k-Throwbox Placement Problem in Throwbox-Assisted Delay Tolerant Networks

Fan Li* Zhiyuan Yin* Shaojie Tang† Chao Zhang‡ Yu Cheng§ Yu Wang‡
* School of Computer Science, Beijing Institute of Technology, Beijing, 100081, China.
† Naveen Jindal School of Management, University of Taxes at Dallas, Richardson, TX 75080, USA.
‡ Department of Computer Science, University of North Carolina at Charlotte, Charlotte, NC 28223, USA.
§ Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, IL 60616, USA.

***Abstract*—Recent advances in Delay Tolerant Networks (DTNs) have overcome limitations in connectivity by relying on intermittent contacts between mobile nodes to deliver packets. However, lack of rich contact opportunities 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, to create a greater number of contact opportunities. In this paper, we study a key optimization problem in a time-evolving throwbox-assisted DTN: k-throwbox placement problem, to answer "where should I put my k throwboxes to optimize the performance?". We model a time-evolving DTN as a weighted space-time graph which includes both spacial and temporal information. We prove that k-throwbox placement problem is NP-hard and propose a set of greedy algorithms which can efficiently provide quality solutions. One of the proposed algorithms can guarantee an $(1-1/e)$ approximation for the k-throwbox placement problem. Simulation results based on random time-evolving DTNs and real life DTN traces demonstrate the efficiency of the proposed methods.**

I. Introduction

Delay Tolerant Networks (DTNs) have recently drawn much attention from networking researchers due to the wide applications of these networks in challenging environments, such as space communications, military operations, vehicular ad hoc networks, mobile sensor networks, and pocket switched networks. The intermittent connectivities in DTNs result in the lack of instantaneous end-to-end paths, large transmission delay and unstable network topology. These characteristics pose new challenges in the design and deployment of DTNs. Recent advances in DTN routing [1]–[3] 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 [4]–[9]. 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 [5]–[9] have demonstrated that introducing small number of throwboxes can indeed improve the network performances and overall throughputs.

One of the key design problems in throwbox-assisted DTNs is throwbox placement. Given a set of potential locations of throwboxes and a budget to only deploy k throwboxes, we need to find where to put these throwboxes to maximize their benefits to the performance. General relay placement in static wireless networks [10], [11] 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 except for [4], which is a joint throwbox deployment and routing optimization problem. However, their focus is only on the long term average capacity. Instead, in this paper we study how to deploy throwboxes in a time-evolving and predictable DTN so that the network reliability is maximized.

In this paper, we model a time-evolving and predictable DTN as a weighted space-time graph (defined in Section II) 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; and (2) there is a finite set of potential locations for throwboxes. We then formally define the k-throwbox placement problem in Section III: aiming to find places to put k throwboxes so that the network reliability is maximized over time. We prove that this problem is NP-hard. Thus, we then propose a set of greedy algorithms in Section IV which can efficiently provide quality solutions for this optimization problem. One of the proposed algorithms can particularly guarantee an $(1-1/e)$ approximation ratio. We also conduct extensive simulations over random time-evolving DTNs and real life DTN traces

The work of F. Li is partially supported by the National Natural Science Foundation of China under Grant No. 61370192, 61432015 and 60903151, and the Beijing Natural Science Foundation under Grant No. 4122070. The work of Y. Cheng is supported in part by the US National Science Foundation under grant CNS-1320736. The work of Y. Wang is supported in part by the US National Science Foundation under Grant No. CNS-1050398, CNS-1319915, and CNS-1343355, and by the National Natural Science Foundation of China under Grant No. 61428203.

978-1-4799-3512-3/14/$31.00 ©2014 IEEE

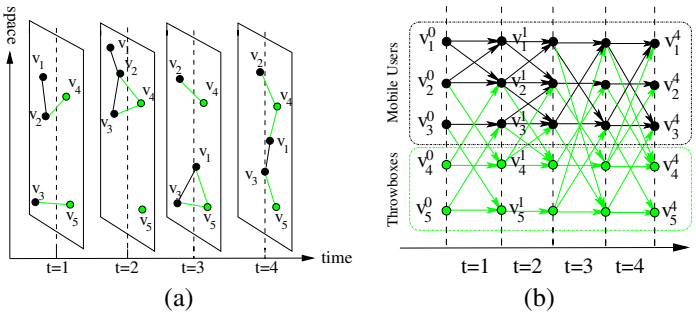


Fig. 1. (a) Example of a time-evolving throwbox-assisted DTN (two green nodes are throwboxes). (b) The corresponding space-time graph.

to demonstrate the efficiency of the proposed methods in Section V. Finally, some discussions on possible variations of the problem are provided in Section VI and a brief conclusion is given in Section VII.

II. Models and Assumptions

How to model time-evolving networks has been studied in both mobile ad hoc networks [12], [13] and DTNs [14], [15]. In this paper, we adopt the *space-time graph* [12], [14] to model the time-evolving DTNs, since it can capture the evolving characteristics in both spacial and temporal spaces.

Assume that $V_{user} = \{v_1, \cdots, v_n\}$ and $V_{throwbox} = \{v_{n+1}, \cdots, v_{n+m}\}$ be the set of all individual users (wireless devices) and the set of all potential throwboxes in the network over a period of time T. Here, time is divided into discrete and equal time slots, e.g., $\{1, \cdots, T\}$. We assume the throwboxes can only be placed in a finite number of locations instead of any places in the 2D plane. This is reasonable since in practice throwboxes cannot be placed in any places due to existing obstacles or high deployment costs to certain locations. Let $V = V_{user} + V_{throwbox}$ be the whole set of all nodes. 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. 1(a) 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 snapshots in Fig. 1(a)). This makes routing tasks over them very 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. 1(b) 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 $\mathcal{V} = \{v_j^t | j = 1, \cdots, n+m \text{ and } t = 0, \cdots, T\}$ and there are $(n+m)(T+1)$ nodes in total. 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. 1(b)) connects the same node v_j across consecutive $(t-1)$th and tth layers, which represents the node carrying the message in the tth time slot. A spacial link $\overrightarrow{v_j^{t-1} v_k^t}$ represents forwarding a message from one node v_j to its neighbor v_k in the tth 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. Any space-time path from v_i^0 to v_j^4 shows a particular DTN routing strategy to deliver the packet from v_i to v_j in the network using 4 time slots.

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 original 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 *reliability 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 reliability probability of each link can be obtained through link estimation techniques at the link and physical layers [16] or mobility prediction techniques [17]. Given the reliability of each link, we can then define the reliability of a path P or a structure $\mathcal{H}$. 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 $\mathcal{H}$, we can define the *most reliable path* $P_r^{\mathcal{H}}(u, v)$ as the path from u to v in $\mathcal{H}$ with the highest reliability. Let $r^{\mathcal{H}}(u, v) = \Pi_{e \in P_r^{\mathcal{H}}(u,v)} r(e)$ be the reliability of path $P_r^{\mathcal{H}}(u, v)$. Then the *reliability* of the topology $\mathcal{H}$ is defined as follows

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

Notice that when $\mathcal{H}$ is given, it is easy to calculate $r(\mathcal{H})$ by using any shortest path algorithms.

III. Throwbox Placement Problem

With the helps from throwboxes there will be more forwarding opportunities among devices, thus it increases the reliability of the network. However, the deployment of throwboxes has certain cost. Therefore, we limit the number of throwboxes to a constant k due to fixed budget of the network operator. We then need to decide where to put them while achieving the best reliability. We now formally define the *k-throwbox placement problem* over the weighted space-time graph model.

Definition 1: Given a connected and weighted space-time graph $\mathcal{G}$ (with n mobile users and m potential throwboxes)

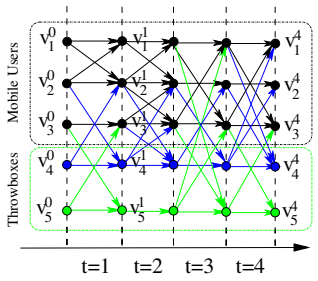


Fig. 2. Select one throwbox (marked as blue).

and a constant k ($k \leq m$), the aim of *k-throwbox placement problem* is to find a space-time graph $\mathcal{H}$ with n mobile users and k throwboxes (such as Fig. 2 with one selected throwbox), which is a subgraph of $\mathcal{G}$, such that $\mathcal{H}$'s reliability is maximized.

This newly defined throwbox placement problem is different from traditional relay node placement problems [10], [11], since the network is not static but evolves over time. It is also different from the topology design (TD) problems [18], [19], which aim to build a sparse subgraph while guaranteeing the connectivity or reliability. In TD, arbitrary links from any node at any time slot can be removed. To prove the NP-hardness of the k-throwbox placement problem, we will use the following theorem.

Theorem 1: For a submodular function f, if f only takes non-negative values and is monotone, finding a k-element set A for which $f(A)$ is maximized is an NP-hard optimization problem [20], [21].

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 throwboxes 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 throwboxes 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. ■

By combining Theorem 1 and Lemma 1, the following theorem about NP-hardness of the k-throwbox placement problem immediately follows:

Theorem 2: The *k-throwbox placement problem* is NP-hard.

IV. Throwbox Placement Algorithms

Since the k-throwbox placement problem is NP-hard, in this section, we propose a set of different heuristics to carefully select throwbox placements to maximize the reliability. 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 $M = O((n+m)^2 T)$.

A. General Greedy Frameworks: Adding or Removing One Throwbox Per Round

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 frameworks simply make a single throwbox choice in each round by adding or removing one throwbox from the network. The procedure will guarantee to terminate after at most k or $m - k$ rounds, which is much more efficient than exponential brute force algorithm. The detailed general frameworks are given in Algorithm 1 and Algorithm 2.

The first algorithm (GrdAddTBs) starts with a space-time graph $\mathcal{H}$ only including mobile users $(v_1, v_2, \cdots, v_n)$. Then it greedily adds in more throwboxes until k throwboxes are selected. The second algorithm (GrdDelTBs) starts with the original space-time graph $\mathcal{G}$ with all throwboxes selected, and gradually deletes throwboxes until only k throwboxes are left. In both algorithms, during the process, it greedily selects one single throwbox in each round based on certain criteria (we will introduce different criteria in the next subsection). 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 $\mathcal{H}$. The time complexities of GrdAddTBs and GrdDelTBs are $O(kX)$ and $O((m-k)X)$, respectively.

Algorithm 1 Greedy - Adding Throwboxes (GrdAddTBs)

Input: the original space-time graph $\mathcal{G}$ (including potential throwbox set $V_{throwbox}$), a constant k.

Output: the selected throwbox set $V_{selected-throwbox}$ and the corresponding new space-time graph $\mathcal{H}$.

1: $\mathcal{H} \leftarrow \mathcal{G} - \{V_{throwbox}\}$ and $V_{selected-throwbox} = \emptyset$
2: **while** $|V_{selected-throwbox}| < k$ **do**
3: Greedily select a throwbox v_i from all unselected throwboxes $V_{throwbox} - V_{selected-throwbox}$, i.e., $v_i = GreedySelect(V_{throwbox} - V_{selected-throwbox}, \mathcal{H})$
4: $\mathcal{H} \leftarrow \mathcal{H} + \{v_i\}$
5: $V_{selected-throwbox} \leftarrow V_{selected-throwbox} + \{v_i\}$
6: **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.

Output: the selected throwbox set $V_{selected-throwbox}$ and the corresponding new space-time graph $\mathcal{H}$.

1: $\mathcal{H} \leftarrow \mathcal{G}$ and $V_{selected-throwbox} = V_{throwbox}$
2: **while** $|V_{selected-throwbox}| > k$ **do**
3: Greedily select a throwbox v_i from $V_{selected-throwbox}$, i.e., $v_i = GreedySelect(V_{selected-throwbox}, \mathcal{H})$
4: $\mathcal{H} \leftarrow \mathcal{H} - \{v_i\}$
5: $V_{selected-throwbox} \leftarrow V_{selected-throwbox} - \{v_i\}$
6: **return** $V_{selected-throwbox}$ and $\mathcal{H}$

B. Greedy Heuristics: 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.

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 $O(mnT)$.

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 terms 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 framework. 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.

C. 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, as we have proved in Lemma 1, the reliability function $r(\mathcal{H})$ is non-negative, monotone, and submodular. Submodular functions have a number of nice properties. One of them is a result from [20], [21], 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 an $(1-1/e)$-approximation.

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

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

V. Simulations

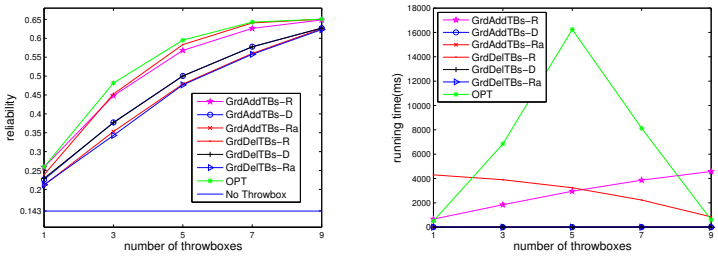
To evaluate our proposed algorithms for k-throwbox placement problem, we have conducted extensive simulations over randomly generated time-evolving networks and real DTNs extracted from realistic contact traces [22]. We implement and test the following algorithms:

- **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.
- **OPT:** the optimal solution for throwbox optimization problem obtained by brute force method.

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., $r(\mathcal{H})$) and the actual running time of each method.

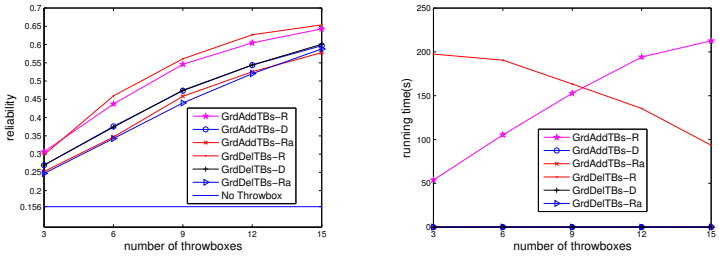
A. Simulations on Random Time-evolving Networks

We first test our algorithms on randomly generated networks. We generate a sequence of static random graphs with



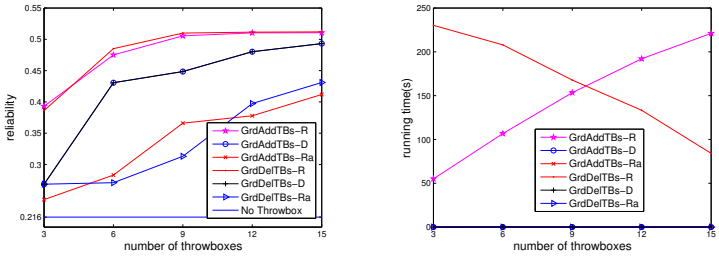
(a) $r(\mathcal{H})$ (b) running time

Fig. 3. Results on random networks ($n = 20$, $m = 10$).



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

Fig. 4. Results on random networks ($n = 50$, $m = 20$).



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

Fig. 5. Simulation results on networks from Infocom 2006 trace data [22] ($n = 40$ and $m = 20$ static throwboxes).

$n+m$ 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.

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. For these small networks, we are able to find the optimal solution OPT with brute force algorithm. Fig. 3(a) shows the reliabilities achieved 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 of the network without any throwboxes (i.e., V_{user}). Fig. 3(b) also plots the running time of each algorithm. Note that OPT needs to search $\binom{m}{k}$ times to find the optimal throwboxes deployment, thus when the number of throwbox is 5, its running time is the longest. Via these two figures, we can conclude: (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) perform poorly in terms 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 time 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 ($n = 50$ and $m = 20$) to discover the scalability of our algorithms. Here, we do not obtain the OPT since the running time of bruce force algorithm is too long for these large networks. Fig. 4 shows the results. We can draw the similar conclusions with the previous smaller networks. Overall, GrdAddTBs-R and GrdDelTBs-R can achieve the highest reliability.

B. Simulations on Real DTN Tracing Data

Taking advantages of public wireless tracing data, we also test our algorithms over a realistic contact trace dataset: the Infocom 2006 trace data [22]. 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. Fig. 5 shows the results. 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 the optimization problem.

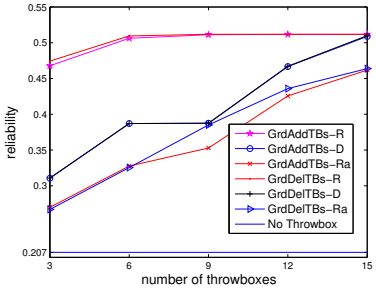


Fig. 6. Simulation results $r(\mathcal{H})$ on networks from Infocom 2006 trace data [22] ($n = 40$ and $m = 20$ mobile throwboxes).

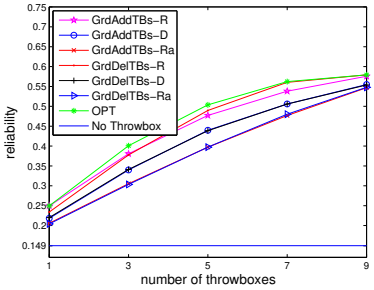


Fig. 7. Results on random networks ($n = 50$, $m = 20$) with short-memory throwboxes.

VI. Variations on Types of Throwboxes

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 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. 6 shows a set of results from experiments over Infocom traces [22], 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.

Throwboxes with Short-Memory: In our model, we assume that each throwbox can hold any amount of packets for any duration of time period. In practice, a throwbox device may have capacity to hold only certain amount of packets or have limited energy resource so that it can only hold packets for certain duration. One of the extreme cases is that throwboxes can only hold the packet within each time slot, in other words, they cannot buffer the packets longer than one time slot. Our weighted space-time graph can easily handle this case by setting the reliabilities of every temporal link of all throwboxes to zero. Fig. 7 shows results for such setting in random time-evolving networks. Again, our algorithms can handle this scenario well, too.

VII. Conclusion

Recent studies have shown the enhancement of DTN performances with throwboxes. This paper investigates a key problem, throwbox placement, in a time-evolving throwbox-assisted DTN modeled by a weighted space-time graph. The k-throwbox placement problem is formally introduced and shown to be NP-hard. A set of greedy algorithms which can efficiently provide quality solutions are then proposed. We show the efficiency of the proposed methods through extensive simulations over both random time-evolving DTNs and real life DTN traces.

References

[1] P. Hui, J. Crowcroft, and E. Yonek, "Bubble rap: Social-based forwarding in delay tolerant networks," in *Proc. of ACM MobiHoc*, 2008.
[2] C. Liu and J. Wu, "Scalable routing in delay tolerant networks," in *Proc. of ACM MobiHoc*, 2007.
[3] V. Erranmilli, M. Crovella, A. Chaintreau, and C. Diot, "Delegation forwarding," in *Proc. of ACM MobiHoc*, 2008.
[4] W. Zhao, Y. Chen, M. Ammar, M. Corner, et al., "Capacity enhancement using throwboxes in DTNs," in *Proc. of IEEE MASS*, 2006.
[5] N. Banerjee, M. D. Corner, and B. N. Levine, "Design and field experimentation of an energy-efficient architecture for DTN throwboxes," *IEEE/ACM Transactions on Networking*, 18(2):554–567, 2010.
[6] M. Ibrahim, A.Al Hanbali, et al., "Delay and resource analysis in MANETs in presence of throwboxes," *Perf. Eva.*, 64(9):933-947, 2007.
[7] B. Gu and X. Hong, "Latency analysis for thrown box based message dissemination," in *Proc. of IEEE GLOBECOM*, 2010.
[8] M. Ibrahim, P. Nain, and I. Carreras, "Analysis of relay protocols for throwbox-equipped DTNs," in *Proc. of WiOPT*, 2009.
[9] B. Gu and X. Hong, "Capacity-aware routing using throw-boxes," in *Proc. of IEEE GLOBECOM*, 2011.
[10] E. L. Lloyd and G. Xue, "Relay node placement in wireless sensor networks," *IEEE Transactions on Computers*, 56(1):134–138, 2007.
[11] X. Cheng, D.-Z. Du, L. Wang, and B. Xu, "Relay sensor placement in wireless sensor networks," *Wireless Networks*, 14(3):347–355, 2008.
[12] S. Merugu, M. Ammar, and E. Zegura, "Routing in space and time in networks with predictable mobility," Tech. Rep. GIT-CC-04-07, 2004.
[13] J. Monteiro, A. Goldman, and A. Ferreira, "Performance evaluation of dynamic networks using an evolving graph combinatorial model," in *Proc. of IEEE WIMOB*, 2006.
[14] C. Liu and J. Wu, "Routing in a cyclic mobispace," in *Prof. of ACM MobiHoc*, 2008.
[15] L. Arantes, A. Goldman, and M. V. D. Santos, "Using evolving graphs to evaluate DTN routing protocols," in *Proc. of ACM ExtremeCom*, 2009.
[16] N. Baccour, et al., "Radio link quality estimation in wireless sensor networks: A survey," *ACM TOSN*, 8(4), 2012.
[17] W. Su, S.-J. Lee, and M. Gerla, "Mobility prediction in wireless networks," in *Proc. of IEE MILCOM*, 2000.
[18] M. Huang, S. Chen, et al., "Topology control for time-evolving and predictable delay-tolerant networks," in *IEEE MASS*, 2011.
[19] M. Huang, S. Chen, et al., "Topology design in time-evolving delay-tolerant networks with unreliable links," in *IEEE Globecom*, 2012.
[20] G. Cornuejols, M. L. Fisher, and G. L. Nemhauser, "Location of bank accounts to optimize float: An analytic study of exact and approximate algorithms," *Management Science*, 23(8), 1977.
[21] G. Nemhauser, L. Wolsey, and M. Fisher, "An analysis of approximations for maximizing submodular set functions-I," *Mathematical Programming*, 14(1):265–294, 1978.
[22] J. Scott, R. Gass, J. Crowcroft, P. Hui, C. Diot, and A. Chaintreau, "CRAWDAD trace set cambridge/haggle/imote (v. 2009-05-29)," Downloaded from http://crawdad.cs.dartmouth.edu/cambridge/haggle/imote.