# Tree Bound on Probabilistic Connectivity of Underwater Sensor Networks

Md Asadul Islam
Department of Computing Science
University of Alberta
Edmonton, T6G 2E8, Canada
E-Mail: mdasadul@ualberta.ca

Ehab S. Elmallah
Department of Computing Science
University of Alberta
Edmonton, T6G 2E8, Canada
E-Mail: elmallah@ualberta.ca

***Abstract*—This paper considers Underwater Sensor Networks (UWSNs) with unanchored nodes that can move freely with water currents. Thus, node locations at any instant can only be specified probabilistically. When connectivity among some of the sensor nodes is required to perform a given function, the problem of estimating the likelihood that the network achieves such connectivity arises. Our work here formulates a parameterized probabilistic connectivity problem that serves this purpose when the network contains both sensor nodes and relay nodes. We show an exact dynamic programming algorithm for solving the problem on networks with tree-like structure; the algorithm yields lower bounds on the solution of the problem on any arbitrary network. The obtained simulation results investigate the gap between our obtained lower bounds and exact solutions on small networks, as well as the usefulness of our method in analyzing the effect of adding relay nodes to the network.**

***Index Terms*—Underwater sensor networks, probabilistic connectivity, probabilistic graphs.**

## I. Introduction

Research on Underwater Sensor Networks (UWSNs) has intensified in recent years. Interest in such research has been fueled by many important underwater sensing applications and services that can be supported by such networks (see, e.g., [1], [2]). The domains of such applications are diverse and can be roughly classified as scientific applications (e.g., collecting data on geological processes, analyzing water characteristics, study of marine life), industrial applications (e.g., monitoring underwater equipment and pipelines used in oil industry), humanitarian applications (e.g., search and survey missions, prediction of natural disturbances) and/or military and home land security applications (e.g., securing port facilities).

To serve the above diverse types of applications, various types of UWSN deployments are used. In [2], for example, UWSN deployments are classified as being either static, semi-mobile, or mobile. Static networks have nodes attached to underwater ground, anchored buoys, or docks. Semi-mobile networks may have collection of nodes attached to a free floating buoy. Nodes in semi-mobile networks are subject to small scale movements. Mobile networks may be composed of drifters with no self mobility capability, or nodes with mobility capability. Nodes in such networks are subject to large scale movements. UWSN deployments may occur over many short periods of times (e.g., several days at a time), so as to conduct several missions over a large area of interest.

Due to the importance of such applications, and the challenges encountered in their design, extensive work spanning all five layers of the Internet protocol stack appear in the literature. See, e.g., the chapters in [3], [4], and the survey articles in [5], [6]. Examples of real-world UWSN work include [7]–[11].

In this paper, we consider semi-mobile and mobile networks. Maintaining connectivity in such networks is a crucial aspect for any task requiring node collaboration. Our interest in on developing methodologies that allow a designer to analyze the likelihood that a network (or part of it) is connected at a given time interval.

For literature review, we note that several experimental and analytical results in oceanography literature have shaped our current understanding of mobility for underwater sensor networks. Of the vast literature existing in the field, we recall the following early landmark results. In [12], the authors report on several observations collected in the Gulf Stream using thirty-seven RAFOS drifters launched off Cape Hatteras. Mobility of the free floating drifters are tracked for 30 or 45 days. In [13], the author describes a 2-dimensional kinematic model of a meandering jet. The model captures the striking patterns of cross-stream and vertical motion associated with meanders observed in [12].

Investigations and results obtained in the above directions have been valuable for networking researchers approaching the challenge of modelling the mobility of underwater sensor networks. In [14], the authors adopt a kinematic model for capturing the effect of meandering sub-surface currents and vortices of free floating sensor nodes. The model, called the meandering current mobility model, is useful for large coastal environments that span several kilometers. It captures the strong correlations in mobility of nearby sensor nodes. Using simulation, the authors investigate several network connectivity, coverage, and localization aspects.

In [15], the authors consider sensor nodes with movement capability. Each node incurs both uncontrollable and controllable mobility (abbreviated as U-mobility and C-mobility). Using a grid layout that divides a geographic area into cells, the authors adopt a probabilistic U-mobility model. The model

978-1-4799-3784-4/14/$31.00 ©2014 IEEE

takes into consideration two types of effects: *local variety* effects (caused by reefs, turbulence), and main circulation effects (caused by wind, salinity). Using such model, the authors present an energy efficient approach for satisfying network coverage requirements.

In this paper, we adopt a simple *probabilistic locality* model where the geographic area under consideration is partitioned into disjoint regions (rectangles) using a grid layout. In our model, the use of a high grid resolution (i.e., a layout with small regions) can potentially give accurate results at the expense of decreased solution efficiency. We assume that one can utilize a physical model of underwater currents to compute the probability that a sensor node is located at a given region during some time interval of interest. For example, one may utilize the kinematic model adopted in [14] to compute such probabilities from a sufficiently large number of node trajectories generated by the model. Using such probability distribution, one obtains a *probabilistic* graph model of the network.

Our work here formalizes two problems, denoted A-CONN and S-CONN, that call for determining the likelihood that a probabilistic graph is entirely or partially connected. Our main contribution is an efficient dynamic programming algorithm to solve both problems on tree-like networks. The algorithm can be used to derive lower bounds on the solution of any arbitrary probabilistic network. Our devised algorithm extends a result in [16] to compute the probability that a given sequence of nodes in a probabilistic network forms a simple connected path. To the best of our knowledge, both the problems formulation and devised algorithm are novel aspects of the paper.

In the next two sections we outline the system model and problem formulations. Section 4 then gives a detailed description of the algorithm.

## II. Network Model

In this section, we present a network model that deals with UWSNs with arbitrary topologies where node location is described probabilistically. In addition to using sensor nodes, the model allows networks to utilize relaying nodes that do not perform sensing functions, but can enhance the overall network connectivity.

### A. Node Locality Sets

We denote by $V = V_{sense} \bigcup V_{relay}$ the set of nodes in a given UWSN $G$ where $V_{sense}$ is a subset of sensor nodes that can perform both sensing and data communication, and $V_{relay}$ is a subset of relay only nodes that do not perform sensing. We assume that $V_{sense}$ has a distinguished *sink* node, denoted $s$, that performs network wide command and control functions.

After some time interval $T$ from network deployment time, each node $x$ can be in some location determined by water currents causing node movement.

To simplify analysis, approaches in the literature typically divide the geographic area containing nodes into rectangles of a superimposed grid layout. Thus, at time $T$, each node $x$ can be in any one of a possible set of grid rectangles denoted $\mathrm{Loc}(x) = \{x[1], x[2], \cdots\}$. We call $\mathrm{Loc}(x)$ the *locality* set of $x$ (for simplicity, we omit the dependency on $T$ from the notation). Depending on the mobility model induced by water currents, node $x$ can be in any possible grid rectangle $x[i]$ with a certain probability, denoted $p_x[i]$. Computing such probability distribution is outside the scope of the paper. We assume, however, that such distribution is computable given enough information on the dynamical aspects of the water currents.

Henceforth, we use $x[i]$ to refer to node $x$ at the $i$th location index. For brevity, we also refer to $x[i]$ as the location of $x$ (rather than the grid rectangle containing $x$) at an instant of interest. To gain efficiency in solving large problem instances with large locality sets, it may be convenient to truncate some locality sets to include only locations of high occurrence probability, and ignore the remaining locations. In such cases, we get $\sum_{x[i] \in \mathrm{Loc}(x)} p_x(i) \leq 1$, if $\mathrm{Loc}(x)$ is truncated.

### B. Node Reachability

At any instant, node $x$ can reach node $y$ if the acoustic signal strength from $x$ to $y$ (and vice versa) exceeds a certain threshold value. In acoustic UWSN, the directions of water currents play an important role in signal delay (see, e.g., [11]). For simplicity, we assume that given the exact locations of $x$ and $y$, we can determine if $x$ and $y$ can reach each other, and if so, we set the link indicator $E_G(x, y) = 1$. Else, if no satisfactory communication can take place then we set $E_G(x, y) = 0$.

Our general objective in this paper is to develop effective methodologies for computing lower bounds on the likelihood that the network is totally, or partially, connected. To this end, we adopt the following rule: we set $E_G(x[i], y[j]) = 1$ if and only if the two nodes $x$ and $y$ can reach each other if they are located anywhere in their respective rectangles $x[i]$ and $y[j]$.

The above rule implies that connectivity between $x$ and $y$ is ignored if they can reach each other at some (but not all) pairs of points in their respective rectangles. As can be seen, ignoring connectivity in such cases results in computing lower bounds on the network connectivity, as required. Our devised algorithm presented below is **exact** with respect to the given relation $E_G$ that defines the input network $G$.

## III. Problem Formulation

We now introduce two fundamental probabilistic connectivity problems on any given UWSN $G = (V = V_{sense} \bigcup V_{relay}, E_G, \mathrm{Loc}, p)$. The first problem is the *all* sensor node connectivity problem:

**Definition (the A-CONN problem).** Given an UWSN $G$, compute the probability $Conn(G)$ that the network is in a state where the sink node $s$ can reach all sensor nodes. ■

The second problem is the *subset* sensor node connectivity problem:

**Definition (the S-CONN problem).** Given an UWSN $G$, and

an integer $n_{req}$, $n_{req} \leq |V_{sense}|$, compute the probability $Conn(G, n_{req})$ that the network is in a state where the sink node $s$ can reach a subset of sensor nodes having at least $n_{req}$ sensor nodes. ■

Thus, the A-CONN problem is an important special case of the S-CONN problem where $n_{req} = |V_{sense}|$. We next remark that the above problems share some basic aspects with the class of network reliability problems discussed in [17]. In particular, all such problems are defined over some type of probabilistic graphs where each node (and/or link) can be in any one of two, or more, states. In case of network reliability problems, a node or link can either be *operating* or *failed*, whereas in our present context, a node can be in any one of a possible set of locations.

Events of interest on such probabilistic graphs occur when the given network is in some particular *network states*. In our present context, a network state $S$ of $G$ arises when each node $x \in V$ is located at some specific location in its respective locality set $\mathrm{Loc}(x)$. Thus, if $V = \{v_1, v_2, \cdots, v_n\}$ then a state $S$ of $V$ can be specified by $\{v_1[i_1], v_2[i_2], \cdots, v_n[i_n]\}$ where each $v_\alpha[i_\alpha] \in \mathrm{Loc}(v_\alpha)$. Two states $S_1$ and $S_2$ are different if they differ in the location of at least one node. Assuming node locations are independent of each other, we have $\Pr(S) = \prod_{v_\alpha \in V} p_{v_\alpha}[i_\alpha]$.

In the A-CONN problem, a sate is *operating* if the sink $s$ can reach all sensor nodes in $V_{sense}$. Likewise, in the S-CONN problem, a state $S$ is *operating* if the sink node $s$ can reach a subset having at least $n_{req}$ sensor nodes. Let $\mathbf{S}$ be the set of all operating states $S$ of a given A-CONN or S-CONN problem then the required solution is given by $\sum_{S \in \mathbf{S}} \Pr(S)$.

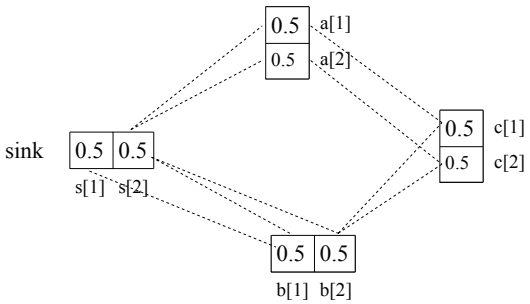


Fig. 1: An example network with locality sets

**Example.** Fig. 1 illustrates a probabilistic graph on 4 nodes where $V = \{s, a, b, c\}$, and the locality set of each node has 2 locations. The network has $2^4$ states. For the A-CONN problem, state $S_1 = \{s[2], a[2], b[2], c[2]\}$ is operating, and state $S_2 = \{s[1], a[1], b[1], c[2]\}$ is failed. ■

## IV. Exact Tree Algorithm

In this section, we first identify a class of probabilistic networks that have tree-like topologies. Next, we present an efficient algorithm that computes the exact $Rel(G, n_{req})$ on such class of networks. Our algorithm is conceptually simple, and the design is optimized to solve the A-CONN problem so as to alleviate the need for implementing a restricted version of the algorithm.

### A. Probabilistic Networks with Tree Topologies

To start, we say that a probabilistic network $G = (V, E_G, \mathrm{Loc}, p)$ has the topology of a conventional tree $T = (V, E_T)$ if whenever $E_G(x[i], y[j]) = 1$ then $(x, y) \in E_T$. Note that the definition allows two nodes $x$ and $y$ to be adjacent in $T$, and yet they can take positions, say $x[i]$ and $y[j]$, such that $E_G(x[i], y[j]) = 0$. Thus, each state $S$ of $G$ gives the tree $T$ with possibly some missing links.

In any such tree network $G$, one may safely delete a relay node $x \in V_{relay}$ that appears as a leaf node without changing the problem solution. This observation holds since relay nodes are relevant only if they connect some sensor node to the sink $s$. We henceforth assume, without loss of generality, that the input tree network $G$ has no relay leaf.

### B. Overview of the Algorithm

The algorithm (Function Conn in Fig. 2) employs a dynamic programming approach. It takes as input an instance $(G, n_{req})$ of the S-CONN problem, and a tree $T = (V, E_T)$ on the set $V$ of nodes with no relay leaves. The function computes the exact solution $Conn(G, n_{req})$ of the given instance.

We consider $T$ as a tree rooted at the sink $s$. Each node $y$ in $T$ has a parent node $x$ on the unique path from $y$ to the root $s$. Each such node $y$ is a root of a subtree, denoted $T_y$, obtained by removing the link $(y, parent(y))$ from $T$.

The key variables and data structures in the function are as follows.

- $type(x)$: For any node $x$, $type(x) = 0$ if $x$ is a relay node, and $type(x) = 1$ if $x$ is a sensor node.
- $n_x$ $(= n_{x,sense} + n_{x,relay})$: $n_x$ is the total number of sensor nodes $n_{x,sense}$ and relay nodes $n_{x,relay}$ in subtree $T_x$ (including node $x$)
- $n_{x,min}$: The minimum number of sensor nodes in $T_x$ that should be connected to the sink in any operating state of the network. So, $n_{x,min} = \max(0, n_{req} - (n_{sense} - n_{x,sense}))$. Here, $n_{sense} - n_{x,sense}$ is the number of available sensor nodes not in $T_x$, and thus $n_{req} - (n_{sense} - n_{x,sense})$, if non-negative, is the minimum number of sensor nodes of $T_x$ required in any operating state of the network.
- $count_{x,min}$: The minimum number of sensor nodes in the part of $T_x$ processed thus far that should be connected to the sink in any operating state of the network.
- $DCH(x)$: Each iteration of the main loop in Step 2 identifies a non-sink leaf node $y$ whose parent is denoted $x$. The function then processes, and then deletes node $y$. Thus, in any iteration, each node $x$ may have some of its children processed and deleted. We denote such set of $x$'s deleted children by $DCH(x)$.
- Tables $R_x$ (and $R'_x$): Each node $x \in V$ is associated with a table $R_x$. The table stores *key-value* mappings. Each key is a pair $(i, count)$ where $i$ is a possible location

index of $x$, and $count$ is a number of sensor nodes that are descendants of $x$ (including $x$ itself) in the graph processed thus far. Roughly speaking, at any iteration of the main loop, $R_x(i, count)$ is the probability of obtaining a state over the subset of nodes in $T_x$ processed in previous iterations where $x[i]$ reaches exactly $count$ sensor nodes in such subset of $T_x$.

**Function** $\mathrm{Conn}(G, T, n_{req})$
**Input:** An instance of the S-CONN problem where $G$ has a tree topology $T$ with no relay leaves
**Output:** $Conn(G, n_{req})$

1. **foreach** (node $x$ and a valid location index $i$)
   {
   set $R_x(i, type(x)) = 1$
   $n_{x,min} = \min(0, n_{req} - (n_{sense} - n_{x,sense}))$
   }
2. **while** ($T$ has at least 2 nodes)
   {
3. Let $y$ be a non-sink leaf of $T$, and $x = parent(y)$
4. **foreach** (key $(i, count) \in R_y$) $R_y(i, count) *= p_y[i]$
5. set $R'_x = \phi$
6. **foreach** (pair of keys $(i_x, count_x) \in R_x$ and $(i_y, count_y) \in R_y$)
   {
7. $count = \min(count_x + count_y, n_{req})$
8. $count_{x,min} = type(x) + n_{y,min} + \sum_{z \in DCH(x)} n_{z,min}$
9. **if** ($count < count_{x,min}$) **continue**
10. $R'_x(i_x, count) += R_y(i_y, count_y) \times R_x(i_x, count_x) \times E_G(x[i_x], y[i_y])$
    }
11. set $R_x = R'_x$; remove $y$ from $T$
    }
12. return $\sum_{x[i] \in \mathrm{Loc}(x)} R_s(i, n_{req}) * p_s[i]$

Fig. 2: Pseudo-code for function Conn

### C. Main Steps

Step 1 initializes table $R_x$ for each node $x$ as follows. For each possible location index $i$ of $x$, set $R_x(i, count = 1) = 1$ if $x$ is a sensor node. Else ($x$ is a relay node), set $R_x(i, count = 0) = 1$. Step 1 also initializes $n_{x,min}$.

Steps 2-11 form the main loop of the function. The loop iteratively finds a leaf node $y$ that is not the sink $s$, processes node $y$, and then removes $y$ from the tree $T$. Processing a node $y$ with parent $x$ is done as follows.

Step 4 updates each entry $R_y(i, count)$ by multiplying the entry with $p_y[i]$. As can be seen, this update operation is done in the iteration that ends by removing $y$. Step 5 initializes the temporary table $R'_x$ to empty.

Steps 6-10: the loop in step 6 performs a *cross product* of tables $R_x$ and $R_y$, storing the result in table $R'_x$. In the cross product, each pair of possible keys $(i_x, count_x)$ and $(i_y, count_y)$ are processed. More specifically, suppose that $x[i_x]$ can reach $count_x$ sensor nodes in the part of $T_x$ processed thus far with probability $R_x(i_x, count_x)$. Also, suppose that $y[i_y]$ can reach $count_y$ sensor nodes in $T_y$ with probability $R_y(i_y, count_y)$. Thus, if $x[i_x]$ reaches $y[i_y]$ (i.e., $E_G(x[i_x], y[i_y])$= 1) then $x[i_x]$ can reach a total of $count = count_x + count_y$ nodes.

If $count > n_{req}$ then Step 7 truncates $count$ to $n_{req}$. On the other hand, if $count < count_{x,min}$ (i.e., $count$ is below the minimum number of nodes required to construct an operating state) then Step 9 skips Step 10 and starts a new iteration. Step 10 updates the probabilities accumulated in $R'_x(i_x, count)$.

After exiting the main loop, the current tree $T$ contains only the sink node $s$. Step 12 computes the solution $Conn(G, n_{req})$ from the table $R_s$ associated with the sink $s$.

### D. Correctness

To prove correctness, we first introduce the following notation and definitions. For a given node $x$, and iteration $r \in [1, n-1]$ of the main loop in Step 2, we have the following:

- $DCH(x, r)$: The set of $x$'s deleted children at the start of iteration $r$.
- $V_{x,deleted,r}$ $(= \bigcup_{y \in DCH(x,r)} V_y)$: The set of $x$'s deleted descendants at the start of iteration $r$.

For brevity, we omit $r$ when the iteration number is not important, or understood by the context.

In the following definitions, $x$ is any node in $T$, $i$ is a possible location index of $x$, and $V_{x,delete}$ is the set of deleted descendants associated with $x$ at the start of some iteration.

**[D1]** Let $S$ be a state over nodes in $\{x\} \bigcup V_{x,delete}$. The *type* of $S$ is a pair $(i, count)$ where
- $x$ is at location $x[i]$
- If $count = n_{req}$ then the number of sensor nodes connected to $x[i]$ in $S$ is $\geq n_{req}$
- Else ($count < n_{req}$), then the number of sensor nodes connected to $x[i]$ is $< n_{req}$

**[D2]** We say that table $R_x$ is *complete* with respect to a given set $V_{x,delete}$ if the following conditions hold:
a) For each key $(i, count)$ in $R_x$, $R_x(i, count)$ is the probability of obtaining states over $\{x[i]\} \bigcup V_{x,delete}$ of type $(i, count)$. (Before multiplying by $p_x[i]$, the probability is conditioned on $x$ being at location $x[i]$).
b) Each key $(i, count)$ not in $R_x$ does not contribute to computing the solution $Conn(G, n_{req})$.

We now show the following theorem.

**Theorem** *1:* At the start of each iteration of the main loop in Step 2, if $x$ is a node in the current tree $T$ then table $R_x$ is complete with respect to the associated set $V_{x,delete}$ of deleted nodes.

**Proof.**
**Loop initialization:** At the start of the 1st iteration, $T$ contains all nodes $V$, and each node $x$ has $V_{x,delete} = \emptyset$. Node $x$ in location $x[i_x]$ is associated with one state of type $(i_x, count_x = 0)$ if $x$ is a relay node, or type $(i_x, count_x = 1)$

if $x$ is a sensor node. For each such state type, Step 1 correctly sets $R_x(i, count)$.

**Loop maintenance:** Assume the theorem holds for all possible iterations $r$, where $r \leq n-2$. We show that it holds in iteration $r+1$. Let $y$ be the leaf node deleted in iteration $r$, and $x = parent(y)$. $R_x$ is the only table that may have changed between iterations $r$ and $r+1$. Thus, it suffices to show that $R_x$ is complete with respect to $\{x\} \bigcup V_{x,delete,r+1}$ at the start of iteration $r+1$. To this end, we note the following in iteration $r$:

- Step 4: this step adjusts the probability of each state type $(i, count)$ in $R_y$ by taking into account $p_y[i]$.
- Step 6: this loop exhaustively generates all state types over the set $V_y \bigcup V_{x,delete,r}$ where $V_y$ is all nodes of the subtree rooted at node $y$.
- Step 9: this step discards all state types that can not be extended (by adding sensor nodes from the unprocessed part of the tree) to satisfy the $n_{req}$ requirement.
- Step 10: this step updates the probability of $R'_x(i_x, count)$ by adding the right hand side when states of type $(i_x, count)$ can be extended to operating states.

■

Following an argument similar to the loop maintenance argument, one can show that at Step 12, table $R_s$ associated with the sink node is complete with respect to all nodes $V$ in the network. Thus, the function returns the required solution $Conn(G, n_{req})$.

### *E. Running Time*

Let $n$ be the number of nodes in $G$, and $\ell_{max}$ be the maximum number of locations in the locality set of any node.

**Theorem** *2:* Function Conn solves the S-CONN problem in $O(n \cdot n_{req}^2 \cdot \ell_{max}^2)$ time

**Proof.** We note the following.

- Step 1: storing the tree $T$, and computing $n_x$ and $n_{x,min}$ for each node $x$, require $O(n)$ time.
- Step 2: the main loop performs $n-1$ iterations. Each of Steps 3, 5, and 11 can be done in constant time.
- Step 4: this loop requires $O(n_{req} \cdot \ell_{max})$ time.
- Step 6: this loop requires $O(n_{req}^2 \cdot \ell_{max}^2)$ iterations. Steps 7, 8, 9, and 10 can be done in constant time.

Thus, the overall running time is determined by the main loop that requires $O(n \cdot n_{req}^2 \cdot \ell_{max}^2)$ time. ■

**Theorem** *3:* Function Conn solves the A-CONN problem in $O(n \cdot \ell_{max}^2)$ time.

**Proof.** It suffices to show that the main loop requires the above time. In the A-CONN problem, $n_{req} = |V_{sense}|$, and all sensor nodes in any subtree $T_y$ should be connected to the root $y$ in any operating state. So, in any iteration of the main loop, each table $R_y$ contains keys $(i, count)$ for only one value of $count$ (the maximum value obtainable from descendants of $y$ processed and removed thus far). That is, the maximum length of any table is $\ell_{max}$ independent of $n_{req}$. This gives the running time shown in the theorem. ■

## V. Simulation Results

In this section, we present simulation results that explore the following performance aspects of our devised algorithm:

- the execution time of the algorithm,
- the optimality gap for solving the A-CONN and S-CONN problems, and
- the effect of using relay nodes.

To explore the optimality gap of the devised algorithm, we have implemented an exhaustive algorithm for computing exact solutions. The algorithm works by generating all possible network states of a given probabilistic network. The exhaustive algorithm has a complexity that grows exponentially with the number of nodes in the network. However, it can process a graph with 10, or fewer, nodes in a reasonable time. Both of the exhaustive algorithm, and our devised tree algorithm are implemented in C++ with the use of STL (Standard Template Library) container classes.

**Maximum Spanning Trees.** To use our devised algorithm for obtaining lower bounds on $Conn(G, n_{req})$ of a given network, we need to select a spanning tree. Ideally, one would prefer to use a spanning tree that gives the best possible lower bound. Currently, however, this latter problem appears to be an open problem. In the absence of a known algorithm to compute such an optimum tree, we resort to using a heuristic algorithm. The algorithm works as follows. We associate with each link $(x, y) \in E_G$ a probability, denoted $p(x, y)$, of having the link $(x, y)$ present, given the locality sets $\mathrm{Loc}(x)$ and $\mathrm{Loc}(y)$. We then seek to compute a spanning tree with the highest possible product of link probabilities. This latter problem can be solved efficiently by using a standard minimum spanning tree algorithm. The results discussed utilizes such maximum spanning trees.

**Test Networks.** For simplicity of constructing test networks and analyzing the obtained results, we assume that all nodes have the same transmission range $R_{tr}$, and we set the $E_G$ relation according to the Euclidean distance between the involved nodes.

We have experimented with networks of different sizes in the range [10,20] nodes where each node has a locality set in the range $[2, 8]$ rectangles. Here, we present selected results using the two networks in Fig. 3, denoted $G_{10}$, and Fig. 4, denoted $G_{10,3}$. The selected results are representative of the important findings observed when using other networks. The network $G_{10}$ has 10 sensor nodes, where $v[1]$ is the sink node. As indicated in Fig. 3, node locality sets vary in the range $[3, 6]$ locations where each location is a grid square of unit side length. The figure illustrates a subset of links in $E_G$ that occur when $R_{tr} = 6.5$ units. To avoid cluttering the diagram, we omit the $(x, y)$-coordinates of the locality sets. The network

$G_{10,3}$ adds 3 relay nodes to $G_{10}$. The exhaustive algorithm for computing exact connectivity deals only with the links shown in $G_{10}$ and $G_{10,3}$ (some links may not arise if $R_{tr} < 6.5$). Solid lines in the figures indicate the links used in a maximum spanning tree.

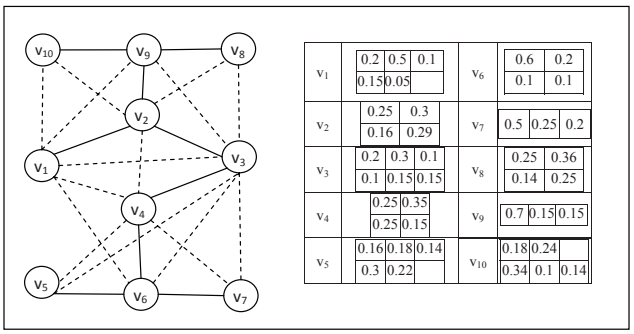


Fig. 3: The $G_{10}$ network

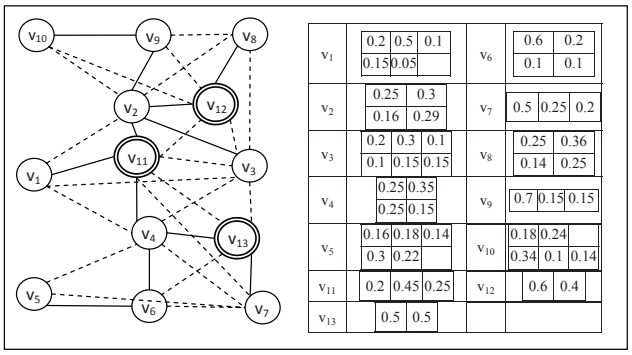


Fig. 4: The $G_{10,3}$ network

**1. Running Time.** The tree algorithm is empirically fast. The running time is typically less than 50 millisec for the tested tree networks of size $\leq 20$ nodes. In contrast, the exhaustive algorithm may require an hour to solve a network of 10 nodes.

**2. Tree bound versus exact solution for the** A-CONN **problem.** Fig. 5 illustrates the obtained results on $G_{10}$ when $R_{tr}$ varies in the range $[3.5, 6.5]$ units. Varying $R_{tr}$ in this range results in topologies with possibly fewer links than shown in $G_{10}$. When $R_{tr} \in [4.5, 6.5]$, the ratio $Conn(G)/Conn(T)$ is found to be in the range $[4.4, 1]$. The ratio decreases monotonically as $R_{tr}$ increases. The above finding suggests that the lower bound is more effective for networks with good overall connectivity probability.

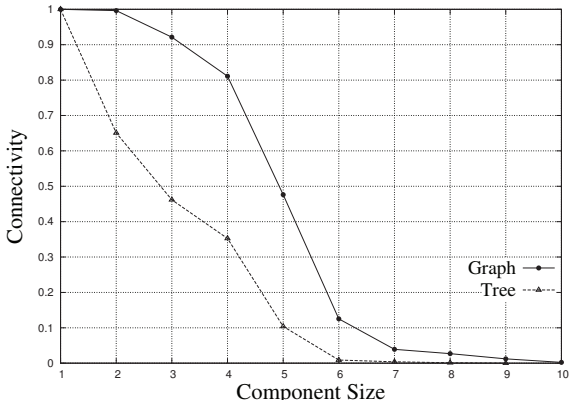


Fig. 6: Connectivity versus $n_{req}$

**3. Tree bound versus exact solution for the** S-CONN **problem.** Fig. 6 illustrates the obtained results on $G_{10}$ when $n_{req}$ varies in the range $[1, 10]$ nodes, and $R_{tr} = 6.5$. The decreasing shape of the curves occur since increasing $n_{req}$ results in larger network states that arise with smaller probability. When $n_{req}$ varies in the range $[2, 5]$, the ratio $Conn(G, n_{req})/Conn(T, n_{req})$ is found to be in the range $[1.5, 4.5]$. The ratio deteriorates (to larger values) as $Conn(G, n_{req})$ takes small values. So, our finding here too is that the tree bound tends to be more useful when the expected connectivity probability is relatively high.

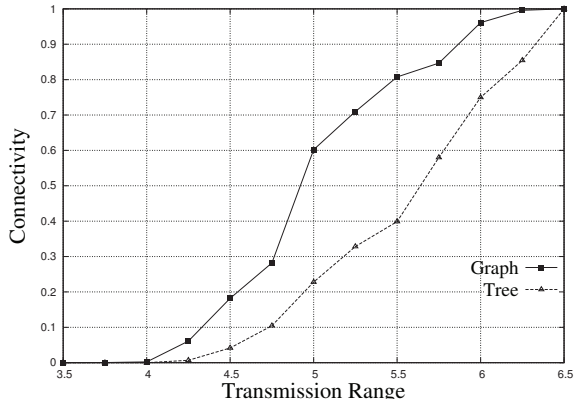


Fig. 5: Connectivity versus transmission range

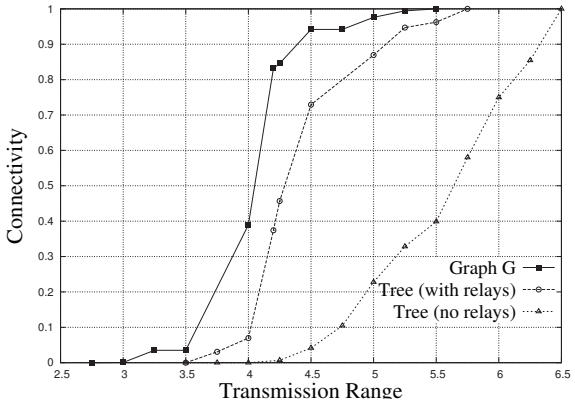


Fig. 7: Effect of using relay nodes

**4. Effect of adding relays.** Deploying relay nodes can potentially increase the number of paths from the sink node to many sensor nodes in the network. Thus, the use of relay nodes can potentially improve the $Conn$ measures. As it turns out, this positive effect is also reflected when using a tree subnetwork. This follows since relay nodes located in close proximity of a number of sensor nodes can provide higher connectivity probability among the nodes than when removing them. Fig. 7 illustrates such positive effect for the A-CONN problem when $R_{tr}$ varies in the range $[2.5, 6.5]$. We have also encountered some unexpected cases where the tree bound $Conn(T)$ of a spanning tree of a network with relays is higher than $Conn(G)$ where $G$ is the corresponding network without relays. Such findings support the positive role of using relays in UWSNs.

## VI. Concluding Remarks

Quantifying the likelihood that a sufficient number of sensor nodes is connected to a sink node in a given UWSN is a challenging problem for networks with semi-mobile and mobile nodes. Our work here adopts a flexible probabilistic graph model to formulate a class of parameterized network connectivity problems. The problem setting allows the network to utilize both sensor nodes and relay nodes. For scenarios where the model is exact, we show that the exact probabilistic connectivity can be computed efficiently for tree-like networks. Our simulation results show the usefulness of the algorithm in computing efficient lower bounds on the solution of any arbitrary network. Scenarios with deployed relay nodes are also analyzed. Future work will consider obtaining stronger bounds of the formulated problems.

## References


[1] I. F. Akyildiz, D. Pompili, and T. Melodia, "Underwater acoustic sensor networks: research challenges," *Ad hoc networks*, vol. 3, no. 3, pp. 257–279, 2005.

[2] J. Heidemann, M. Stojanovic, and M. Zorzi, "Underwater sensor networks: applications, advances and challenges," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 370, no. 1958, pp. 158–175, 2012.

[3] Y. Xiao, *Underwater acoustic sensor networks*. CRC Press, 2010.

[4] R. Otnes, A. Asterjadhi, P. Casari, M. Goetz, T. Husøy, I. Nissen, K. Rimstad, P. van Walree, and M. Zorzi, *Underwater acoustic networking techniques*, ser. SpringerBriefs in Electrical and Computer Engineering. Springer, 2012.

[5] J. Partan, J. Kurose, and B. N. Levine, "A survey of practical issues in underwater networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 11, no. 4, pp. 23–33, 2007.

[6] S. Climent, A. Sanchez, J. V. Capella, N. Meratnia, and J. J. Serrano, "Underwater acoustic wireless sensor networks: Advances and future trends in physical, mac and routing layers," *Sensors*, vol. 14, no. 1, pp. 795–833, 2014.

[7] J. Rice, B. Creber, C. Fletcher, P. Baxley, K. Rogers, K. McDonald, D. Rees, M. Wolf, S. Merriam, R. Mehio, J. Proakis, K. Scussel, D. Porta, J. Baker, J. Hardiman, and D. Green, "Evolution of Seaweb underwater acoustic networking," in *OCEANS 2000 MTS/IEEE Conference and Exhibition*, vol. 3, 2000, pp. 2007–2017.

[8] J. Jaffe and C. Schurgers, "Sensor networks of freely drifting autonomous underwater explorers," in *1st ACM international workshop on Underwater networks*. ACM, 2006, pp. 93–96.

[9] S. Roy, P. Arabshahi, D. Rouseff, and W. Fox, "Wide area ocean networks: Architecture and system design considerations," in *the 1st ACM International Workshop on Underwater Networks*, ser. WUWNet '06. ACM, 2006, pp. 25–32.

[10] J. A. Rice, "US navy Seaweb development," in *Proceedings of the Second Workshop on Underwater Networks*, ser. WuWNet '07. ACM, 2007, pp. 3–4.

[11] L. Pu, Y. Luo, H. Mo, Z. Peng, J.-H. Cui, and Z. Jiang, "Comparing underwater mac protocols in real sea experiment," in *IFIP Networking Conference, 2013*. IEEE, 2013, pp. 1–9.

[12] A. Bower and T. Rossby, "Evidence of cross-frontal exchange processes in the gulf stream based on isopycnal rafos float data," *Journal of Physical Oceanography*, vol. 19, no. 9, pp. 1177–1190, 1989.

[13] A. S. Bower, "A simple kinematic mechanism for mixing fluid parcels across a meandering jet," *Journal of Physical Oceanography*, vol. 21, no. 1, pp. 173–180, 1991.

[14] A. Caruso, F. Paparella, L. F. M. Vieira, M. Erol, and M. Gerla, "The meandering current mobility model and its impact on underwater mobile sensor networks," in *INFOCOM 2008*. IEEE, 2008, pp. 221–225.

[15] J. Luo, D. Wang, and Q. Zhang, "Double mobility: coverage of the sea surface with mobile sensor networks," in *IEEE INFOCOM 2009*. IEEE, 2009, pp. 118–126.

[16] E. Elmallah, H. Hassanein, and H. AboElFotoh, "Supporting QoS routing in mobile ad hoc networks using probabilistic locality and load balancing," in *Proceedings of the IEEE GLOBECOM Symposium on Ad Hoc Wireless Networks, San Antonio*, vol. 5, 2001, pp. 2901–2906.

[17] C. J. Colbourn, *The Combinatorics of Network Reliability*. Oxford University Press, 1987.