Deployment of a Reinforcement Backbone Network with Constraints of Connection and Resources

Peng Wei, Shan Chu, Xin Wang and Yu Zhou
State University of New York at Stony Brook
Stony Brook, NY 11794, USA
Email: pwei@ieee.org, {schu,xwang}@ece.sunysb.edu, yuzhou@notes.cc.sunysb.edu

Abstract

In recent years, we have seen a surge of interest in enabling communications over meshed wireless networks. Particularly, supporting peer-to-peer communications over a multi-hop wireless network has a big 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 Reinforcement 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 Generic Algorithm. Finally, we present our simulation results under various settings and compare the performance of the proposed ITA algorithm and the generic algorithm. Our study indicates that the proposed ITA algorithm is promising for deploying a connected RBN with a limited number of available backbone nodes.

1. Introduction

Supporting peer-to-peer communications over a meshed wireless network has seen big 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 not reliable or have 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 will be 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 more reliable transmissions, lower delay and higher throughout to remote destination nodes, higher quality links.

The backbone network problems have been recently studied in [3]–[5]. The works in [3] [4] assume there are 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 assumed 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 *Reinforcement 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 ascribe wireless nodes into two types. 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.

The objective of our backbone deployment is to minimize 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 the 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 algorithm and gene algorithm. Section 6 concludes this paper.

2. Problem Formulation

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:

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

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 receiving node j.

Definition 1 (Link): A node i can reach a node j if $SINR_{i,j}$ is larger than a threshold γ_j, which depends on the decoding capability of j. For two nodes i and j, if i can reach j and j can reach i, there is a link between i and j.

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.

We assume 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 $1, 2, ..., n$. The link between 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 Reinforcement Backbone Network (RBN) to enable RNs to communicate more efficiently over a long distance. 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 backbone network, and the long-range backbone communications will not interrupt 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 $b_1, b_2, ..., b_k$.

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 a RN needs to communicate with another node that is farther away through a path over only RNs, there will be a large number of hops between the source and the destination, which will not only incur a high transmission delay but also lead to a low end-to-end throughput. Therefore, it is necessary to introduce an Reinforcement 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 backbone network construction time, the actual transmission needs are not known. Also, a BN has a much higher transmission bandwidth and longer transmission range than a 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, we consider it critical to reduce the delay for a RN to access the backbone network.

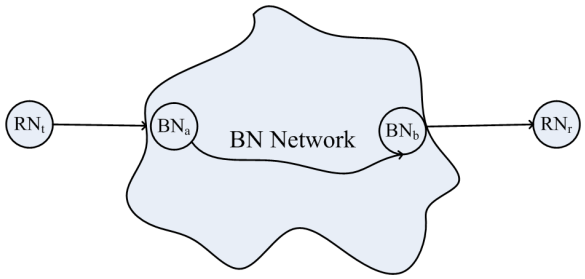

Figure 1. An end to end long range RN transmission.

For each RN i, there will be an assigned BN $b(i)$ for the RN 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 transmission delay, if too many RNs want to route their packets into the backbone network through the same BN, the BN will 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.

Let's denote the regular nodes by the set $N_R = \{1, 2, ..., n\}$ and the backbone nodes 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 backbone node k. Let $h(i, b_J)$ be the hop number from 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 will be k groups of RNs, each RN group is associated with one of the k BNs. In this work, every RN is assigned to exactly one BN, thus $\sum_{I=1}^{k} |b_I| = n$. A 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 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 link model in Eq. (1) with a safety threshold to ensure the connectivity under some fading conditions.

When multiple RNs are associated with one BN, there will be 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 will directly be 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 a 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 we do not intend to model the accurate delay in backbone access, but only consider the factors that impact the backbone access delay.

To evaluate the overall backbone network deployment performance, we will consider the average backbone access delay cost factor $\bar{c}$ of all RNs, and the objective of the backbone network deployment is to minimize $\bar{c}$ which is:

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

2.2. The Problem

Our problem is to deploy k BNs to form a Reinforcement 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:

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

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 Reinforcement Service to RNs, a 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 a RN to the BN network, each BN should be within the transmission range of at least one RN. Each BN can choose which RN transmission area it will stay and totally we have $\binom{n}{k} = n^k$ candidate deployment options. With the BN network connection constraint, only some of these candidate deployment locations are feasible. So there will be less than n^k types of deployment. Because k is known as a constant, the deployment solution has a polynomial complexity. As we assume $R > r$, we can first deploy the BNs to the center of the RN transmission area, i.e., the current positions of the RN. Adjusting the position of BN within the RN's transmission range will not affect the connection constraint significantly.

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

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

where the item $h(i,j)$ is the shortest path hop number between RNs i and j, and $h(i,i)$ is 0. Apparently, H is a natural symmetric matrix. Assuming our BN nodes $\{b_1, b_2, ..., b_k\}$ have been already 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 in Eq. (4) is NP-hard.

3. Iterative and Adaptive Backbone Deployment

As discussed earlier, in order to minimize the average backbone network access delay cost, it will be good for BNs to stay close to the RNs and for each BN to be within the transmission range of one or more RNs. Considering the transmission range 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 possible combinations for BN deployment and all the possible association between n RNs and k BNs which may take 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. In our deployment, the change of BN positions is virtual until a solution is found.

To reduce the number of transmission hops to the backbone nodes and balance the association between 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. We first find the RN position closest to the mass center, $\vec{L}_{(0)1}$, to virtually deploy the first BN. Since the n RNs already form a connected network, to keep the remaining $k-1$ BNs close to the first BN, we perform a Breadth First Search to traverse the RN network to deploy the remaining $k-1$ BNs virtually on the positions of the closest $k-1$ RNs to the root BN denoted by $\vec{L}_{(0)2}, \vec{L}_{(0)3}, ..., \vec{L}_{(0)k}$, where (0) means the round 0 of the iteration of the algorithm, which is the initial stage. As the transmission range of a BN is much larger than an RN, the connectivity of the RN network will ensure the initial deployment of BNs forms a connected network.

3.2. Random Greedy Assignment

Given the locations of the k BNs in Round 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. We propose a Random Greedy Assignment (RGA) approach, in which the n RNs will take turn to associate with a BN in a random order as shown in Algorithm 1.

For each iteration round, at the beginning of the association process, the association results from the last round will be cleared up in Line 2, i.e.,, $\forall b_p \in \{b_1, b_2, ..., b_k\}, |b_p| = 0$. For each RN assignment, the access delay cost from the RN to a BN can be calculated based on Eq. (2), and the RN will associate with the BN that provides the least access cost. After each RN association, the number of RNs associated with the selected backbone node p, i.e., $|b_p|$, is increased by one. 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.

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. With the same association, adapting the position of the BN to the best position $\vec{T}_{(t)I}$ to minimize the average hop number from all RNs will reduce the average delay cost, as the competition cost will stay the same. The $\vec{T}_{(t)I}$ can be determined as follows. The backbone node b_I constructs a sub hop-matrix $H_{(t)I}$ based

Algorithm 1 Random Greedy Assignment

load the list L with RN set $\{1, 2, ..., n\}$
$\forall b_I \in \{b_1, b_2, ..., b_k\}, |b_I| = 0$
while L is not empty **do**
  randomly select an RN i from L
  for $I = 1$ to k **do**
    $b(i) = b_I$
    $c_i = \alpha h(i) + (1 - \alpha)|b(i)|$
  end for
  find the BN with the smallest c_i and assign RN i to it
  $|b(i)| = |b(i)| + 1$
end while

on the hop matrix 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 $|b_I| \times |b_I|$. 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:

$$\vec{T}_{(t)I} = Location\_of\_RN(\arg\min_i (\frac{1}{|b_I|} \sum_{j \in b_I} h_{(t)I}(i,j))), \quad (6)$$

where $h_{(t)I}(i,j)$ is the (i,j)th item of $H_{(t)I}$. Because H is symmetric, the sub hop-matrix $H_{(t)I}$ is also symmetric. Based on Eq. (6), $\vec{T}_{(t)I}$ 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 letting a BN to 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 will move towards its target location gradually. Specifically, if the BN has the current position $\vec{L}_{(t)I}$ and the target position $\vec{T}_{(t)I}$, the BN will move with a step length proportional to the vectorial difference between $\vec{L}_{(t)I}$ and $\vec{T}_{(t)I}$ towards the intermediate location $\vec{L}'_{(t)I}$:

$$\vec{L}'_{(t)I} = \vec{L}_{(t)I} + l \cdot (\vec{T}_{(t)I} - \vec{L}_{(t)I}). \quad (7)$$

To deploy the BN close to an RN, the position of the RN closest to $\vec{L}'_{(t)I}$ will be found, i.e, $\vec{L}_{(t+1)I}$. Therefore, the BN will move from $\vec{L}_{(t)I}$ to $\vec{L}_{(t+1)I}$. This can be illustrated using the example in Fig. 2, where position 1 is the current b_I position $\vec{L}_{(t)I}$. The RNs between "RN Network Part A" and "RN 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 from RNs to the BN network is smaller.

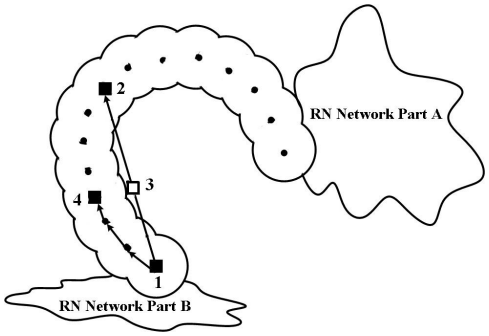

Figure 2. Move toward the BN Target

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 methods to check whether the BN network is connected: with the construction of a k-node spanner tree or with the construction of 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. 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 deployment, the position of a BN node can be adjusted based on the strength of the signals received from the neighboring nodes.

3.5. Complete Algorithm

The complete algorithm will be an iterative process through Random Greedy Assignment, adaptation of BN positions, and checking the connectivity as shown in Algorithm 2.

On lines 1 to 4, the positions of k BNs are initialized, and RGA is performed on line 6 for RN association. On lines 7 to 11, the updated positions of the BNs are determined. If the BN network remains connected, the BN positions are updated and the algorithm moves to the next round; otherwise the positions of BNs are adapted to where they can still keep the network connected.

The adaption of the BN positions is based on the control law. The check of the network connectivity is initially performed over the updated positions of BNs on line 12. When the connectivity cannot be maintained, an adjustment process is used on line 25 for a BN to explore the neighboring area to potentially adapt the its position for a lower cost under the connection constraint until the objective function can no longer be improved.

Our algorithm stops when the cost cannot be reduced any more or the BN network is disconnected. For the second situation, the algorithm uses an adjustment process to control each BN to explore its neighbor area until the objective function can't get improved.

Algorithm 2 Iterative and Adaptive Backbone Deployment

1: n RNs, k BNs
2: calculate $\vec{L}_{mass}$
3: $\vec{L}_{(0)1} = \vec{L}_{mass}$ as the root
4: get $\vec{L}_{(0)2}, ..., \vec{L}_{(0)k}$ by BFS RN network
5: **while** $\exists \vec{L}_{(t)I}$!= $\vec{T}_{(t)I}$ **do**
6: run Random Greedy Assignment
7: **for** $b_I = b_1$ to b_k **do**
8: $\vec{T}_{(t)I} = \arg\min_i \left(\frac{1}{|b_I|} \sum_{j \in b_I} h_{(t)I}(i,j)\right))$
9: $\vec{L}'_{(t)I} = \vec{L}_{(t)I} + l \cdot (\vec{T}_{(t)I} - \vec{L}_{(t)I})$
10: $\vec{L}_{(t+1)I} = closest(\vec{L}'_{(t)I})$
11: **end for**
12: Constructing the spanning tree SPT_BN over the updated backbone network.
13: **if** Size_of (SPT_BN) == k **then**
14: connectivity = 1;
15: **else**
16: connectivity = 0;
17: **end if**
18: **if** connectivity == 0 **then**
19: return to the previous locations $\vec{L}_{(t)I}$, $I = 1, ..., k$
20: break
21: **else**
22: $t = t + 1$
23: **end if**
24: **end while**
25: Adjusting BN positions to reduce the cost while keeping the network connectivity.

4. Gene Algorithm for Performance Bound

Genetic Algorithm(GA) [11] has been shown to be good in finding global optimal solution. In this section, we present the application of GA to look for 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 will be 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 gene 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, and 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 above, in each generation, there will be a fixed-size S solutions.

4.2. initialization

To start the GA, we need to set up an initial population of solutions. Normally, the initial solutions will be 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 $P(s_i)$:

$$P(s_i) = \frac{F(s_i)}{\sum_{i=1}^{p} F(s_i)}. \quad (8)$$

$F(s_i)$ is the fitness function of solution s_i. We set $F(s_i)$ by applying the σ-truncation method [12] 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 [12]. Thus, the fitness function of s_i is given by

$$F(s_i) = \begin{cases} g(s_i), & \text{if } g(s_i) \geq 0; \\ 0, & \text{otherwise.} \end{cases} \quad (9)$$

4.4. crossover

Since we consider the deployment of an enforcement backbone network based on the existence of an RN network, the objective function is not dynamic [13]. We thus decide not to adopt the crossover operation, because they will cause a longer running time and much more generations to converge to the optimum [13].

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

$$Prob(s_i(j) = p) = \frac{\lambda c_{max} - \bar{c}(s_i(j) = p)}{\sum_{q=1}^{k} (\lambda c_{max} - \bar{c}(s_i(j) = q))} \quad (10)$$

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. (10), 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

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

```
1: while untested BN deployment combination exists do
2:    pick an untested deployment combination
3:    if this BN network is not connected or is an obviously bad option then
4:       continue
5:    else
6:       set up an initial population P_0 with S solutions
7:       i = 1
8:       for i = 1 to G_t do
9:          P'_i = Selection(P_{i-1})
10:         update the best c̄(s_i)
11:         split P'_i into P'_{i1} and P'_{i2}
12:         P_{i1} = DirectionalMutation(P'_{i1})
13:         P_{i2} = RandomMutation(P'_{i2})
14:         P_i = P_{i1} ∪ P_{i2}
15:         i = i + 1
16:      end for
17:   end if
18: end while
19: pick 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. 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

We use simulations to evaluate the performance of our proposed algorithms. 100 RNs are generated one by one in random locations. Each new RN is ensured to get connected with those distributed RNs. Thus these 100 RNs form a connected network with random topology. RN transmission range $r = 30m$, with default BN number $k = 5$ and default BN transmission range $R = 6r$, trade-off coefficient α in Eq. (2) is 0.5 and the moving proportion l in Eq. (7) is 0.55. 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. 3. The Reinforcement Backbone Network deployed using the ITA algorithm is also shown in Fig. 3.

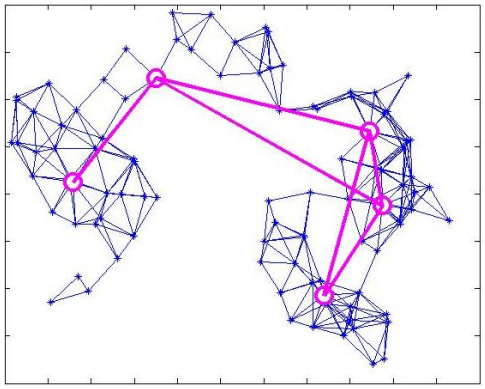

Figure 3. RN network and its RBN deployment.

5.1. Impact of k

The impact of BN number k is shown in Fig. 4. The BN number varies from 5 to 10 while other parameters keep the default values. ITA has the close performance to the simulation optimum performance bound provided by GA. As expected, the adjustment process of ITA executed at the end of algorithm 2 can effectively reduce the average delay cost when k is small because if we have fewer BNs, the algorithm will probably come out of the iteration when the BN network gets disconnected. When k is bigger, 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 connection constraint. So there will be no adjustment needed. However, when k increases, both the average delay costs of ITA and performance bound decrease because if we have more BNs, the average BN association size will be reduced thus each RN inside its BN association will have less competition delay cost.

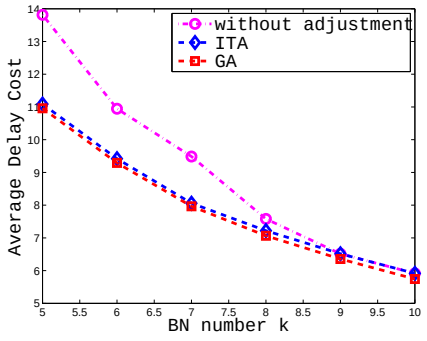

Figure 4. Impact of k.

5.2. Impact of R/r

The impact of transmission range ratio R/r is shown in Fig. 5 with R/r varying from 3 to 8. The adjustment process is shown to be very effective in reducing the average delay cost when R/r is small because if BN transmission range R is small, the algorithm will jump out of the iterations when the BN network becomes disconnected, therefore the adjustment is crucial to improve the performance in this situation. When R is bigger, ITA and ITA without adjustment have the same performance because when R has a long range that the BN network can keep connected almost everywhere it is deployed, the algorithm usually guarantees the iteration is stopped when the average access delay cost factor cannot be improved without conflicting with the connection constraint. When R is increased further (when $R/r > 6$), however, the

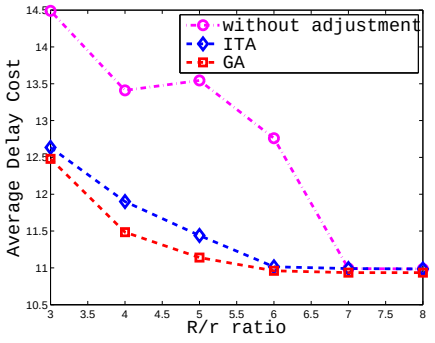

Figure 5. impact of R/r.

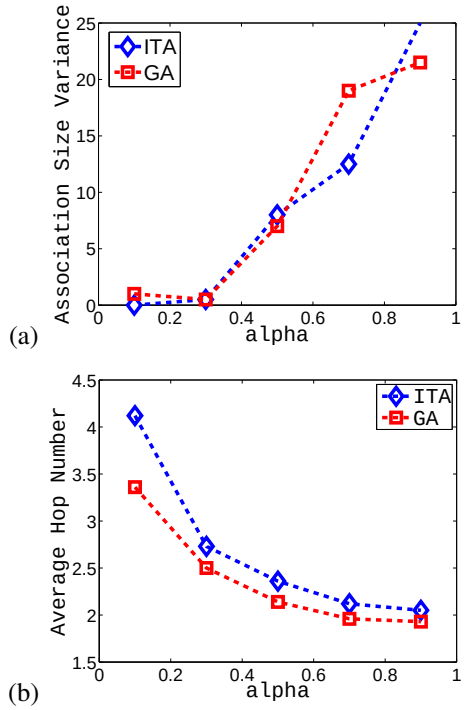

Figure 6. Impact of α: (a) Association size variance; (b) Average hop number.

average delay costs of ITA and GA will keep the same value. Because the RN network area is fixed, no matter how much more R increases, the RN network area and topology don't change, thus the average delay cost will stay the same.

5.3. Impact of α

With other parameters fixed, α changes from 0.1 to 0.9. The smaller α puts more emphasis on achieving the balance of association sizes to get rid of hot spots. As a result of controlling the balance of the load associated with each BN, the variance of association sizes is very small in Fig. 6 (a). On the other hand, the average hop number is bigger in Fig. 6 (b). The bigger α relaxes the control of load balancing among BNs, and instead gives the freedom for each RN to take the closest BN greedily in order to reduce the hop delay. Therefore, the average hop number in Fig. 6 (b) is smaller when α is bigger at the cost of higher association variance in Fig. 6 (a).

5.4. Impact of l

Varying from 0.55 to 0.95, the moving proportion l introduces a trade-off between the number of iteration steps and the number of adjustment steps as shown in Fig. 7. When we have a lower l, the BNs move to 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 high, although a BN will move towards its target position in a very few iterations, it will jump over and ignore those positions that may provide a lower access delay. Thus after all the iterations, the algorithm needs more steps to do the adjustment, reconsidering those positions it ignored before.

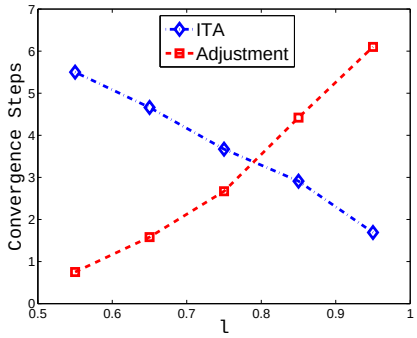

Figure 7. Impact of l

5.5. Impact of $InitialPosition$

We propose the method to find the initial positions by find the mass center of the RNs and then do BFS traverse. In this simulation, we did another 5 experiments in 5 different random RN network. In each RN network, we set initial positions as both the mass center (50 simulations) and the random pick (50 simulations) to obtain the average result. With random pick up initial position, we first randomly pick an RN position and then still do BFS traverse rooted on this RN to find other $k-1$ BN initial positions as close as possible to the root RN. Illustrated in Fig. 8 (a), the performances with the mass center and the random pick are close, which means our BFS initial deployment is pretty robust with different initial positions. The initialization in reference to the mass center, however, offers a faster convergence speed, while the initialization through random pick generally takes longer time to converge as shown in Fig. 8 (b).

6. Conclusion

In this paper we propose algorithms for the deployment of a Reinforcement 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 prove that the problem is NP-hard.

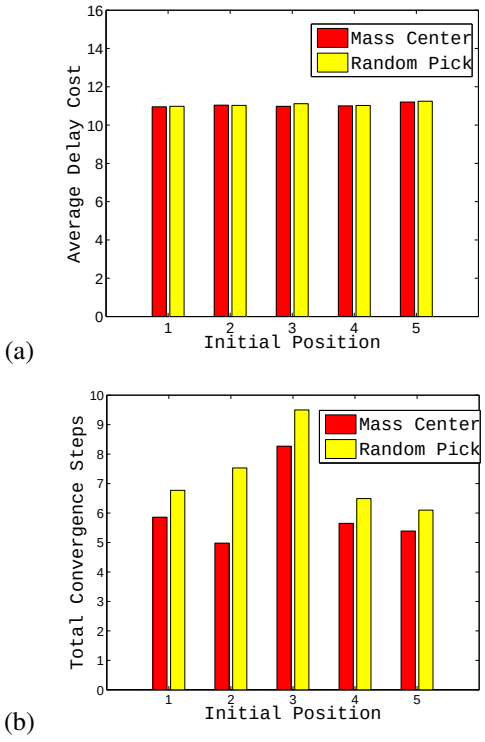

Figure 8. Impact of the initial BN positions (a) Average delay cost; (b) Total number of convergence steps.

Inspired by the theories in data mining and robotics fields, we propose an iterative and adaptive algorithm (ITA) which can construct a robust Reinforcement Backbone Network which is not sensitive 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 performance close to that obtained through the generic algorithm, which requires several days of running. Our study indicates that the proposed ITA algorithm is promising for the deployment of a connected Reinforcement 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 Trans. Inf. 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 *Proc. IEEE Computer Society's 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'04)*, Oct. 2004.

[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*, Oct. 2008.

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

[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 Trans. Knowl. Data Eng.*, 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)*, Jun. 2002.

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

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

[13] 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.