Deployment of a Connected Reinforced Backbone Network with a Limited Number of Backbone Nodes

Shan Chu, *Student Member, IEEE,* Peng Wei, *Student Member, IEEE,* Xu Zhong, *Student Member, IEEE,* Xin Wang, *Member, IEEE,* and Yu Zhou, *Member, IEEE,*

Abstract—In recent years, we have witnessed a surge of interest in enabling communications over meshed wireless networks. Particularly, supporting peer-to-peer communications over a multi-hop wireless network has great potential in enabling ubiquitous computing. However, many wireless nodes have limited capabilities, for example, sensor nodes or small handheld devices. Also, the end-to-end capacity and delay degrade significantly as the path length increases with the number of network nodes. In these scenarios, the deployment of a backbone network could potentially facilitate higher performance network communications. In this paper, we study the novel Reinforced Backbone Network (RBN) deployment problem considering the practical limitation in the number of available backbone nodes and enforcing backbone network connectivity. We propose an iterative and adaptive (ITA) algorithm for efficient backbone network deployment. In addition, in order to provide the performance bound, we redefine and solve the problem by implementing the Genetic Algorithm. Finally, we present our simulation results under various settings and compare the performance of the proposed ITA algorithm and the genetic algorithm. Our study indicates that the proposed ITA algorithm is promising for deploying a connected RBN with a limited number of available backbone nodes.

Index Terms—Deployment, wireless, backbone.

1 Introduction

Supporting peer-to-peer communications over a meshed wireless network has shown a great potential in enabling ubiquitous computing. However, when the nodes have lower capabilities or limited resources, such as small sensor nodes or handheld wireless devices, the network may be unreliable or have a very low capacity. In addition, a flat homogeneous ad hoc network has been shown to have poor scalability. As the number of network nodes and therefore the average number of hops per path increases, there is a rapid reduction of path throughput [1] and an increase of the end-to-end delay [2]. The placement of a backbone network with more capable nodes can potentially bring in a lot of benefits in these scenarios, including higher quality links, more reliable transmissions, lower delay and higher throughput to remote destination nodes.

The backbone network deployment problems have been recently studied in [3]–[5]. The works in [3] [4] assume there is an unlimited number of backbone nodes, and the goal is to minimize the total number of backbone nodes in the deployment. In many practical scenarios, however, there is only a fixed number of backbone nodes that can be deployed, and the deployment can be only performed under the constraint of the available backbone resources. Although the authors in [5] also perceived the issues and attempted to deploy a limited number of backbone nodes, they failed to consider an important constraint, i.e., backbone network connection. In addition, the paper implicitly assumes that a regular node can reach any backbone nodes directly, which is not very practical.

The aim of this work is to optimally deploy a *Reinforced Backbone Network* to enhance the performance and robustness of an underlying wireless network that consists of nodes with lower capabilities and to facilitate high capacity and long-range network communications. We consider two types of wireless nodes in a network. The first type of nodes are called regular nodes (RNs), which normally have limited capacities and transmit at a shorter range. The second type of nodes are called backbone nodes (BNs), which generally have much higher communication and computation capacities and can transmit at a longer range. As the cost metric, we consider the factors that impact the delay for an RN to access the backbone network, including the number of hops from an RN to its associated BN and the competition delay due to a large number of RNs trying to access the backbone network through the same BN.

The objective of our backbone deployment is to min-

- *Shan Chu and Xin Wang are with the Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook, NY 11794. E-mail: {schu,xwang}@ece.sunysb.edu.*
- *This work was performed while Peng Wei was with the Department of Electrical and Computer Engineering, Stony Brook University. He is currently with the School of Aeronautics and Astronautics, Purdue University, West Lafayette, IN 47907. E-mail: weip@purdue.edu.*
- *Xu Zhong and Yu Zhou are with the Department of Mechanical Engineering, Stony Brook University, Stony Brook, NY 11794. E-mail: zhongxu2@gmail.com,yuzhou@notes.cc.sunysb.edu.*
- *Xin Wang's research was supported by U.S. National Science Foundation under grant numbers CNS-0751121 and CNS-0628093.*

imize the average backbone access delay from all the regular nodes while satisfying the backbone connection constraint. To our best knowledge, this is the first work that studies the optimal deployment of backbone network with use of a limited number of backbone nodes and ensuring backbone connectivity. The backbone deployment problem is made much more challenging with the practical consideration of the limitation of available backbone nodes and the enforcement of backbone network connectivity. We formulate the problem with a practical objective function and propose an iterative and adaptive algorithm to solve the problem. In addition, in order to find the performance bound, we re-define the problem and solve the problem using genetic algorithm.

The rest of the paper is organized as follows. In Section 2, we formulate the problem and discuss its complexity. In Section 3, we present the iterative and adaptive backbone deployment algorithm. We re-define the problem and solve the problem through genetic algorithm in Section 4. In Section 5 we evaluate the performance of our algorithm via simulations, and compare the performance of our iterative and adaptive algorithm and genetic algorithm. Section 6 provides an overview of the related work and Section 7 concludes this paper.

2 Problem Formulation

In this section, we introduce the concepts and present the formulation of our problem.

Before formulating the problem, we first introduce our link and network connection models. For a sending node i and a receiving node j, the receiving *signal to interference and noise ratio* (SINR) at j is defined as:

$$SINR_{i,j} = \frac{G_{i,j} \cdot p_t(i) \cdot d_{i,j}^{-\beta}}{N + I_j}, \quad (1)$$

As the backbone deployment is at a larger time scale, we only consider the large scale path loss factor in our link model. In Eq. (1), $G_{i,j}$ is the channel gain between nodes i and j, $p_t(i)$ is the transmitting power of i, $d_{i,j}$ is the distance between i and j, β is the path loss exponent typically ranging between 2 and 4, N is the ambiance noise power and I_j is the interference power at the receiving node j.

Definition 1 (Link): Given a threshold γ_j depending on the decoding capability of node j, a node i can reach node j if $SINR_{i,j}$ is larger than γ_j. There is a link between i and j if i can reach j and j can reach i.

Definition 2 (Path): There is a path $P_{i,j}$ between two nodes i and j if i and j can reach each other directly through one link or over multiple links with relay nodes.

Definition 3 (Connected): A network is connected if $\forall$ node pair i and j in the network, there is a path $P_{i,j}$ between i and j.

Suppose that there are n wireless Regular Nodes (RN) in a 2D plane, which form a *connected* ad hoc network $G_R = (N_R, E_R)$. The set N_R contains all the Regular Nodes, and $|N_R| = n$ is the size of the RN network. We name the RNs as a set $\{1, 2, ..., n\}$. The link between any RN pair i and j is denoted as e_{ij}, and the set E_R contains all the links in the RN network.

There are k backbone nodes (BN) to be deployed to form a Reinforced Backbone Network (RBN) to enable RNs to communicate more efficiently over a long distance. The fixed number of BNs is inspired by the practical case that the budget for BNs is limited. Each BN has two communication interfaces, one is used to communicate with RNs, and the other is used to communicate with other BNs. The two radio interfaces are tuned to different radio channels so concurrent communications can be carried in the RN network and the backbone network, and the long-range backbone communications does not interfere with the short-range communications in RN network. After the deployment, the backbone network can be denoted as $G_B = (N_B, E_B)$, where N_B is the set of BNs with $|N_B| = k$, and E_B is the set of backbone links. We denote the BNs as a set $\{b_1, b_2, ..., b_k\}$.

2.1 Objective Function

It is important to optimally deploy the backbone network to achieve a desired objective. Based on [2] and [6], one of the major delay factors in a random access based wireless network is the hop-number from the transmitting node to the receiving node. When an RN needs to communicate with another node that is farther away through a path over only RNs, there could be a large number of hops between the source and the destination, which not only incurs a high transmission delay but also leads to a low end-to-end throughput. Therefore, it is necessary to introduce a Reinforced Backbone Network to provide efficient long-range communication. As shown in Fig. 1, RN_t can take advantage of the backbone network to speed up the communication with RN_r. The communication has three parts: RN_t to BN_a, BN_a to BN_b inside the backbone network and BN_b to RN_r. During the backbone network construction time, the actual transmission needs are not known yet. Also, a BN has a much higher transmission bandwidth and longer transmission range than an RN, thus the delay between two BNs is much smaller than that between two RNs with equal distance. Therefore, to facilitate the long-range communication of regular nodes, it is critical to reduce the delay for an RN to access the backbone network.

For each RN i, there is an assigned BN $b(i)$ for it to efficiently access the backbone network. As the transmission delay is directly impacted by the number of hops in a path, we consider the number of hops $h(i)$ between the RN i and $b(i)$ as *hop delay* between the RN and the backbone network. Besides the transmission delay, if too many RNs want to route their packets into the backbone network through the same BN, the BN would become the *hot spot*, leading to a large *competition delay*. Therefore, for each RN, we consider a delay cost factor as a function of both hop delay and competition delay.

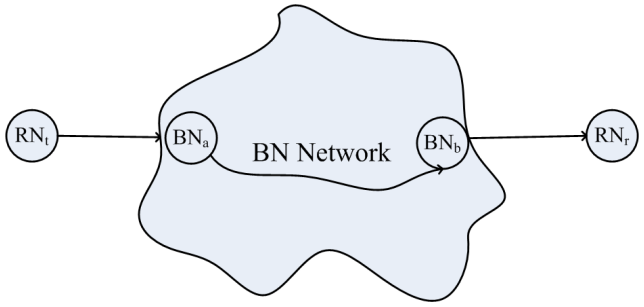


Fig. 1. An end to end long range RN transmission.

Recall that the regular nodes are denoted by the set $N_R = \{1, 2, ..., n\}$ and the backbone nodes are denoted by the set $N_B = \{b_1, b_2, ..., b_k\}$. For the regular node i, $b(i) = b_k$ represents that the regular node i is assigned to the k-th backbone node. Let $h(i, b_J)$ be the hop number from the regular node i to the backbone node b_J, and $h(i)$ be the hop number from RN i to its assigned BN $b(i)$.

For a backbone node b_I, the RNs assigned to the BN form a cluster, and $|(b_I)|$ represents the number of RNs associated with b_I. For k BNs, there are k groups of RNs, and each RN group is associated with one of the k BNs. In this work, every RN is assigned to exactly one BN, thus $\sum_{I=1}^{k} |b_I| = n$. An RN can be assigned to a different BN if its associated BN becomes unavailable, and could also route around the assigned BN over a path of RNs if the BN is not directly reachable .

Before the deployment of backbone nodes, the signal strength cannot be measured, we use the reference transmission ranges R for BNs and r for RNs to guide the backbone node deployment. Generally, we just need to know $R > r$, and our algorithm is not constrained by the disk model. The connection between two neighboring nodes needs to be calculated based on the receiving $SINR$ defined in Eq. (1) with a safety threshold γ to ensure the connectivity under some fading conditions.

When multiple RNs are associated with one BN, there exists channel competition in accessing the BN. If each node has an equal probability of accessing the BN, the delay as a result of node competition is directly impacted by the number of RNs associated with the BN. To capture the impact of both hop delay and competition delay, the delay cost factor c_i between an RN i and its assigned BN $b(i)$ can be represented as:

$$c_i = \alpha h(i) + (1 - \alpha)|b(i)|, \tag{2}$$

where $|b(i)|$ indicates how many RNs are assigned to $b(i)$. The parameter $\alpha \in [0, 1]$ is used to adjust the trade-off between the hop delay and the competition delay. Note that this cost function is only defined to guide the deployment of backbone nodes, and we only consider the factors that impact the backbone access delay. As a matter of fact, the accurate transmission delay is difficult to know during the deployment phase, and it is difficult to model the exact relationship between the two cost factors as many dynamic factors such as traffic and transmission collisions in each hop are unknown. In this paper, we focus on designing a general deployment algorithm, and the accurate modeling of the cost function is out of our scope. As the goal of deployment is for the longer period network planning, our algorithm does not depend on specific MAC scheme. If a MAC with well scheduled transmission slot such as TDMA is used, there would be no backoff.

To evaluate the overall backbone network deployment performance, we consider the average backbone access delay cost factor $\bar{c}$ of all RNs as follows:

$$\bar{c} = \frac{1}{n}\sum_{i=1}^{n} c_i = \frac{1}{n}\sum_{i=1}^{n} (\alpha h(i) + (1 - \alpha)|b(i)|), \tag{3}$$

and the objective of the backbone network deployment is to minimize $\bar{c}$.

2.2 The Problem

Our problem is to deploy k BNs to form a Reinforced Backbone Network where the BNs are connected and each RN i is assigned to exact one BN $b(i)$, with the objective of minimizing the average backbone access delay cost of all RNs. The problem is described as follows:

$$\min \bar{c}, \tag{4}$$

Subject to:

$$\exists P_{I,J}, \forall b_I, b_J \in N_B,$$
$$\exists b(i) \in N_B, \forall i \in N_R.$$

A solution to this problem involves two parts: the deployment of k BNs and the assignment of each RN to a BN.

In order to provide the Reinforced Service to RNs, an RN should be able to access at least one BN, either directly or through multi-hop RN relays. As the objective of the backbone deployment is to minimize the average backbone access delay from all RNs, thus for a lower overall access delay and connectivity from an RN to the BN network, each BN should be within the transmission range of at least one RN. Consider each RN corresponds to a transmission area, and each BN can choose which RN transmission area it should stay. In total, there are n^k candidate deployment options. With the BN network connection constraint, only some of these candidate deployment locations are feasible. As a result, there are fewer than n^k potential ways of feasible deployment. Because k is known as a constant, the deployment solution has a polynomial complexity. As we assume $R > r$, we can first *virtually* deploy a BN to the center of an RN transmission area, i.e., the current position of the RN. In this way, the RN positions serve as the reference searching points to coarsely determine the BN positions, which makes the deployment possible and at a lower

computational cost. The actual deployment position of BN can be close to the RN and different, and adjusting the position of the BN within a short distance of the RN will not affect the connection constraint significantly. After physically deploying the BNs, a BN can adjust its position within the corresponding RN's range while maintaining the connectivity with its neighbors through physical signal strength measurement.

After the deployment of all BNs, the next step is to associate each RN to a specific BN. Since the RN network is already given, we can have the following hop number matrix H:

$$H_{(n\times n)} = \begin{pmatrix} h(1,1) & h(1,2) & \dots & h(1,n) \\ h(2,1) & h(2,2) & \dots & h(2,n) \\ \vdots & \vdots & \ddots & \vdots \\ h(n,1) & h(n,2) & \dots & h(n,n) \end{pmatrix}, \quad (5)$$

where the item $h(i,j)$ is the shortest path hop number between RNs i and j, and $h(i,i)$ is 0 for any RN i. Apparently, H is a natural symmetric matrix. Assuming the BN nodes $\{b_1, b_2, ..., b_k\}$ have already been deployed on the existing RN positions, the hop number from an RN to every BN can be found in the matrix H. Note that given the locations of BN nodes as the candidate facility locations, and the hop number $h(i, b_J)$ from an RN i to each BN b_J as the connection cost, if we let $\alpha = 1$ in Eq. (3), the simplified problem is equivalent to the NP-hard k-facility location problem [7]. As a result, the assignment part of the problem is NP-hard and thus the problem defined in Eq. (4) is NP-hard. Therefore, it is important to design a practical algorithm with acceptable complexity to solve the problem efficiently.

3 Iterative and Adaptive Backbone Deployment

As discussed earlier, in order to minimize the average backbone network access delay cost, it would be good for BNs to stay close to the RNs. Specifically, each BN should be within the transmission range of one or more RNs. Considering the transmission area of each RN as a deployment option for a BN, with n RNs and k BNs, there are $O(n^k)$ combinations for the deployment. Instead of searching through all the $O(n^k)$ possible combinations for BN deployment and all the $O(k^n)$ possible association between n RNs and k BNs which takes a significantly long time, we propose an iterative and adaptive (ITA) backbone deployment algorithm. The algorithm has four steps:

1) Initial deployment to determine the initial positions of k BNs;
2) RN association, which greedily assigns the RNs to associate with k BNs based on the current round of BNs deployment to minimize the average backbone network access delay cost;
3) Adaptation of the positions of k BNs based on the association of n RNs;
4) Checking the connectivity to ensure that the k BNs forms a connected backbone network.

The algorithm runs iteratively through the steps 2, 3 and 4 until either the objective function cannot be improved any more or the BN network becomes disconnected.

3.1 Initial Deployment of Backbone Nodes

A simple solution of initial deployment is to randomly pick k RN positions and put k BNs close to these reference locations. However, this cannot guarantee that k BNs are connected. The classic Furthest First [8] scheme or Subset Furthest First [9] scheme also cannot ensure that the initial BN deployment meets the connection constraint. Motivated by the self-deployment scheme in robotic research area [10] where robot nodes spread out from a central location until no node could move out further, we deploy the k BNs initially within a close distance between each other so that the backbone network is connected. Different from robot deployment which only considers a flat network with a simple goal of keeping the nodes connected, the problem is made much more challenging with the objective of deploying a limited number of backbone nodes to optimally serve the RNs while ensuring the backbone network connectivity. Note that in our deployment, the change of BN positions is *virtual*, i.e. no actual placement is involved, until a solution is found.

In order to reduce the number of transmission hops from RNs to their assigned backbone nodes for a lower hop delay and balance the association between the n RNs and k BNs for a lower competition delay, we choose to initially deploy the BNs close to the mass center $\vec{L}_{mass}$ of the n RNs, i.e. the mean location of all the RNs in the network. As discussed earlier, we use the RN positions as the reference to find the BN deployment points. Denote the locations of the RNs as vectors $\vec{L}_1^{RN}, \vec{L}_2^{RN}, ..., \vec{L}_n^{RN}$. We first find the RN position closest to the mass center, denoted as $\vec{L}_{(0)1} = \vec{L}_{i^*}^{RN}$ where $i^* = \arg\min_{i\in RN} |\vec{L}_i^{RN} - \vec{L}_{mass}|$, to virtually deploy the first BN. In order to ensure the backbone network connectivity, we initially keep the remaining $k-1$ BNs as close to the first BN as possible. Since the n RNs already form a connected network, a Breadth First Search (BFS) can be performed to traverse the RN network and a spanning tree consisting of the n RN positions is derived. Note that nodes on the d-th layer of the tree returned by BFS is exactly d hop away from the root node. The remaining $k-1$ BNs are then virtually deployed on the positions of the $k-1$ RNs closest to the root BN in terms of the hop number on the tree. The k locations of the initial BN positions are denoted as $\vec{L}_{(0)1}, \vec{L}_{(0)2}, ..., \vec{L}_{(0)k}$, where (0) indicates the index of the iteration during the progressing of the algorithm, which is currently in the initial stage. As the transmission range of a BN is much larger than an RN, the connectivity of the k selected RN locations in the RN network ensures that the initial deployment of BNs forms a connected BN network.

3.2 Random Greedy Assignment

Given the locations of the k BNs in iteration t, $\vec{L}_{(t)1}, \vec{L}_{(t)2}, ..., \vec{L}_{(t)k}$, there is a need to associate each RN to a BN to reduce its delay in accessing the backbone network. In this section, we first formulate the assignment problem, and then propose a random greedy solution.

Without loss of generality, we assume $b_1, b_2, ..., b_k$ have been deployed on RN positions $1, 2, ..., k$ and it's a feasible deployment. Then we can show that the assignment part of the objective function in Eq. (4) is a the convex cost flow problem [11] [12], with the general format as below:

$$\min x'Qx + c'x$$

$$\text{subject to} \begin{cases} \sum x_{out} - \sum x_{in} = n, & \text{for source node } s; \\ \sum x_{out} - \sum x_{in} = -n, & \text{for sink node } t; \\ \sum x_{out} - \sum x_{in} = 0, & \text{for all the other nodes;} \end{cases} \tag{6}$$

$$\text{and } 0 \le x_e \le u_e, e \in E.$$

Next, we construct a flow problem in Fig. 2. We have a source node s on the left side and a sink node t on the right side, as well as a column of nodes denoting the RNs $1, 2, ..., n$ and another column of nodes denoting the BNs $b_1, b_2, ..., b_k$ in the middle.

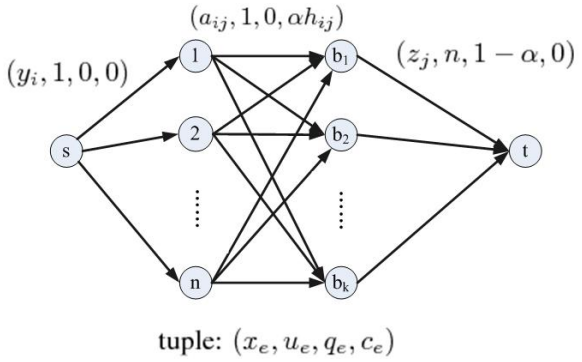


Fig. 2. Convex Cost Flow

For every node in Fig. 2, the outgoing flow equals to the incoming flow except for the source s and the sink t, satisfying Eq. (6). For each arc e in the flow, it has a flow x_e, a flow upper bound u_e, a quadratic cost q_e and a linear cost c_e. The tuple (x_e, u_e, q_e, c_e) is used to represent the state of each arc.

There are three types of arcs. The arcs originated from node s are represented as $y_1, y_2, ..., y_n$, with the flow upper bound on each arc being 1 and all the costs being 0. Thus the quadruple of these arcs is $(y_i, 1, 0, 0)$. The arcs terminated the flows at node t are represented as $z_1, z_2, ..., z_k$. The upper bound, the quadratic cost, and the linear cost on each arc are n, $(1-\alpha)$ and 0 respectively. Thus the quadruple is $(z_j, n, 1-\alpha, 0)$. The arc starting from an RN i and ending at a BN b_j between the middle two columns has flow value a_{ij}, upper bound 1, quadratic cost 0, and linear cost αh_{ij}. The parameter h_{ij} is obtained by looking up the hop number matrix in Eq. (5). The quadruple of this kind of arcs is $(a_{ij}, 1, 0, \alpha h_{ij})$.

We define the flow function in Fig. 2 as:

$$\sum_{i=1}^{n}(0y_i + 0y_i^2) + \sum_{i=1}^{n}\sum_{j=1}^{k}(\alpha h_{ij}a_{ij} + 0a_{ij}^2) + \sum_{j=1}^{k}(0z_j + (1-\alpha)z_j^2),$$

which can be simplified as:

$$\sum_{i=1}^{n}\sum_{j=1}^{k}\alpha h_{ij}a_{ij} + \sum_{j=1}^{k}(1-\alpha)z_j^2. \tag{7}$$

Next we show that the constructed flow function has the same format as our objective function in Eq. (4). In Fig. 2, there is exactly one unit flow coming out of each RN, as each RN on the left can only be assigned to one BN $b(i)$ on the right. So for RN i, there is only one a_{ij} is 1 and others are all 0, and $h_{ij} = h(i)$ for $b_j = b(i)$. Therefore, we have

$$\sum_{i=1}^{n}\sum_{j=1}^{k}\alpha h_{ij}a_{ij} = \sum_{i=1}^{n}\alpha h(i). \tag{8}$$

For the BNs, since $\sum x_{out} - \sum x_{in} = 0$, all the unit flows into a BN b_I need to flow out. The number of unit flows into b_I is determined by the number of RNs assigned to this BN. So the flow z_j coming out of the BN b_I is equal to the number of RNs assigned to b_I, which is denoted by $|b_I|$. As a result, we have

$$\begin{aligned} & \sum_{j=1}^{k} z_j^2 = \sum_{I=1}^{k} |b_I|^2 \\ = & \sum_{i:b(i)=b_1} |b_1| + \sum_{i:b(i)=b_2} |b_2| + ... + \sum_{i:b(i)=b_k} |b_k| \\ = & \sum_{i=1}^{n} |b(i)|. \end{aligned} \tag{9}$$

By substituting the results in Eq. (8) and (9) into Eq. (7), the cost of our flow problem turns to be

$$\begin{aligned} & \sum_{i=1}^{n}\sum_{j=1}^{k}\alpha h_{ij}a_{ij} + \sum_{j=1}^{k}(1-\alpha)z_j^2 \\ = & \sum_{i=1}^{n}\alpha h(i) + \sum_{I=1}^{k}(1-\alpha)|b_I|^2 \\ = & \sum_{i=1}^{n}\alpha h(i) + \sum_{i=1}^{n}(1-\alpha)|b(i)|. \end{aligned} \tag{10}$$

Eq. (10) is exactly the same as our objective function in Eq. (4) except one more divisor n, which can be ignored here. So far, we have formulated the assignment part of our problem as a convex cost flow problem.

According to [12] some variant of the convex cost flow problem can be solved through a polynomial time

algorithm. However, the complexity is still considerably high. Therefore, we propose a Random Greedy Assignment (RGA) approach, in which the n RNs take turns to associate with a BN in a random order as shown in Algorithm 1.

The algorithm is performed at each iteration round. At the beginning of the association process, the association results from the last round is cleared up as in Line 2. The RNs are associated with BNs one by one in a random sequence to reduce the occurrence of any unpreferable solution. For the assignment of each RN, the access delay cost from the RN to any BN is calculated based on Eq. (2), and the RN is associate with the BN that provides the least access cost, as on lines 5-9. After the association of RN i is completed, the number of RNs associated with the selected BN $b(i)$ is increased by one, as on line 10, so that the updated competition delay can be used in the next iteration of the loop. By considering the current load of the BNs, the algorithm attempts to balance the access load among k BNs to reduce the competition delay in accessing the backbone network. The complexity of the algorithm is $O(kn)$.

Algorithm 1 Random Greedy Assignment

1: $L \leftarrow \{1, 2, ..., n\}$
2: $|b_I| \leftarrow 0, \forall b_I \in \{b_1, b_2, ..., b_k\}$
3: **while** $L \neq \emptyset$ **do**
4: Randomly select an RN i from L
5: **for** $I = 1$ to k **do**
6: $b(i) = b_I$
7: $c_i^I = \alpha h(i) + (1-\alpha)|b(i)|$
8: **end for**
9: $i^* = \arg\min_I c_i^I$, $b(i) = b_{i^*}$
10: $|b(i)| = |b(i)| + 1$
11: $L = L \setminus \{i\}$
12: **end while**

Due to the prohibitively high complexity of the optimum solution for the assignment problem, we can only compare it with the proposed random greedy assignment problem in small scale networks, where 25 connected and randomly deployed RNs are assigned to 2 BNs whose locations are fixed. The simulation results is shown in Fig. 3 for different value of R/r. It is observed that the difference between the optimum algorithm and the heuristic algorithm (RGA) is less than 5%, and the performance of GA is very close to that of the optimum solution. Therefore, it can be expected that the heuristic algorithm is effective in achieving near optimum performance with much lower complexity.

3.3 Adaptation of BN Positions

After associating the RNs with the k BNs, for a BN b_I, there are $|b_I|$ RNs assigned to it. Without change of association, adapting the position of the BN b_I to its best position $\vec{T}_{(t)I}$ can help reduce the average delay cost, as the average hop number from all RNs associated to b_I

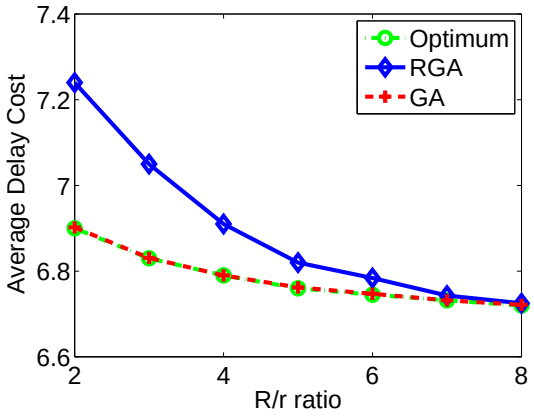


Fig. 3. Performance comparison of assignment algorithms.

and thus the average hop delay can be minimized while the competition delay stays the same. The value of $\vec{T}_{(t)I}$ can be determined as follows.

The BN b_I constructs a sub hop-matrix $H_{(t)I}$ based on the hop matrix H in Eq. (5) by simply obtaining the rows and columns corresponding to the RNs assigned to it in the current round t. The sub hop-matrix thus has the size $|b_I| \times |b_I|$. Recall that the locations of the RNs are denoted as vectors $\vec{L}_1^{RN}, \vec{L}_2^{RN}, ..., \vec{L}_n^{RN}$. As a BN uses an RN position as the reference in each movement, b_I can pick up the target position $\vec{T}_{(t)I}$ as follows:

$$i^* = \arg \min_{i:b(i)=b_I} \left(\frac{1}{|b_I|} \sum_{j:b(j)=b_I} h(i,j) \right), \vec{T}_{(t)I} = \vec{L}_{i^*}^{RN}. \tag{11}$$

Here $h(i,j)$ is the (i,j)-th element of H where RN i and j are both assigned to BN b_I. In other words, it is an element of the sub hop-matrix $H_{(t)I}$. Because H is symmetric, the sub hop-matrix $H_{(t)I}$ is also symmetric. Based on Eq. (11), $\vec{T}_{(t)I}$ corresponds to the position of the RN whose associated row has the minimum summation. Overall, it takes $O(n)$ time to construct the sub hop-matrix and $O(n)$ time to sum up each row. So the finding of new targeted positions for all BNs has a linear running time $O(kn)$ in each round t.

After finding the target position, instead of having a BN directly move to its target position which may lead to a large oscillation and prevent the system from reaching a better deployment option, in our scheme, the BN moves towards its target location gradually. Specifically, if the BN has the current position $\vec{L}_{(t)I}$ and the target position $\vec{T}_{(t)I}$, it moves with a step length l proportional to the vectorial difference between $\vec{L}_{(t)I}$ and $\vec{T}_{(t)I}$ towards an intermediate location $\vec{L}'_{(t)I}$:

$$\vec{L}'_{(t)I} = \vec{L}_{(t)I} + l \cdot (\vec{T}_{(t)I} - \vec{L}_{(t)I}). \tag{12}$$

To deploy the BN close to an RN, the position of the RN closest to $\vec{L}'_{(t)I}$ should be found, denoted as $\vec{L}_{(t+1)I}$,

which could be the virtual position of the BN in the following iteration:

$$i^* = \arg\min_{i \in N_R} |\vec{L}_i^{RN} - \vec{L}'_{(t)I}|, \vec{L}_{(t+1)I} = \vec{L}_{i^*}^{RN}. \quad (13)$$

Therefore, the BN will virtually move from $\vec{L}_{(t)I}$ to $\vec{L}_{(t+1)I}$ at the end of iteration t, and stay at $\vec{L}_{(t+1)I}$ in iteration $t+1$. This can be illustrated using the example in Fig. 4, where the position 1 is the current b_I position $\vec{L}_{(t)I}$. The RNs between "N Network Part A" and "N Network Part B" are assigned to b_I after one round of the RGA. The position 2 is the target $\vec{T}_{(t)I}$, and the position 3 is $\vec{L}'_{(t)I}$. As the initial BN movement is in the granularity of RN hop, the proportion should be bigger than 0.5 to make sure a BN could move to its target position in case sometimes $\vec{T}_{(t)I}$ is only one hop away from $\vec{L}_{(t)I}$. The position 4 of the RN node closest to 3 is finally taken as the new position $\vec{L}_{(t+1)I}$ of b_I, to ensure that the average access delay cost from RNs to the BN network is smaller.

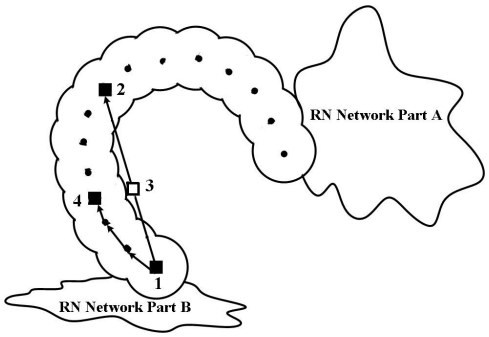


Fig. 4. Move toward the BN Target

3.4 Checking the BN Network Connectivity

In order for the backbone network to be functional and support more efficient and long range transmission for RNs, the BN network needs to be connected. So the adaption of backbone node positions is under the constraint of the backbone network connectivity. There are two common methods to check whether the BN network is connected, one is with the construction of a k-node spanning tree and the other is with the construction of an adjacent matrix. In this work, we take the first method as it has a lower computational complexity. The construction can root from an arbitrary BN and attempt to traverse all the k BNs. If the constructed spanning tree has a size k, the BN network is connected; otherwise, the step length of adapting the BN position needs to be adjusted or the adaptation cannot be performed.

As the determination of the BN deployment is a virtual process, the connection between two neighboring nodes needs to be calculated based on the link model in Eq. (1) with a safety threshold to ensure the connectivity under some fading conditions. After the physical deployment, the position of a BN node can be adjusted based on the strength of the signals received from the neighboring nodes.

3.5 The Complete Algorithm

The complete algorithm is an iterative process consisting of random greedy assignment, adaptation of BN positions, and connectivity checking as shown in Algorithm 2.

On lines 1 to 4, the positions of k BNs are initialized, and Random Greedy Assignment is performed on line 5 for initial RN association. As long as the average delay cost $\bar{c}$ is reduced with the current BN assignment, the iterations proceed and the updated positions of the BNs are then determined as on lines 8 to 12. The adaption of the BN positions is based on the control law. The check of the network connectivity is performed over the updated positions of BNs on line 13. If the BN network remains connected, the BN positions are virtually updated and the algorithm proceeds to the next round. When the connectivity cannot be maintained, the positions of BNs are adjusted to where they can still keep the network connected and an adjustment process is performed on line 21 for a BN to adapt its position under the connection constraint until the delay cost can no longer be improved.

The idea of the self-adjustment process is to let the BNs probe the neighboring areas and find their optimal positions without any change to the current RN assignment. In this way, the value of $\frac{(1-\alpha)}{n}\sum_{i=1}^{n}|b(i)|$ in Eq. (3) remains the same while the value of $\frac{\alpha}{n}\sum_{i=1}^{n}h(i)$ can be optimized. Each BN b_I probes its neighboring RNs, denoted as RN_I, by tentatively moving to each of them and then calculates the corresponding objective function value $\sum_{i:b(i)=b_I}h(i)$. It then can find which position among RN_I could obtain the minimum total hop delay for RNs that are assigned to b_I, i.e. $j^* = \arg\min_{j \in RN_I}\sum_{i:b(i)=b_I}h(i)$. The BN b_I is thus adjusted to the position of RN j^* as it is the position that can improve the objective function most. As each BN probe $O(n)$ RNs, so k BNs probe $O(kn)$ times in total. As k is a constant, the running time of probing is linear to the network size. There is a tradeoff between the number of BN adaptation steps which depends on the step length parameter l in the equation 12 and the total cost of the deployed backbone network. If l is set to a larger value, it would lead to the faster adaptation of BN positions towards its target one and an overall faster speed of finding the virtual BN deployment positions, but a larger adaptation step may skip some candidate BN positions in between thus making the total backbone cost higher. The self-adjustment process is used to fine tune the position of BN to find a better solution. Normally RN_I includes the positions of the RN nodes that are connected to BN b_I in one-hop, but the adaptation process can be carried successively if necessary when l is set to a very large value. For example, when a BN moves to the position of a selected neighboring RN

node i, it can further probe the neighbors of i to find a better solution. Therefore, the reduction of the number of adaptation steps is at the cost of number of adjustment steps needed to find a good solution. The tradeoff has been confirmed in our performance evaluation when we study the impact of l.

Our algorithm stops when the average delay cost cannot be reduced any more or the BN network is disconnected. Suppose that the maximum value of delay is $\bar{c}_{max}$ and the minimum is $\bar{c}_{min}$, both are finite values depending on the network topology. As the algorithm decreases the average delay cost in each iteration, and the decreasing step is no less than $\min\{\alpha, 1-\alpha\}/n$ according to Eq. (3), the convergence step of the algorithm is bounded by $n\,(\bar{c}_{max} - \bar{c}_{min})/\min\{\alpha, 1-\alpha\}$ in the worst case. The actual steps of convergence are studied by simulations in Section 5.

Algorithm 2 Iterative and Adaptive Backbone Deployment

Input: n RNs, k BNs
Calculate $\vec{L}_{mass}$, $i^* = \arg\min_{i \in RN} |\vec{L}_i^{RN} - \vec{L}_{mass}|$
Let $\vec{L}_{(0)1} = \vec{L}_{i^*}^{RN}$ be the root, get $\vec{L}_{(0)2}, ..., \vec{L}_{(0)k}$ by BFS over the RN network
$t = 0$; $\bar{c}_{(0)} = MAX\_VALUE$; $\vec{T}_{(0)I} = NULL$, $\forall I = 1, ..., k$
Run *Random Greedy Assignment*, and get $b(i), \forall i = 1, ..., n$
Calculate $\bar{c}_{(1)}$ based on the assignment
while $\bar{c}_{(t+1)} < \bar{c}_{(t)}$ **do**
  for $b_I = b_1$ to b_k **do**
    $i^* = \arg\min_{i:b(i)=b_I} \left(\frac{1}{|b_I|} \sum_{j:b(j)=b_I} h(i,j)\right)$, $\vec{T}_{(t)I} = \vec{L}_{i^*}^{RN}$
    $\vec{L}'_{(t)I} = \vec{L}_{(t)I} + l \cdot (\vec{T}_{(t)I} - \vec{L}_{(t)I})$
    $i^* = \arg\min_i |\vec{L}_i^{RN} - \vec{L}'_{(t)I}|$, $\vec{L}_{(t+1)I} = \vec{L}_{i^*}^{RN}$
  end for
  Constructing a spanning tree SPT_BN over the backbone network with BNs tentatively placed at $\vec{L}_{(t+1)I}, \forall I = 1, ..., k$.
  if Size_of (SPT_BN) == k **then**
    Move BN b_I virtually to $\vec{L}_{(t+1)I}, \forall I = 1, ..., k$
    $t = t + 1$
    Run *Random Greedy Assignment*, and get $b(i), \forall i = 1, ..., n$
    Calculate $\bar{c}_{(t+1)}$ based on the assignment
  else
    Keep BN b_I at the current location $\vec{L}_{(t)I}$, $\forall I = 1, ..., k$
    Adjusting BN positions to reduce the cost while keeping the network connected
    break
  end if
end while
Return the current values of $\vec{L}_{(t)I}, \forall I = 1, ..., k$

4 Genetic Algorithm for Performance Bound

Genetic Algorithm(GA) [13] has been shown to be a good solution in finding a global optimal solution. In this section, we present the application of GA in seek of the solution that can achieve the optimum, i.e., the performance bound of our algorithm.

Genetic Algorithm is a stochastic optimization algorithm based on the mechanisms of natural selection and natural genetic operation. It starts with a fixed-size population of solutions. Each solution consists of a string of numbers, alphabets or other types of variables, typically binary numbers. The solutions in GA evolve generation by generation. For each generation, GA decides which solution can stay in the next generation based on the probability generated according to a solution's fitness function which is related to the objective function of the optimization problem. After this natural selection process, once we have the next generation's population, GA applies the genetic operators such as mutation or crossover to these solutions and therefore produces the new solution for the next round of natural selection.

As discussed earlier, since $R > r$ and also the objective of the deployment is to minimize the average backbone access cost, the BNs are deployed within the communication ranges of one or a set of RNs. The total possible combinations of the deployment is $O(n^k)$, which can be achieved in polynomial time. We will find all the possible BN deployment combinations first and then use GA to search in the possible assignment combinations to look for the optimum solution. In the following we introduce our design in applying genetic algorithm for achieving the optimal assignment for a given deployment option, which consists of several steps.

4.1 Coding

Each solution s_i corresponds with a string of integer numbers with string length n, which represents one of the assignment results of n RNs. Each RN could be assigned to one of the k BNs from b_1 through b_k. For simplicity, we use $1, 2, ..., k$ to represent the BN that an RN is assigned to. For each s_i, we can find out the hop number from each RN to a BN (located within the transmission range of an RN) by looking up Eq. (5) and the number of RNs assigned to each BN respectively. As mentioned earlier, there are S solutions in each generation, where S is a fixed size.

4.2 Initialization

To start the GA, we need to set up an initial population of solutions. Normally, an initial solutions is generated randomly, but in this case, total randomness may cause no RN assignment to one or more BNs. To avoid this problem, we first randomly pick k RNs and assign them to BN from b_1 to b_k individually, and then assign the remaining $(n-k)$ RNs to the set of BNs randomly.

4.3 Selection

GA selects the next generation with population size S from the previous solutions, and pick one each time with the probability related to the fitness functions of the solutions in the previous generation. For example, the solution s_i is picked with the probability of $P(s_i)$s

$$P(s_i) = \frac{F(s_i)}{\sum_{i=1}^{S} F(s_i)}, \tag{14}$$

where $F(s_i)$ is the fitness function of solution s_i. We set $F(s_i)$ by applying the σ-truncation method [14] to the average delay cost as a result of the assignment s_i. Denoting the average delay cost value with solution s_i as $\bar{c}(s_i)$. Based on the σ-truncation method, we have $c_{ne}(s_i) = -\bar{c}(s_i)$ and $g(s_i) = c_{ne}(s_i) - (\bar{c}_{ne} - c \cdot \sigma)$, where $\bar{c}_{ne}$ and σ are the mean and standard deviation of $c_{ne}(s_i)$ in the current population respectively. The parameter c is a constant between 1 and 3 [14]. Thus, the fitness function of s_i is given by

$$F(s_i) = \begin{cases} g(s_i), & \text{if } g(s_i) \geq 0; \\ 0, & \text{otherwise.} \end{cases} \tag{15}$$

4.4 Crossover

In genetic algorithms, crossover is an operation to vary the programming of a chromosome or chromosomes from one generation to the next. Since we consider the deployment of an enforced backbone network based on the existence of an RN network, the objective function is not dynamic [15]. We thus use simple random crossover operation, where a single crossover point is randomly selected on both parents' solution sequences. All data beyond that point in either sequence is swapped between the two parent solutions. The resulting solutions are the children.

4.5 Mutation

For every generation, we divide the whole generation into two halves. We apply the *directional mutation* on the first half in order to speed up the convergence of the good solutions to their nearby local optimal value. At the same time, we apply the *random mutation* on the second half solutions to have some solutions get out of the bad local optimal point.

4.5.1 Directional Mutation

For a solution s_i, the mutation of the jth item happens if the assignment for the jth RN changes from its current BN to another BN b_p. The mutation probability is given by

$$Prob(s_i(j) = p) = \frac{\lambda c_{max} - \bar{c}(s_i(j) = p)}{\sum_{q=1}^{k} (\lambda c_{max} - \bar{c}(s_i(j) = q))}. \tag{16}$$

Each solution s_i is an assignment combination with a cost $\bar{c}(s_i)$, and $c_{max} = \max_j \bar{c}(s_i(j))$ is the maximum delay cost factor and λ is a constant larger than 1. In Eq. (16), each RN has a higher probability to take the BN with the lower delay cost factor. This selection is greedy in probability and can conduct a local optimal solution.

4.5.2 Random Mutation

To avoid our solution being trapped to an unfavorable local optimal position, we employ the random mutation to let the solution move away from the local optimal value. For a solution s_i, each item has a probability $Prob_{rm}$ to change from its current BN assignment, and probability $1 - Prob_{rm}$ to stay with the current BN. Therefore, if some RN of s_i is assigned to BN b_I, then it will have the probability $\frac{Prob_{rm}}{k-1}$ to be assigned with $k-1$ other BNs except the current b_I. Normally the $Prob_{rm}$ is very low.

4.6 The Complete GA

The complete steps of GA is described in Algorithm 3. GA has several ways of termination. We set a limit to the generation number G_t, beyond which the algorithm will stop and report the best solution it has ever found.

Algorithm 3 Genetic Algorithm

```
while untested BN deployment combination exists do
  Pick an untested deployment combination
  if This BN network is connected and is not an obviously bad option then
    Set up an initial population P_0 with S solutions
    i = 1
    for i = 1 to G_t do
      P'_i = Selection(P_{i-1})
      Update the best c̄(s_i)
      Split P'_i into P'_{i1} and P'_{i2}
      P_{i1} = DirectionalMutation(P'_{i1})
      P_{i2} = RandomMutation(P'_{i2})
      P_i = P_{i1} ∪ P_{i2}
      i = i + 1
    end for
  end if
end while
Select the best c̄(s_i) among O(n^k) group of results
```

In summary, we first find all the deployment combinations within polynomial running time $O(n^k)$. We can get rid of a number of deployment combinations, as they either don't satisfy the connectivity constraint or are obviously bad options (e.g, more than one BN are positioned at the same place, or all BNs are at the boundary of the network).

For each possible deployment combination, we use GA to look for an optimum. After GA running through every deployment combination, we select the best one as the overall optimum, which is used as our simulation bound to evaluate our ITA's performance.

5 Performance Evaluation

In this section, we use simulations to evaluate the performance of our proposed algorithms. 100 RNs are generated one by one in random locations in a $350m \times 350m$ area. Each new RN is ensured to get connected with those RNs that are already distributed. Thus these 100 RNs form a connected network with random topology. We set the RN transmission range $r = 30m$, with the default number of BNs $k = 5$ and default BN transmission range $R = 6r$. The trade-off coefficient α in Eq. (2) is set to be 0.5 and the moving proportion l in Eq. (12) is 0.55. The value of G_t defined in Section 4.6 is 150 in our implementation, which is large enough to obtain the near optimal solution. A simulation result is obtained by averaging over several runs of simulations with different random seeds.

According to the default parameter setup, an RN network with random topology is formed as in Fig. 5, where a Reinforced Backbone Network deployed using the ITA algorithm is also shown.

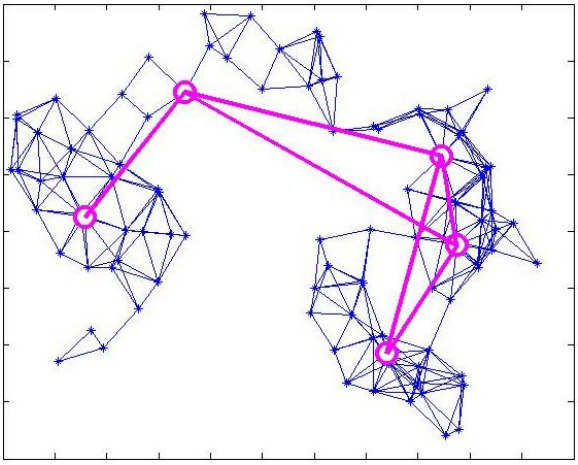


Fig. 5. RN network and its RBN deployment.

5.1 Impact of k

The impact of the number of BNs k is shown in Fig. 6, where k varies from 5 to 10 while other parameters keep the default values. ITA has the close performance to the optimum performance bound provided by GA for all the tested values of k. To demonstrate the effectiveness of the adjustment process of ITA executed in line 21 of Algorithm 2, we also implement a reference scheme without the adjustment process. As expected, the adjustment process of ITA can effectively reduce the average delay cost when k is small. This is due to the fact that with fewer BNs the iterations in Algorithm 2 are prone to terminate when the BN network gets disconnected. When the value of k is larger, the adjustment does not affect the performance much. With more BNs, the algorithm usually guarantees the iteration and adaption loop stops when the average access delay cost cannot be improved without conflicting with the connectivity constraint. In this case, the adjustment process is not needed. It can also be observed that the average delay cost of both ITA and the performance bound decrease with increasing k, as with more BNs, the average BN association size is reduced and thus each RN can have a lower competition delay cost.

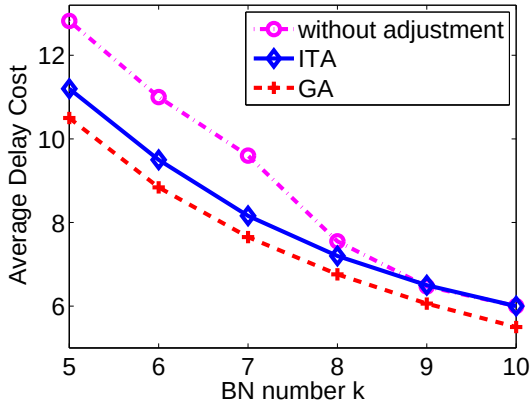


Fig. 6. Impact of k.

5.2 Impact of R/r

The transmission range ratio R/r indicates the transmission range difference between BNs and RNs. The impact of R/r is presented in Fig. 7 with R/r varying from 3 to 8. When R/r is small, the adjustment process is shown to be very effective in reducing the average delay cost compared with the reference scheme which does not perform adjustment. When the BNs have a relatively short transmission range R, Algorithm 2 are more possible to jump out of the iterations when the BN network becomes disconnected, therefore the adjustment process is crucial to improve the performance in this situation. Alternatively, when R is relatively longer, e.g. $R/r \geq 7$, the connectivity of BN network can be guaranteed irrespective of where the BNs are deployed, and the iterations in Algorithm 2 generally terminate

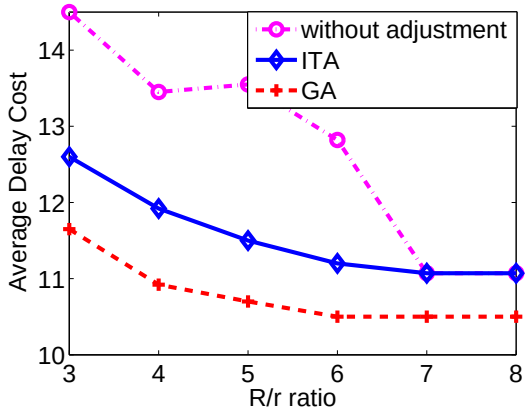


Fig. 7. impact of R/r.

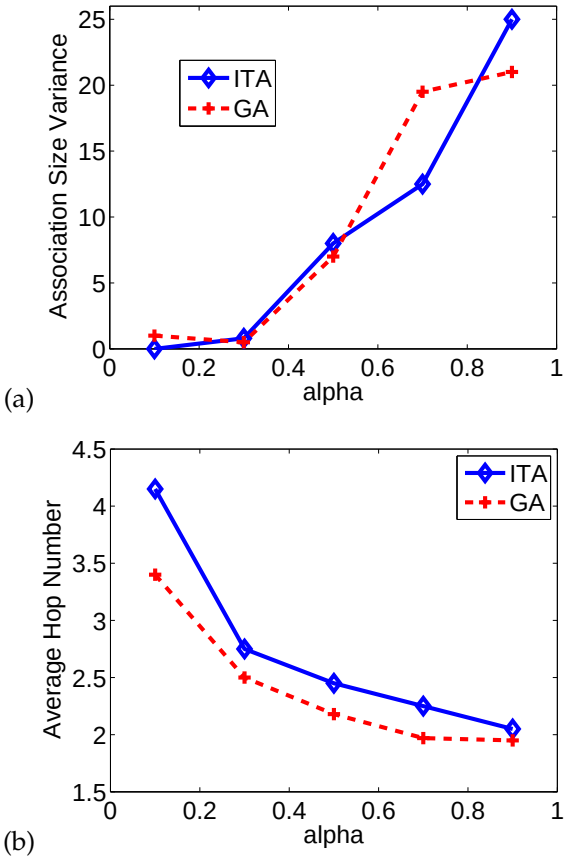


Fig. 8. Impact of α: (a) Association size variance; (b) Average hop number.

only when the average access delay cost factor cannot be further improved. As a result, ITA and the reference scheme without adjustment are observed to have very close performance in this case. Another interesting observation is that there is little change in average delay costs of ITA and GA when R increases further, i.e. $R/r > 6$. As the RN network area is fixed, a further increase of R does not change the network topology and thus the average delay cost stays the same. This observation provides a reference for setting the transmission range of BNs in a practical deployment.

5.3 Impact of α

In Section 2.1, a parameter α is introduced to adjust the trade-off between the hop delay and the competition delay in calculating the delay cost. In Fig. 8, the impact of α is studied when α changes from 0.1 to 0.9 and other parameters are fixed. A smaller value of α emphasizes more on achieving the balance of association sizes to relieve the impact of hot spots. On one hand, as a result of controlling the load associated with each BN, the variance of association sizes is very small in Fig. 8 (a) with smaller values of α, e.g. $\alpha \leq 0.3$. On the other hand, the average hop number is relatively larger in Fig. 8 (b) for smaller α as RNs are possibly assigned to BNs far away in this case. On the contrary, a larger value of α relaxes the control of load balancing among BNs, and instead gives the freedom for each RN to select the closest BN greedily in order to reduce the hop delay. Therefore, the average hop number in Fig. 8 (b) reduces significantly when α increases at the cost of a higher association variance as in Fig. 8 (a).

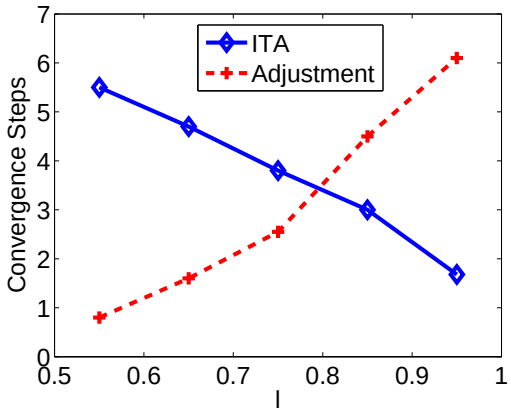


Fig. 9. Impact of l

5.4 Impact of l

Varying from 0.55 to 0.95, the moving proportion l defined in Section 3.3 introduces a trade-off between the number of iteration steps and the number of adjustment steps as shown in Fig. 9. With a lower l, the BNs move towards their target positions for a smaller distance in each iteration, thus it requires more iteration steps to move to their target positions. When l is higher, although a BN could move to its target position in much fewer iterations, it could probably pass over those positions that may provide a lower access delay due to the higher moving granularity. As a result, after all the iterations, the algorithm needs more steps to perform the adjustment, reconsidering those positions it has ignored.

The results in Fig. 9 also demonstrate that the iterative algorithm described in Algorithm 2 converges quite fast in actual network settings, i.e. the number of total convergence steps is less than 10, and each convergence step takes only $O(kn)$ time to complete. As a result, ITA is much more time efficient than GA, as the former takes less than 1 hour to complete while the latter takes about 100 hours to obtain the solution for the default network setting even though many bad positions have been removed manually which has significantly reduced the GA time.

5.5 Impact of $InitialPosition$

In Section 3.1, our proposed algorithm finds the initial positions around the mass center of the RNs through the BFS traversal of the BN network. An alternative and conventional way of initial position setup is to randomly pick up the initial positions, where an RN position is

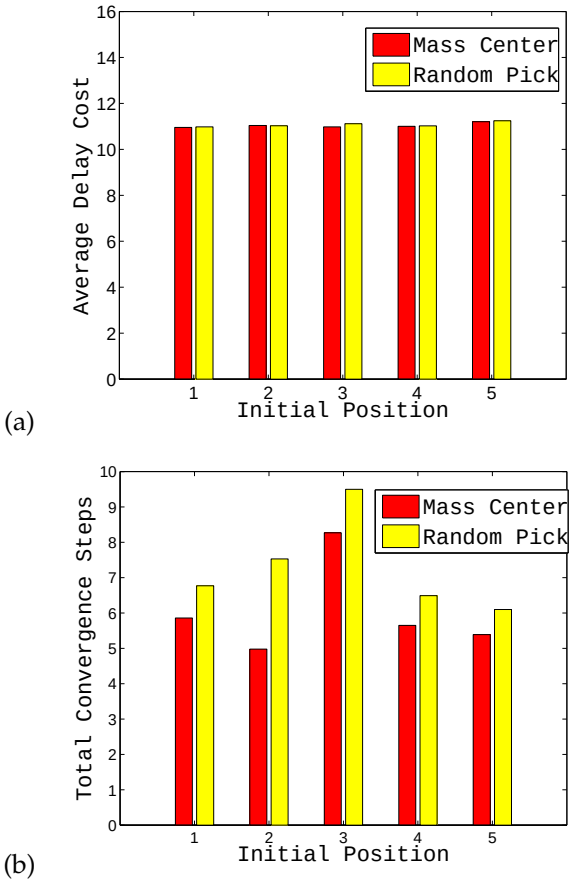


Fig. 10. Impact of the initial BN positions (a) Average delay cost; (b) Total number of convergence steps.

randomly picked first and a BFS traversal of the RN network is then performed around this selected RN to find the other $k-1$ BN initial positions so as to guarantee the connectivity of the initial BN network. In Fig. 10, we study the impact of the initial positions by comparing the two ways of initial position setup. The results are obtained by averaging over five different random RN networks and each way of initial positions setup is run for 50 simulations in each RN network. As illustrated in Fig. 10 (a), the average delay costs for the two ways of initial positions selection are close, indicating that the initial deployment through BFS is quite robust with different initial positions. The initialization in reference to the mass center, however, leads to a faster convergence speed of the backbone network formulation, while the initialization using randomly selected locations generally takes a longer period of time to converge as shown in Fig. 10 (b). The performance results demonstrate that our proposed way of BN position initialization could help expedite the BN network formulation without increasing the average delay cost.

6 Related Work

Many efforts have been made in recent years to construct a backbone network to carry the total network traffic by selecting a minimum set of backbone nodes out of the total network nodes, in order to reduce the total network transmissions and hence collisions for improving the network throughput [16]–[23]. These studies normally assume the network nodes have the same transmission ranges, and the backbone nodes are selected from existing nodes. Although backbone nodes with higher transmission ranges were considered in [24]–[27], these studies also assume the RNs and MBNs are already placed, and a-priori form a connected network. Xu et al. [25] simply selects the nodes that first claim the leadership in a neighborhood to be clusterheads, while TBONE proposed in [24] attempts to minimize the number of backbone nodes, giving priority to higher weight nodes. The focus of these efforts relates to developing system-level protocols for routing, scheduling, MBN election, etc. In [26], [27], the authors exploited network spectrum domain characteristics and designed a novel cost metric to simultaneously increase multiple types network performance.

Instead of selecting backbone nodes, more recently, the backbone network deployment problems are studied in [3], [5]. In [3], the authors formulate the Connected Disk Cover (CDC) problem, which aims to place the minimum number of mobile backbone nodes (MBNs) such that all RNs are covered by at least one MBN, and the placed MBNs have the same coverage distance and form a connected network. In many practical scenarios, however, there is only a fixed number of backbone nodes that can be deployed, and the deployment can be only performed under the constraint of the available backbone resources. Although the authors in [5] also perceived the issues and attempted to deploy a limited number of backbone nodes, they failed to consider an important constraint, i.e., backbone network connection. In addition, the paper is impractically based on an implicit assumption that a regular node can reach any backbone nodes directly.

Besides work on mobile backbone networks, other closely related work includes base station selection/placement for cellular and indoor wireless systems, e.g. [28], [29]. However, practical considerations for cellular base station placement usually restricts the set of possible locations to a discrete set of candidates and trivial solutions are used for the assignment problem (e.g. assign each RN to the nearest MBN). Cluster organization has been widely studied in the literature [30], [31] and is generally performed in two steps, selecting cluster heads among nodes based on some criteria and forming clusters by associating each cluster head with a set of members. The work in [4] assumes that there are an unlimited number of cluster heads, and the goal is to minimize the total number of cluster heads in the deployment. However, the algorithms for cluster organization cannot be directly applied to our backbone deployment problem, where we consider backbone nodes have a longer transmission range which is different from the cluster heads. The placement of Internet transit access points is studied in [32] to provide

Internet connectivity in multi-hop wireless networks, and the gateway placement for throughput optimization in multi-hop wireless mesh networks is addressed in [33] and [34]. Besides the difference in network types and thus constraints on the deployment, these studies do not consider the cost of access delay and the gateways that can access Internet are assumed to be connected.

Our work distinguishes itself from the aforementioned work in that it studies the optimal deployment of backbone network with use of the limited number of backbone nodes and ensuring backbone connectivity. The initial results have been presented in [35]. In this paper, we present more details of our design as well as a more complete overview of related work.

7 Conclusions

In this paper we propose algorithms for the deployment of a Reinforced Backbone Network to improve the communication performance of a meshed wireless network. The objective of the deployment is to minimize the average backbone access delay cost from regular mesh network nodes which have lower capabilities. We formulate the problem and discuss its complexity. Inspired by the theories in data mining and robotics fields, we propose an iterative and adaptive algorithm (ITA) which can construct a robust Reinforced Backbone Network insensitive to the initial deployment positions. Moreover, we exploit genetic algorithm to obtain the lower cost bound of the problem. We have performed extensive simulations to study the impact of different parameters on the performance of the proposed ITA algorithm and compare the results with that obtained through the genetic algorithm. The results indicate that the ITA algorithm can quickly converge and achieve the performance close to that obtained through the genetic algorithm, which requires several days of running. Our study indicates that the proposed ITA algorithm is promising for the deployment of a connected Reinforced Backbone Network with a limited number of available backbone nodes.

For future work, we will extend the algorithm to work in a mobile environment and develop a closed-form mathematical performance bound.

References

[1] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Transactions on Information Theory*, vol. 46(2), pp. 388–404, Mar. 2000.

[2] R. M. de Moraes, H. R. Sadjadpour, and J. J. Garcia-Luna-Aceves, "On mobility-capacity-delay trade-off in wireless ad hoc networks," in *Proceeding of IEEE Computer Society's 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'04)*, Oct. 2004, pp. 12–19.

[3] A. Srinivas, G. Zussman, and E. Modiano, "Mobile backbone networks: Construction and maintenance," in *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*, 2006, pp. 166–177.

[4] S. Pandey, P. A. S. Dong, and K. M. Sivalingam, "On performance of node placement approaches for hierarchical heterogeneous sensor networks," *Journal of Mobile Networks and Applications*, vol. 14(4), pp. 401–414, Oct. 2008.

[5] A. Srinivas and E. Modiano, "Joint node placement and assignment for throughput optimization in mobile backbone networks," in *Proceeding of IEEE INFOCOM - The 27th Conference on Computer Communications*, 2008, pp. 1130 – 1138.

[6] B. Liu, P. Thiran, and D. Towsley, "Capacity of a wireless ad hoc network with infrastructure," in *Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*, 2007, pp. 239–246.

[7] P. Zhang, "A new approximation algorithm for the k-facility location problem," *Theoretical Computer Science*, vol. 384(1), pp. 126–135, Sep. 2007.

[8] D. Hochbaum and D. Shmoys, "A best possible heuristic for the k-center problem," *Mathematics of Operations Research*, vol. 10(2), pp. 180–184, 1985.

[9] D. Turnbull and C. Elkan, "Fast recognition of musical genres using RBF networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17(4), pp. 580–584, Apr. 2005.

[10] A. Howard, M. J. Matari'c, and G. S. Sukhatme, "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem," in *Proceedings of the 6th International Symposium on Distributed Autonomous Robotics Systems (DARS02)*, 2002.

[11] A. Tamir, "A strongly polynomial algorithm for minimum convex separable quadratic cost flow problems on two-terminal series-parallel networks," *Mathematical Programming: Series A and B*, vol. 59(1), pp. 117 – 132, Mar. 1993.

[12] R. K. Ahuja, T. L. Magnanti, and J. Orlin, *"Network Flows:Theory, Algorithms, and Applications"*. Prentice Hall, 1993.

[13] J. H. Holland, *"Adaptation in natural and artificial systems"*. MI: Univ. of Michigan Press, 1975.

[14] D. E. Goldberg, *"Genetic Algorithms in Search, Optimization, and Machine Learning"*. Addison-Wesley Professional, 1989.

[15] W. Rand, R. Riolo, and J. H. Holland, "The effect of crossover on the behavior of the GA in dynamic environments: a case study using the shaky ladder hyperplane-defined functions," in *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, Jul. 2006, pp. 1289–1296.

[16] K. M. Alzoubi, P. J. Wan, and O. Frieder, "New distributed algorithm for connected dominating set in wireless ad hoc networks," in *Proceeding of IEEE Hawaii International Conference on System Science*, 2002, pp. 3849–3855.

[17] J. Wu and H. Li, "On calculating connected dominating set for efficient routing in ad hoc wireless networks," in *Proceeding of Third International Workshop on Discrete Algorithm and Methods for Mobile Computation and Communications*, 1999, pp. 7–14.

[18] F. Dai and J. Wu, "An extended localized algorithm for connected dominating set formation in ad hoc wireless networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, no. 10, pp. 908–920, oct 2004.

[19] Y. Wang, W. Wang, and X. Li, "Efficient distributed low-cost backbone formation for wireless networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, no. 7, pp. 681–693, Jul 2006.

[20] S. Basagni, "Finding a maximal weighted independent set in wireless networks," *Telecommunications Systems*, vol. 18, no. 1-3, pp. 155–168, Sep 2001.

[21] C. Bettstetter and R. Krausser, "Scenario-based stability analysis of the distributed mobility-adaptive clustering (DMAC) algorithm," in *Proceeding of the Second ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2001, pp. 232–241.

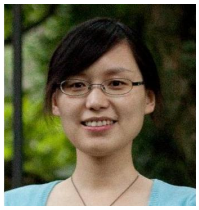
[22] U. C. Kozat, G. Kondylis, B. Ryu, and M. K. Marina, "Virtual dynamic backbone for mobile ad hoc networks," in *IEEE International Conference on Communications (ICC)*, 2001, pp. 250–255.

[23] M. Min, F. Wang, D.-Z. Du, and P. Pardalos, "A reliable virtual back- bone scheme in mobile ad hoc networks," in *IEEE International Conference on Mobile Ad-Hoc and Sensor Systems*, 2004, pp. 60–69.

[24] I. Rubin, A. Behzad, R. Zhang, H. Luo, and E. Caballero, "TBONE: a mobile-backbone protocol for ad hoc wireless networks," in *Proceeding of IEEE Aerospace Conference*, 2002, pp. 271–282.

[25] K. Xu, X. Hong, and M. Gerla, "Landmark routing in ad hoc networks with mobile backbones," *Journal of Parallel and Distributed Computing*, vol. 63, no. 2, pp. 110–122, 2003.

[26] Z. Zhang, X. Wang, and Q. Xin, "A new performance metric for construction of robust and efficient wireless backbone network," *IEEE Transactions on Computers*, 2011.
[27] Z. Zhang, Q. Ma, and X. Wang, "Exploiting use of a new performance metric for construction of robust and efficient wireless backbone networks," in *ACM/IEEE International Workshop on Quality of Service*, 2010, pp. 1–9.
[28] S. Hanly, "An algorithm for combined cell-site selection and power control to maximize cellular spread spectrum capacity," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 4, pp. 1332–1340, 1995.
[29] D. Stamatelos and A. Ephremides, "Spectral efficiency and optimal base placement for indoor wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 4, pp. 110–122, 1996.
[30] C. Lin and M. Gerla, "Adaptive clustering for mobile wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 7, pp. 1265–1275, 1997.
[31] H. Ju and I. Rubin, "Backbone topology synthesis for multi-radio meshed wireless LANs," in *Proceeding of IEEE INFOCOM - The 25th Conference on Computer Communications*, Apr. 2006, pp. 1–12.
[32] R. Chandra, L. Qiu, K. Jain, and M. Mahdian, "Optimizing the placement of internet taps in wireless neighborhood networks," in *Proceeding of the 12th IEEE International Conference on Network Protocols (ICNP)*, 2004, pp. 271–282.
[33] F. Li, Y. Wang, X.-Y. Li, A. Nusairat, and Y. Wu, "Gateway placement for throughput optimization in wireless mesh networks," *Mobile Networks and Applications*, vol. 13, no. 1-2, pp. 198–211, 2008.
[34] B. Aoun, R. Boutaba, Y. Iraqi, and G. Kenward, "Gateway placement optimization in wireless mesh networks with QoS constraints," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 11, pp. 2127–2136, Nov. 2006.
[35] P. Wei, S. Chu, X. Wang, and Y. Zhou, "Deployment of a reinforcement backbone network with constraints of connection and resources," in *Proceeding of IEEE 30th International Conference on Distributed Computing Systems (ICDCS)*, 2010, pp. 10–19.



Shan Chu received her B.S. and M.S. degrees in electrical engineering from Zhejiang University, Hangzhou, China, and the Ph.D. degree in electrical engineering from Stony Brook University, Stony Brook, NY. She is currently with Motorola Solutions Inc., Holtsville, NY. Her current research interests include MIMO and cooperative communications, ad hoc networks and cross-layer design.

Peng Wei received his B.S. degree in automation from Tsinghua University, Beijing, China and M.S. in electrical engineering from Stony Brook University, Stony Brook, NY. He is currently pursuing Ph.D. in the School of Aeronautics and Astronautics Engineering at Purdue University, West Lafayette, IN. His interests include modeling, optimization, algorithms, control theory and their applications in air traffic management, air transportation operation research and airspace design. He also works on multimodel transportation system and other network related research.

Xu Zhong received the B.S. and M.S. degrees in mechanical engineering from China University of Mining and Technology, Beijing, China, in 2005 and 2008, respectively. Currently, he is working towards the Ph.D. degree in mechanical engineering of Stony Brook University, Stony Brook, NY. His research interests include multi-robot systems, artificial intelligent and optimization algorithm.

Xin Wang received the B.S. and M.S. degrees in telecommunications engineering and wireless communications engineering respectively from Beijing University of Posts and Telecommunications, Beijing, China, and the Ph.D. degree in electrical and computer engineering from Columbia University, New York, NY.
She is currently an Associate Professor in the Department of Electrical and Computer Engineering of the State University of New York at Stony Brook, Stony Brook, NY. Before joining Stony Brook, she was a Member of Technical Staff in the area of mobile and wireless networking at Bell Labs Research, Lucent Technologies, New Jersey, and an Assistant Professor in the Department of Computer Science and Engineering of the State University of New York at Buffalo, Buffalo, NY. Her research interests include algorithm and protocol design in wireless networks and communications, mobile and distributed computing, as well as networked sensing and detection.
She has served in executive committee and technical committee of numerous conferences and funding review panels, and is the referee for many technical journals. Dr. Wang achieved the NSF career award in 2005, and ONR challenge award in 2010.

Yu Zhou received his PhD degree in Mechanical Engineering from The Johns Hopkins University, Baltimore, Maryland, USA in 2004. He is currently an assistant professor in Mechanical Engineering at the State University of New York at Stony Brook, New York, USA. His main research area is robotics. His research interests include robot kinematics and dynamics, mobile robot localization and navigation, and multi-robot systems.