2008 IEEE International Conference on
Robotics and Automation
Pasadena, CA, USA, May 19-23, 2008

Backbone-based Roadmaps for Robot Navigation in Sensor Networks

Zhenwang Yao and Kamal Gupta
Robotic Algorithms and Motion Planning (RAMP) Lab
School of Engineering Science, Simon Fraser University
Email: {zyao,kamal}@cs.sfu.ca

***Abstract*—In this paper, we propose a distributed path planning method for robot navigation amidst a wireless sensor network. Our method uses communication backbone as a roadmap. In building and maintaining the roadmap, it takes path safety and network longevity into account, and therefore the roadmap adapts to dynamic dangers and evolves over time to increase network longevity. To find a safe path, a navigation field is propagated over the roadmap and the shortest path is computed. Simulation results show that as compared to existing methods our method finds a safe path with less communication cost, and in dense networks it generates smaller roadmaps. We also provide theoretical bounds on the path quality in terms of path length.**

I. Introduction

Recently there has been significant interest in sensor network for environmental monitoring and many other applications, and in many cases, robots have been an integral part of the system [1]. In this paper, we explore one possible interaction between robots and sensor networks: navigation in a sensor network. By communicating with a sensor network deployed in an environment and continuously perceiving changes, robots can respond to events outside their perception range, and move with guidance obtained from the network. The application can be to guide a robot (or humans) toward a goal across a hazardous environment [2], especially in an emergency situation [3]. The problem of robot navigation in sensor networks can be formulated as a robot path planning problem in presence of obstacles, with danger areas (e.g., spots with excessive heat) of the network being modeled as obstacles.

One naive way to solve the problem is for the robot to gather all sensor readings from the network and plan a path in a centralized manner. The problems with doing so are that (a) the communication cost of gathering information is high, and (b) it is hard to adapt to dynamic environments due to the relatively long time needed for information gathering. Therefore distributed methods are desired for path planning in sensor network environments, and several distributed methods have been proposed for the problem. In [4], [5], [6], a navigation potential field is propagated across the sensor network, and then the field is used to navigate the robot. When the robot moves in the sensor field, the robot interacts with neighboring sensor nodes to get optimal motion with respect to a certain objective, such as distance to goal or safety. The major problem with this type of approach is that flooding algorithms are used to propagate the navigation field, and consequently, the field needs relatively long time to become stable when a goal is specified, and the energy consumption could be high, due to the high volume of communication resulting from flooding.

Recently, inspired by Probabilistic Roadmap Method (PRM) [7], roadmap based methods have been proposed to reduce flooding and hence communication cost [8], [9], [10], [11]. Rather than propagating the navigation field over the entire network, these methods navigate the robot through a roadmap, a smaller subset of the network. As suggested in [9], [11], in addition to being more efficient in dynamic environments, the roadmap based methods can incorporate physical obstacles, provided the robot is mounted with necessary sensors. However, all these methods still need a certain degree of flooding for all nodes in roadmap construction, and for non-roadmap nodes to compute the navigation field to guide the robot toward the roadmap; therefore they do not take full advantage of the roadmap. In this paper, we propose a different roadmap method, which eliminates flooding in the construction of roadmap.

The proposed method is inspired by distributed clustering algorithms for constructing communication *backbone* [12]. We first extract the backbone of the sensor network via a clustering algorithm adapted from [13], and use the backbone as roadmap for path planning. The extracted *backbone* is a virtual network formed by a relatively small subset of the network, and provides a hierarchical organization of the original network. The basic idea is to group a set of nodes based on physical proximity, and represent each group with a single node as *clusterhead*; these clusterheads and selected connections among them form the backbone.

It is advantageous to use backbone network as a roadmap. There is no need of flooding in order to construct the backbone; a node decides whether to become a backbone node, based on its 2-hop neighbor information. The size of backbone depends on network connectivity, and when connectivity is high, the number of clusterheads decreases, resulting in a smaller roadmap. Furthermore, the clusterheads will spread out in the network, and every node will be at most 1-hop away from the roadmap. This property brings two benefits: (a) the length of a computed path is bounded by a constant ratio to the optimal one; and (b) there is no extra cost for a non-roadmap node to compute navigation field toward the roadmap. As seen in the simulation results, compared to existing method [11], the backbone based roadmap results in smaller roadmaps, and fewer messages to construct the roadmap, and fewer messages to navigate a robot to the given goal.

978-1-4244-1647-9/08/$25.00 ©2008 IEEE.

The remainder of the paper is organized as follows. In Section II, we give some background introduction including related works, basic concept of clustering algorithms. Then, we detail the proposed algorithm in Section III, and present some theoretical results in Section IV. In Section V we show simulation results, followed by conclusion and future work in Section VI.

II. Background

A. Problem Overview

A sensor network consists of a set of sensor nodes, $\mathcal{S} = \{s_1, \cdots, s_n\}$, in the environment. A sensor node can measure state of the environment (e.g., temperature) within its sensing range, d_s. Danger areas (e.g., with excessive heat) can be detected by sensors, if the sensor reading is beyond a certain threshold. Two nodes can communicate with each other, if they are within distance, d_c, the communication range. In this paper, we assume a simple fixed radius cookie-cutter communication model. We also assume that sensor nodes know their location. The network formed by sensor nodes is modeled as a proximity graph, $G(V, E)$, whose vertices represent sensor nodes, and edges represent communication links between nodes.

The robot, $\mathcal{R}$, is assumed to be a point (circular) robot. The robot is mounted with sensing and wireless communication devices that can communicate with sensor nodes within communication range, d_c. We assume localization devices such as GPS are not available on the robot (e.g., in indoor environments).

The robot responds to a certain event (e.g., a victim in a rescue scene [3]). When a sensor node s_g in the network detects a target event, the sensor network navigates the robot $\mathcal{R}$ toward s_g while avoiding danger areas (e.g. fire). The challenges of the problem come from that fact that (a) the robot does not have localization device, (b) the path should be danger-free, and (c) the environment may change over time. The problem can be solved in two steps: (1) **Path planning**, finding a feasible path in terms of sensor nodes; and (2) **Path navigation**, navigating the robot along the path by moving the robot from one node to another. The first step is done within sensor network alone, and the second is done with continuous interaction between network and the robot. The focus of this paper is the first step: path planning.

B. Related Work

1) Robot path planning: General approaches to solve autonomous path planning and navigation problems can be found in [14]. For many difficult path planning problems, (e.g., problems with high-dimensional robots, or complicated environments), sampling-based methods such as PRM [7] have been successful in the past decade. The approach we take in this paper is essentially this type of method, in the sense that it uses a limited number of sensors (samples) to navigate the robot. However, for path planning in sensor networks, the key challenge is how to do the path planning in a distributed way [10].

2) Robot navigation in sensor networks: Several distributed algorithms have been proposed for the problem. One class of algorithms is to propagate a navigation field over the entire sensor network: messages flood from the goal, so that each sensor node will have knowledge about best movement to reach goal. [4], [5], [6], [15] fall into this category, and they differ in the definition of navigation field based on different objectives. All the above algorithms, however, use flooding to propagate a navigation field, which is not efficient in terms of network energy consumption due to high communication volume. Different techniques have been proposed to reduce flooding. [8], [9] propose to find a feasible path incrementally as the robot travels through the sensor network, and flooding is reduced by limiting query to only nodes in vicinity of the robot; [9] further reduces communication by building a virtual grid road-map in the area, and limiting query to nodes close to roadmap edges. [10] proposed to scale down the original network by building a skeleton graph based on geographic information. Recently, [11] proposed to build a roadmap by randomly choosing a certain number of nodes as milestones, and making connections among them. As mentioned before, these methods still involve a certain degree of flooding.

C. Clustering and graph domination

Clustering is a technique of scaling down networks with a large number of nodes, and it is mainly used for routing in large scale networks, especially sensor networks. Given a graph $G = (V, E)$, the aim of clustering is to find subsets of V, $\{V_1, V_2, \cdots, V_k\}$, such that each of V_i induces a connected subgraph of G, and $\bigcup_{i=1}^{k} V_i = V$. The subsets are called clusters, and they can overlap with each other. A particular vertex in V_i is chosen to represent the cluster, called the *clusterhead*.

Distributed clustering algorithms have been introduced in [16], and have been extensively studied since the past decade for ad hoc networks. Good reviews of clustering algorithms can be found in [17], [18], [19]. A natural way to cluster an ad hoc network is to use graph domination and its variants. A *dominating set*, D, is a set of vertices that makes all vertices of the graph either in D or adjacent to at least one vertex in D. Formally,

$$D \subseteq V, \quad \forall u \in V - D, \quad \exists v \in D \quad \text{s.t. } (u, v) \in E.$$

The members of a *minimum dominating set* (MDS) can be used to represent *clusterheads*, each of which forms a cluster with their neighbors. A *connected dominating set* (CDS), $C \subseteq V$, is a dominating set of G, such that the subgraph induced by C is connected. A CDS in general includes a set of clusterheads as in MDS, and gateways that connect them. The problems of finding (MDS), and *minimum connected dominating set* (MCDS) have been proved to be NP-hard [20]. Due to the hardness of the domination problems, different algorithms use different heuristics to choose dominating sets as clusterheads (and gateways). In this paper, we adopt the TMPO algorithm, Topology Management by Probity Ordering, in [13], which uses a comprehensive heuristic combining multiple criteria. We

now outline the TMPO algorithm in the next section. Please refer to [13] for detailed description.

D. TMPO: CDS election based on priority

With TMPO, a node decides whether or not to be a CDS member based on a knowledge of its 2-hop neighbors, and their priority which is a function of node energy and mobility. There are three types of nodes in the CDS, **clusterheads (CH)**, **doorways (DW)**, and **gateways (GW)**. Clusterheads form a dominating set, and doorways are a special type of gateways connecting clusterheads. We call clusterheads, doorways, gateways CDS nodes, and all other nodes as regular nodes. A node decides whether to become a CDS node according to the following criteria, and note that these criteria are stated with respect to local 2-hop neighbor information.

A node becomes a **CH**, if it satisfies *either* of the following conditions:

- **(C.1)** It has the highest priority among its 1-hop neighbor;
- **(C.2)** It has the highest priority among some node's 1-hop neighbors;

It has been proved that clusterheads elected based on (C.1) and (C.2) make an approximation to MDS. Furthermore, for any clusterhead, the closest clusterhead (if there exists one) is at most 3-hop away. To form the backbone, connections between clusterheads need to be established. If two clusterheads are only 1-hop away, the link between them is kept. If two clusterheads are 2-hop away, and there is no other clusterhead in between, a *gateway* is needed to connect them. If two clusterheads are 3-hop away, and there is no other clusterhead in between, a *doorway* is needed to bring them one-hop closer, and a *gateway* is needed to connect the doorway and the other clusterhead. Simply put, the shortest path (of length 3) with the highest priority node is used to connect these two clusterheads, and the node with the highest priority becomes *doorway* and connects to one of the clusterheads, and a common neighbor (with the highest priority) of the doorway and the other clusterhead is elected as a gateway to connect the doorway and the other clusterhead. More specifically, a node becomes a **DW**, if it satisfies *all* of the following conditions:

- **(D.1)** It has one clusterhead, c_1, as 1-hop neighbor.
- **(D.2)** It has another clusterhead, c_2, as 2-hop neighbor, but no other clusterhead neighboring c_2.
- **(D.3)** c_1 and c_2 are not neighbors, and there is no other nodes connecting c_1 and c_2;
- **(D.4)** There is no other path between c_1 and c_2 that has a higher priority node.

A node becomes a **GW**, if it satisfies *all* of the following conditions:

- **(G.1)** It has two disjoint clusterheads, or one clusterhead and one doorway, n_1 and n_2 as 1-hop neighbors;
- **(G.1)** There is no 1-hop neighbor that is a common neighbor of n_1 and n_2, and has higher priority.

III. Backbone for navigation

A. Node priority

In [13], priority is defined as a function of energy and mobility, and a node that has higher energy and lower mobility is more likely to be a clusterhead, thereby achieving longer lifetime, and more stable backbone. We adapt and generalize this definition for robot navigation. We define the priority P_i, of node i, as a function of energy and safety:

$$\begin{aligned} P_i(d_o, E) &= B_1 \bigoplus B_2 \bigoplus B_3 \\ B_1 &= \lfloor d_o \cdot \log^2(1 - 0.9E) \rfloor \\ B_2 &= d_o \\ B_3 &= d_o \cdot node\_id \end{aligned} \tag{1}$$

where B_i is a bitstring and $\bigoplus$ is bit-concatenation operation, d_o is the distance to danger, and E is remaining energy. As $d_o \geq 0$, the priority will be non-negative, and when $d_o = 0$, and the node will *lose its privilege to become a CDS node*. $E \in [0, 1]$, when the battery is depleted, the logarithmic term goes to zero, and B_1 becomes zero.

B. Roadmap construction

The backbone construction procedure essentially elects members of CDS, as described in Section II-D. We make two key modifications to the original TMPO algorithm, since the problem imposes safety constraints. First, with the priority defined in previous section, nodes that are further away from danger, and have more energy are more likely to be elected in the CDS. In order to eradicate the possibility of electing a node in danger, which has zero priority, we include an extra criterion for CDS election:

- **(E.1)** A zero-priority node is not eligible to be a CDS node.

Furthermore, during the election, a node simply ignores a neighbor, if this neighbor has zero-priority, as if the neighbor were not in the list of neighbors. Equivalently, the election is done with respect to the network with all nodes in danger removed. Note that inclusion of (E.1) may result in a disconnected backbone, even if the original network is a connected one. In such a case, the backbone will be the union of CDS for each connected component. For the sake of easy description, we may interchange the terms of CDS and backbone throughout the paper.

The second modification is regarding information propagation after election. In [13], clusterheads and doorways need to propagate their type (CH, DW, or GW) information, because election of doorways depends on which nodes are clusterheads in neighborhood, and election of gateways depends on information of clusterheads and doorways. In our problem, in order to (further) reduce the communication volume for goal dissemination in later stages, we propose to propagate extra information for doorways and gateways. That is, besides type information, a doorway or gateway should also propagate the information as to which clusterheads (or doorways) it connects.

The backbone roadmap is constructed in a distributed fashion. The detailed algorithm is shown in Algorithm 1, and the same algorithm runs in every sensor node. When CDS nodes are elected, connections among them are formed implicitly by constructing **employers** and **employees**. For a doorway, **employers** is a list of clusterheads it connects, for a gateway **employers** is a list of clusterheads and doorways it connects, and a clusterhead has an empty set of **employers**. Once a node changes its type (e.g., newly elected as a clusterhead), it

propagates its type by broadcasting a *TYPE_CHG* message for two hops. *TYPE_CHG* message includes **employers**, and upon reception of the message, clusterheads and doorways update their **employees** by including all doorways and gateways that connect them to another clusterhead. Clearly, a gateway has an empty set of **employees**. **employers** and **employees** will be used in the next stage for goal dissemination for navigation.

Algorithm 1: Roadmap construction

```
1  employers ← ∅; type ← Regular;
2  if ((C.1) or (C.2)) and (E.1) then
3  |  type ← CH;
4  else
5  |  if (D.1-4) and (E.1) then
6  |  |  type ← DW;
7  |  |  employers ← CHs that it connects;
8  |  endif
9  |  if (G.1-2) and (E.1) then
10 |  |  type ← GW;
11 |  |  employers ← CHs and DWs that it connects;
12 |  endif
13 endif
14 if type changes then
15 |  omsg.msgid ← TYPE_CHG;
16 |  omsg.content.type ← type ;
17 |  omsg.content.param ← employers ;
18 |  Broadcast omsg ;
19 endif
20 if imsg received and imsg.msgid=TYPE_CHG then
21 |  if from 1-hop neighbor then Broadcast imsg ;
22 |  if (type = CH or DW) then Update employees;
23 endif
```

Algorithm 2: Goal Dissemination

```
1  forme ←false;    //Is the message for me?
2  if imsg received and imsg.msgid=GOAL then
3  |  if (type is CH or GW or DW) then
4  |  |  if sender ∈ employees then forme ←true ;
5  |  |  else if sender ∈ employers then forme ←true ;
6  |  else
7  |  |  forme ←true ;
8  |  endif
9  |  if forme = true and imsg.hops+1<hopstogoal then
10 |  |  imsg.hops ← hopstogoal ← imsg.hops+1;
11 |  |  nexttogoal ← imsg.sender;
12 |  |  if type ≠ Regular then Broadcast imsg ;
13 |  endif
14 endif
```

C. Goal dissemination

The purpose of goal dissemination is to notify, if possible, every node of the specified goal, so that every node can provide guidance (to the robot) in case the robot is in the neighborhood. Now that we have constructed backbone as a roadmap, the goal dissemination propagates a potential field over the network via the roadmap, and the best path is found by following the field. The definition of path quality depends on applications, and different potential functions can be use, e.g., [11] uses a weighted combination of path length and maximum danger level. While nothing prevents one from adopting other potential functions, here we simply choose the shortest path in the roadmap, as we have already taken safety into consideration when constructing the roadmap, and all nodes in the roadmap are in safe areas. The procedure is shown in Algorithm 2. The goal node initiates the procedure by broadcasting a *GOAL* message, and backbone nodes forward the message to every node of the network. If the received message gives a better path to goal, a node updates the information, and the message is re-broadcast only if the receiving node is a backbone node. The number of messages is reduced by limiting rebroadcasting (forwarding): A backbone node only processes messages from its **employers**, **employees**, and the goal node. If a message comes from any other nodes, the receiving node simply discards the message; A regular node receives and processes GOAL messages but never forwards the message.

D. Robot navigation

The robot navigates with the cooperation of sensor nodes. When a robot moves amidst the sensor network, it constantly broadcasts a *QUERY* message, and waits for response from sensor nodes in the neighborhood. When sensor nodes receive *QUERY* message, they respond with *NAVIG* messages which contain current distance to goal (**hopstogoal**) and best movement toward goal (**nexttogoal**). This information for every node has been updated in the goal dissemination stage. When receiving *NAVIG* message, the robot chooses the best movement. After execution, the query-respond-move procedure repeats again, until the goal is reached.

E. Dynamic environments

Algorithm 3 shows how the roadmap adapts to dynamic dangers. When a sensor node detects danger (e.g., high temperature), d_o becomes 0, and it broadcasts a *DANGER* message. Nodes receive the message, update their value of d_0, and forward the message up to a certain distance, d_{max}. Upon change of d_0, node priority changes, and re-election of CDS is done locally again to adjust the roadmap for the changed environment. When some nodes in backbone have changed, the navigation field over the previous backbone needs update. As local changes in backbone may result in global changes in the navigation field, rather than updating the field locally, a *STALE* message is sent to the goal node, and another round of goal dissemination will be initiated.

Remarks: Clearly, danger detection may involve non-backbone nodes, as a non-backbone node can also detect dangers. The propagation of danger information may need a small degree of flooding depending on the value of d_{max}. Note that, however, there is no flooding needed in backbone election/re-election (i.e., roadmap construction), where all messages are sent at most 2 hops.

F. Load balance: network longevity

Generally backbone nodes should be kept alive for navigation, and hence they consume more energy. To achieve overall longevity of the sensor network, nodes in the network should share their roles as backbone nodes. With the definition of priority in Eq.(1), this can be easily achieved. Remaining energy is represented by a set of discrete levels. When the energy

drops to the next level, a node broadcasts an *ENERGY* message to notify its neighbors within two hops. Upon reception of the message, its neighbors recompute the priority and reelect backbone if needed.

Algorithm 3: Roadmap Maintenance

```
   /* Upon sensor reading changes.                    */
 1 if reading > T then
 2 |  d_0 ← 0;
 3 |  omsg.msgid = DANGER;
 4 |  omsg.danger.dist = 0;
 5 |  Broadcast omsg ;
 6 endif
   /* Upon receiption of DANGER msg.                  */
 7 if imsg received and imsg.msgid=DANGER then
 8 |  if imsg.danger.dist+1< d_0 < d_max then
 9 |  |  imsg.danger.dist← d_0 ←imsg.danger.dist+1;
10 |  |  Broadcast imsg ;
11 |  endif
12 endif
   /* Wait for a certain period, then:                */
13 Call procedure in Algorithm 1;
14 if type changes then
15 |  Send a STALE message to goal node;
16 endif
```

G. Other considerations for realistic implementation

a) Synchronization: For simplicity of explanation in description and theoretic analysis, we assume synchronous communication. In real implementation, asynchronous mechanism is used to avoid congestion and synchronous sudden loss of the old network states. As in [13], we use a simple random time slot offset for a node to uniformly distribute the local backbone re-election and communication.

b) Robustness: In our problem, sensor nodes are static, and we do not globally update the backbone topology periodically as in [13], since such update can be costly. Instead, we only update backbone locally when it is indeed necessary: dangers are detected, or a node's battery drops to a certain level. In realistic scenarios, the backbone may not always reflect the actual environment. For instance, a node can be burnt before it has a chance to send out a *DANGER* message, or imperfect communication may result in inconsistent neighbor information. This needs to be taken into consideration, in order to make the proposed algorithm robust enough to implement on a real system. What we do is somewhat similar to the strategy in [11]. We verify the path on the execution phase: before the robot moves, it confirms (with current associated sensor node) that the next sensor node is indeed alive and safe. If otherwise is indicated, a local CDS re-election is initiated, and the robot waits until the backbone is updated and a new movement is given.

IV. Theoretical Results

We define a subset $V_{bad} \subseteq V$ as nodes in dangerous regions, $V_{good} = V - V_{bad}$ as nodes in safe regions, and V_{rdmp} as roadmap nodes. G_{good} is a subgraph of G, induced by V_{good}. $G_{rdmp}(V_{rdmp}, E_{rdmp})$ is the constructed backbone roadmap, and $(u, v) \in E_{rdmp}$ if and only if $u, v \in V_{rdmp}$ and u is in **employers** or **employees** list of v. We have the following lemma:

Lemma 1: All nodes in the roadmap constructed as in Algorithm 1 are safe nodes. That is $V_{rdmp} \subseteq V_{good}$. If G_{good} is a connected graph, V_{rdmp} is a CDS of G_{good}, and G_{rdmp} is a connected graph.

Proof: Recall that the algorithm elects backbone nodes based on priority defined in Eq.(1). When a node, s_i, is in a danger area, $d_o = 0$, and consequently $P_i = 0$. A node with zero priority will not be selected, due to condition (E.1). The second half of the lemma follows from the fact that, on election, a node ignores all neighbors with zero-priority. When s_i is in danger areas, it notifies all its neighbors, and all its neighbors take s_i out of consideration on backbone selection. The neighbor list (in each node) that the CDS election is based on is same as the neighbor list in G_{good}. ■

A more general statement of Lemma 1 is: V_{rdmp} consists of CDSs of all connected components of G_{good}. Assume that the sensing range d_s is larger than half of the communication range $d_c/2$, which implies that moving between two safe nodes results in a safe path. With Lemma 1, the correctness of the algorithm follows.

*Corollary 1: (**Correctness**)* If there exists a safe sensor path in the original network, G, then there exists a safe one in the constructed backbone, G_{rdmp}.

Note that we find the shortest path in the roadmap as the best path, since we have already taken safety into account when constructing the backbone/roadmap: the closer is a node to danger, the smaller the priority it has, and the less chance it has to be chosen as a backbone node. We now give bounds on the path length.

*Theorem 1: (**Path length**)* For a connected graph G_{good}, the shortest path found in the constructed roadmap, G_{rdmp}, is bounded by the shortest path found in G_{good}:

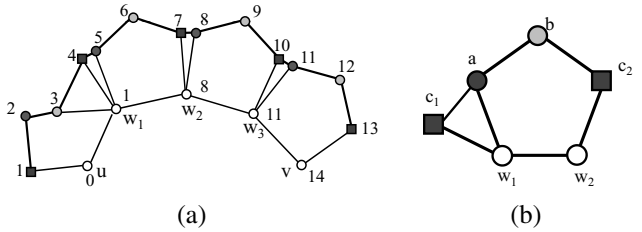
$$D_{rdmp}(u, v) \leq 3D_{good}(u, v) + 2$$

where $D_{rdmp}(u, v)$ and $D_{good}(u, v)$ are lengths of the shortest path between u and v, in G_{rdmp}, G_{good}, respectively.

Proof: We show that we can always find a path in backbone related to the shortest path. From Lemma 1, vertices of G_{rdmp} is a CDS of G_{good}. Assume, in the worst case, all nodes in a shortest path between u and v are not in the backbone. Consider Fig. 1(a). The squared nodes are clusterheads, the dark/grey round nodes are doorway/gateway nodes, and all other white nodes are regular nodes. The thick lines represent connections in backbone, and the thin ones represent those not in backbone. (The path u, w_1, w_2, w_3, v is an example of the worse case shortest path.)

As shown in Fig. 1(b), since each vertex must be dominated by a clusterhead, clusterheads (e.g. c_1 and c_2) dominating two adjacent vertices (e.g. w_1 and w_2) are at most 3-hops away. As w_1 and w_2 are not in backbone, there must exist another path of length ≤ 3 between c_1 and c_2 in backbone (e.g. $c_1 - a - b - c_2$). There must be at least one connection between one of $\{w_1, w_2\}$ and one of $\{b, c\}$, otherwise, w_1, and w_2 would have been in backbone.

For the start and goal nodes(i.e., u, and v), each of them should have a dominating clusterhead node within 1-hop. So in total, $D_{rdmp}(u,v) \leq 3D_{good}(u,v)+2$. Fig. 1(a) also gives a worst case scenario, showing that the bound is tight; once any vertex in the shortest path becomes a backbone node, or two of them share a clusterhead, the path length reduces. ■



(a) (b)

Fig. 1. Bound on path length.

Algorithm 1 for roadmap construction is adapted from the approximation algorithm finding minimum CDS in [13], and the approximation ratio analysis of the algorithm is beyond the scope of this paper. But the following known results on the size of minimum connected dominating set give a rough idea of roadmap size. Let γ_c denote the cardinality of minimum connected dominating set, and Δ the maximum vertex degree.

*Theorem 2: (**Bounds on optimal roadmap size**)* For any connected graph G,

1) [21] $\gamma_c \leq n - \Delta$;
2) [22] $\frac{n}{(\Delta-1)} \leq \gamma_c \leq 2m - n$, where n, m is the number of vertices and edges of G respectively. The equality for lower bound is attained if and only if $\Delta = n-1$, and equality for upper bound is attained if and only if G is a path graph.

*Theorem 3: (**Communication complexity**)* Let $|CDS|$ be the size of the backbone. The roadmap construction procedure in Algorithm 1 takes $O(|CDS| \cdot \Delta)$ messages. The goal dissemination procedure in Algorithm 2 takes $O(|CDS|)$ messages.

Proof: Once a node is elected as a CDS node, a *TYPE_CHG* message will be sent for 2 hops. That is at most $(1+\Delta)$ messages for every node in CDS. Thus, in total, the number of messages is $O(|CDS| \cdot \Delta)$.

Assume that a node forwards the message as soon as it is received and processed. The Algorithm 2 basically finds a shortest path in the backbone, with the hop count as distance model [5]. Thus, if a node receives two GOAL messages, the one received later is further away from goal. So every node forwards only the first message it receives, and hence the total number of messages generated is $|CDS|$. ■

V. Simulation Results

We have performed simulations to show performance of proposed method using backbone as roadmap. We used an in-house developed software simulator, where sensor nodes are uniformly distributed within a $1000m \times 1000m$ field, and we assume there is no noise and interference.

A. Comparison with AER [11]

To compare with *AER*, we have done simulations for different sizes of network, and different communication ranges. In the first simulation, the communication range is fixed to 100m, and the network size ranges from 200 to 800 sensor nodes. In the second simulation, the network size is fixed to 200 nodes, and the communication range varies from 100m to 300m.

TABLE I
Performance in networks with different size, and communication range fixed to 100m.

Size	$\overline{d}$	Method	M_{con}	M_{goal}	Roadmap		Path	
					N_v	N_e	L_o	L
200	5	TMPO	1049	157	121	196		20
		AER	1135	272	96	150	18	21
300	8	TMPO	1624	288	165	322		18
		AER	2952	713	149	282	17	19
400	11	TMPO	2231	377	210	427		19
		AER	5665	1256	191	434	18	20
500	14	TMPO	2850	479	243	568		19
		AER	7852	2083	239	523	17	19
600	17	TMPO	3578	299	266	679		17
		AER	13099	3233	309	776	17	18
700	20	TMPO	4108	339	310	789		17
		AER	18913	4289	359	946	17	17
800	22	TMPO	4814	352	338	884		17
		AER	25305	6321	423	1174	17	17

TABLE II
Performance in networks with different communication range, and network size fixed to 200 nodes.

Range	$\overline{d}$	Method	M_{con}	M_{goal}	Roadmap		Path	
					N_v	N_e	L_o	L
100m	5	TMPO	1049	157	121	196		20
		AER	1135	272	96	150	18	21
150m	15	TMPO	1119	132	97	218		12
		AER	3161	639	86	193	11	13
200m	21	TMPO	1153	102	75	177		9
		AER	5754	860	95	233	8	9
250m	32	TMPO	1193	54	50	119		7
		AER	9465	896	92	296	6	7
300m	44	TMPO	1236	37	29	61		6
		AER	11346	859	74	251	5	6

The results are shown in Table I and II. The performance metrics we compared include: (1) number of messages needed to construct a roadmap, M_{con}, and the number includes messages used to collect neighborhood information; (2) number of messages needed for goal dissemination, M_{goal}. A message can either be a broadcast message or a unicast message, i.e., they both count as 1. (3) roadmap size, represented by number of nodes (N_v) and the number of links (N_e) used in the constructed roadmap. For AER, we set the probability of being a landmark to be 0.10 as used in [11], and roadmap size includes edge nodes. (4) path length, L, in terms of hop count. For each case, we also computed the average degree ($\overline{d}$) of the entire network, and the optimal path length L_o.

The simulation results show that TMPO (i) generates fewer messages for roadmap construction and goal propagation, and (ii) produces a smaller roadmap when the network connectivity is high (network size ≥ 600 in Table I, and range $\geq 200m$ in

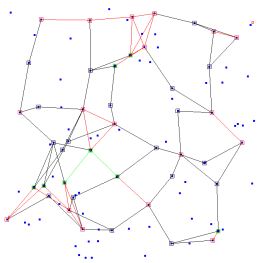


Fig. 2. Original roadmap.

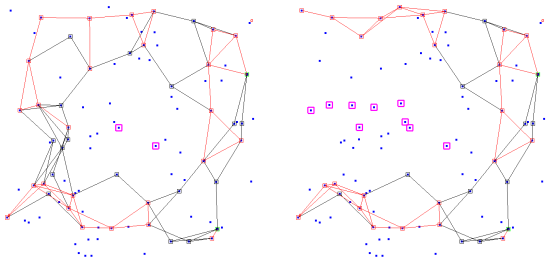


Fig. 3. Roadmap changes with dynamic danger. ⊡ represents danger

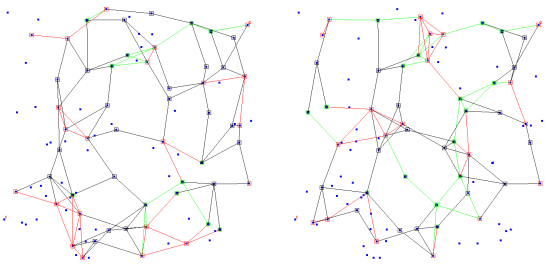


Fig. 4. Roadmap adapts over time.

Table II). The smaller number of messages is mainly due to use of broadcasting as opposed to unicast used in AER. The smaller roadmap for high connectivity is achieved because of the nature of dominating set; intuitively when the connectivity is higher, it allows a node to dominate more nodes, and hence a smaller dominating set in general. Also shown is that, the length of paths computed by both algorithms is generally fairly close to optimal, even though we have given theoretic bounds on path length for the proposed algorithm whereas there is not such guarantee in AER.

B. Roadmap changes upon danger

In this simulation we show how the roadmap changes in response to dynamic dangers. Fig. 2 shows the initial roadmap. When some dangers happen, the roadmap adapts to dangers, as shown in Fig. 3 (a) and (b), and in extreme cases, it may result in a disconnected roadmap.

C. Roadmap changes over time

In this simulation we show that nodes can share their roles as backbone nodes to average out the energy consumption to achieve a longer life of the network. Fig. 2 shows the initial roadmap, Fig. 4(a) shows the roadmap after 1 hour (scalable), and Fig. 4(b) shows the roadmap after 2 hours. We can see the roadmap changes over time, and contains different nodes.

VI. Conclusion

We proposed a new method for robot navigation in sensor networks. The method extracts backbone of the sensor network via a clustering algorithm, and uses the backbone as planning roadmap. The backbone consists of a connected dominating set (CDS) of the (safe portion of the) network, and is elected based on the defined priority which takes energy and distance to danger into account to achieve network longevity and path safety. With backbone as the roadmap, the method avoids flooding in both roadmap construction and path query (goal dissemination). As the backbone is related to network connectivity, when the network has high connectivity, the size of backbone decreases, resulting in a smaller roadmap for navigation. Through simulations, we show that the proposed method uses fewer messages for planning.

VII. Acknowledgement

The research is supported by Natural Sciences and Engineering Research Council (NSERC) of Canada.

References

[1] A. LaMarca, W. Brunnete, D. Koizumi, and M. Lease. Making sensor network practical with robots. In *Pervasive-2002*, 2002.

[2] P. Corke, R. Peterson, and D. Rus. Localization and navigation assisted by networked cooperating sensors and robots. *Int. J. Rob. Res.*, 24(9):771–786, 2005.

[3] V. Kumar, D. Rus, and S. Singh. Robot and sensor networks for first responders. *IEEE Pervasive Computing*, 3(4):24–33, 2004.

[4] M.A. Batalin, G.S. Sukhatme, and M. Hattig. Mobile robot navigation using a sensor network. In *ICRA-2004*, Apr. 2004.

[5] Q. Li and D. Rus. Navigation protocols in sensor networks. *ACM Transactions on Sensor Networks*, 1(1):3–35, August 2005.

[6] A. Verma, H. Sawant, and Jindong Tan. Selection and navigation of mobile sensor nodes using a sensor network. In *PerCom-2005*, March 2005.

[7] L.E. Kavraki, P. Svestka, J.-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566 – 580, 1996.

[8] G. Alanku, N. Atay, Chenyang Lu, and O.B. Bayazit. Spatiotemporal query strategies for navigation in dynamic sensor network environments. In *IROS-2005*, Aug. 2005.

[9] S. Bhattacharya, N. Atay, G. Alankus, C. Lu, B. Bayazit, and G.-C. Roman. Roadmap query for sensor network assisted navigation in dynamic environments. In *Distributed Computing in Sensor Systems*, pages 17–36. 2006.

[10] C. Buragohain, D. Agrawal, and S. Suri. Distributed navigation algorithms for sensor networks. In *INFOCOM-2006*, Apr. 2006.

[11] G. Alankus, N. Atay, Chenyang Lu, and O.B. Bayazit. Adaptive embedded roadmaps for sensor networks. In *ICRA-2007*, Apr. 2007.

[12] B. Das and V. Bharghavan. Routing in ad-hoc networks using minimum connected dominating sets. In *IEEE ICC-97*, 1997.

[13] L. Bao and J.J. Garcia-Luna-Aceves. Topology management in ad hoc networks. In *MobiHoc-2003.*, June. 2003.

[14] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[15] K.J. O'Hara, V.L. Bigio, E.R. Dodson, A.J. Irani, D.B. Walker, and T.R. Balch. Physical path planning using the gnats. In *ICRA-2005*, Apr. 2005.

[16] D. Baker and A. Ephremides. The architectural organization of a mobile radio network via a distributed algorithm. *IEEE Transactions on Communications*, 29(11):1694–1701, 1981.

[17] Y. Chen, A. Liestman, and J. Liu. Clustering algorithms for ad hoc wireless networks. In *Ad Hoc and Sensor Networks*. Nova Science Publisher, 2004.

[18] E. Hossain, R. Palit, and P. Thulasiraman. Clustering in mobile wireless ad hoc networks: issues and approaches. In *Wireless communications systems and networks*, pages 383–424. Springer US, 2004.

[19] Y. Wang, W. Wang, and X.-Y. Li. Distributed low-cost backbone formation for wireless ad hoc networks. In *MobiHoc-2005*, 2005.

[20] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, New York, 1979.

[21] S. T. Hedetniemi and R. Laskar. Connected domination in graphs. In *Graph. Theory and Combinatorics*. Academic Press, London, 1984.

[22] E. Sampathkumar and H.B. Walikar. The connected domination number of a graph. *J. Math. Phys. Sci.*, 13:607–613, 1979.