IEEE Global Telecommun. Conf.*, 2010, pp. 1–5.
[9] M. Ibrahim, P. Nain, and I. Carreras, "Analysis of relay protocols for throw-box-equipped DTNs," in *Proc. 7th Int. Conf. Modeling Optim. Mobile, Ad Hoc, Wireless Netw.*, 2009, pp. 222–230.
[10] B. Gu and X. Hong, "Capacity-aware routing using throw-boxes," in *Proc. IEEE Global Telecommun. Conf.*, 2011, pp. 1–5.
[11] E. L. Lloyd and G. Xue, "Relay node placement in wireless sensor networks," *IEEE Trans. Comput.*, vol. 56, no. 1, pp. 134–138, Jan. 2007.
[12] X. Cheng, D.-Z. Du, L. Wang, and B. Xu, "Relay sensor placement in wireless sensor networks," *Wireless Netw.*, vol. 14, no. 3, pp. 347–355, 2008.
[13] M. Huang, S. Chen, Y. Zhu, and Y. Wang, "Topology control for time-evolving and predictable delay-tolerant networks," *IEEE Trans. Comput.*, vol. 62, no. 11, pp. 2308–2321, Nov. 2013.
[14] F. Li, S. Chen, M. Huang, Z. Yin, C. Zhang, and Y. Wang, "Reliable topology design in time-evolving delay-tolerant networks with unreliable links," *IEEE Trans. Mobile Comput.*, vol. 14, no. 6, pp. 1301–1314, Jun. 2015.
[15] J. Scott, R. Gass, J. Crowcroft, P. Hui, C. Diot, and A. Chaintreau. CRAWDAD trace set cambridge/haggle/imote (v. 2009-05-29). [Online]. Available: http://crawdad.cs.dartmouth.edu/cambridge/haggle/imote, 2009.
[16] F. El-Moukaddem, E. Torng, and G. Xing, "Mobile relay configuration in data-intensive wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 12, no. 2, pp. 261–273, Feb. 2013.
[17] W. Wang, V. Srinivasan, and K.-C. Chua, "Using mobile relays to prolong the lifetime of wireless sensor networks," in *Proc. 11th Annu. Int. Conf. Mobile Comput. Netw.*, 2005, pp. 270–283.
[18] S. Merugu, M. Ammar, and E. Zegura, "Routing in space and time in networks with predictable mobility," Georgia Institute of Technology, Atlanta, Georgia, USA, Tech. Rep. GIT-CC-04-07, 2004.
[19] C. Liu and J. Wu, "Routing in a cyclic mobispace," in *Proc. 9th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2008, pp. 351–360.
[20] N. Baccour, A. Koubaa, L. Mottola, M. A. Z. Zamalloa, H. Youssef, C. A. Boano, and M. Alves, "Radio link quality estimation in wireless sensor networks: A survey," *ACM Trans. Sens. Netw.*, vol. 8, no. 4, pp. 34:1–34:33, Sep. 2012.
[21] W. Su, S.-J. Lee, and M. Gerla, "Mobility prediction in wireless networks," in *Proc. 21st Century Mil. Commun. Conf.*, 2000, pp. 491–495.
[22] G. Cornuejols, M. L. Fisher, and G. L. Nemhauser, "Location of bank accounts to optimize float: An analytic study of exact and approximate algorithms," *Manage. Sci.*, vol. 23, no. 8, pp. 789–810, 1977.
[23] G. Nemhauser, L. Wolsey, and M. Fisher, "An analysis of approximations for maximizing submodular set functions-I," *Math. Program.*, vol. 14, no. 1, pp. 265–294, 1978.
[24] L. A. Wolsey, "An analysis of the greedy algorithm for the submodular set covering problem," *Combinatorica*, vol. 2, no. 4, pp. 385–393, 1982.
[25] P.-J. Wan, D.-Z. Du, P. Pardalos, and W. Wu, "Greedy approximations for minimum submodular cover with submodular cost," *Comput. Optim. Appl.*, vol. 45, no. 2, pp. 463–474, 2010.
[26] F. Kuipers, P. Van Mieghem, T. Korkmaz, and M. Krunz, "An overview of constraint-based path selection algorithms for QoS routing," *IEEE Commun. Mag.*, vol. 40, no. 12, pp. 50–55, Dec. 2002.
[27] D. S. Reeves and H. F. Salama, "A distributed algorithm for delay-constrained unicast routing," *IEEE/ACM Trans. Netw.*, vol. 8, no. 2, pp. 239–250, Apr. 2000.

[28] A. Vahdat and D. Becker, "Epidemic routing for partially connected ad hoc networks," Duke Univ., Durham, NC, USA, Tech. Rep. CS-200006, Apr. 2000.
[29] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Efficient routing in intermittently connected mobile networks: The multiple-copy case," *IEEE Trans. Netw.*, vol. 16, no. 1, pp. 77–90, Feb. 2008.
[30] X. Zhang, G. Neglia, J. Kurose, and D. Towsley, "Performance modeling of epidemic routing," *Comput. Netw*, vol. 51, no. 10, pp. 2867–2891, 2007.
[31] A. Agrawal and R. E. Barlow, "A survey of network reliability and domination theory," *Oper. Res.*, vol. 32, no. 3, pp. 478–492, 1984.
[32] Y. Zhu, C. Zhang, X. Mao, and Y. Wang, "Social based throwbox placement schemes for large-scale mobile social delay tolerant networks," *Comput. Commun.*, vol. 65, pp. 10–26, 2015.