Abstract

Underwater Wireless Sensor Networks (UWSNs) are the subject of interest for many research due to the inherited uncertainty of the underwater environment. Communication in UWSNs is one field that offers great opportunities while suffering from many challenges. In this study, we discuss the connectivity between sensor nodes in an UWSN. Specifically, we investigate the probability that two nodes floating freely with the water currents can communicate with each other. We refer to this problem as the Two-Nodes connectivity ($2Nodes$ connectivity) problem. We emphasise the computational complexity of the problem (i.e., $\#P-Hard$) and propose an approximation algorithm to the problem with polynomial running time. Our algorithm returns the exact connectivity (i.e., an exact solution) between any two nodes, given that the input graph is composed of node-disjoint paths. The exact solution is also a lower bound solution to any type of graphs where node-disjoint paths can be extracted. We evaluate our algorithm using simulations based on realistic UWSN scenarios and compare it with an optimal solution. We demonstrate that our algorithm is remarkably efficient, making it an ideal choice for time-sensitive applications.

Efficient Connectivity Analysis in Underwater Wireless Sensor Networks: A Polynomial-Time Solution for the Connectivity between Nodes

Youssef Altherwy

November 2023

1 Introduction

An Underwater Wireless Sensor Network (UWSN) constitutes a distributed system of sensor nodes strategically positioned across diverse aquatic environments, such as oceans and seas, to perform a spectrum of tasks encompassing the sensing of various water characteristics and facilitating data communication [XH20; HKA20]. The nodes within this network can assume static, semi-static, or fully mobile configurations, with the latter being the focal point of this investigation—a study centered on mobile nodes that traverse freely with the prevailing water currents [HSZ12]. The inherent mobility of these nodes introduces formidable challenges to connectivity, as the fluid nature of their movement may result in intermittent disconnections between them. Consequently, this study undertakes the critical inquiry: What is the probability that two mobile nodes remain connected in a dynamic UWSN environment? This specific challenge is formalized as the Two-Nodes connectivity ($2Nodes$ connectivity) problem, wherein the study analyzes the likelihood of establishing and maintaining a connection between two nodes within the UWSN framework, quantifying the probability that node x reaches node y through a multi-hop path.

To tackle the $2Nodes$ connectivity problem, this research leverages simulations of underwater sensor networks, applying methodologies articulated in prior works [Car+08; Bou+19]. The outcomes of these simulations contribute to the construction of a dataset, facilitating the evaluation of the proposed algorithm's efficacy in addressing the $2Nodes$ connectivity problem.

The primary contribution and novelty of this study lie in its dedicated focus on addressing the communication likelihood between two nodes in a UWSN, encapsulated in the $2Nodes$ connectivity problem. As elucidated in §3, this problem is inherently $\#P$-Hard, underscoring its computational complexity. The distinctive contribution of this research is the development of an algorithm for the $2Nodes$ connectivity problem that operates within polynomial time complexity, thereby surmounting the computational challenges associated with this intricate problem.

The remainder of this paper is organized as follows: §2 provides an overview of related work, delving into existing research pertinent to underwater sensor networks and connectivity challenges. §3 introduces the novel algorithm designed to address the $2Nodes$ connectivity problem. The subsequent section, §4, critically evaluates the algorithm's performance through comprehensive analysis, and the study concludes with key insights and implications presented in §5.

2 Literature Review

Underwater Wireless Sensor Networks (UWSNs) have attracted substantial attention for their pivotal role in monitoring and collecting data from underwater environments. The deployment of sensor nodes in aquatic settings introduces unique challenges, and recent research has addressed various facets of UWSNs to enhance their efficiency and reliability. In this literature review, we first examine recent work in the area of UWSNs [Jou+19; Awa+19; SS22; XHW20; Zhu+22], then we discuss work that utilize node-disjoint paths in networks and communications [Lu+22; Wu+22; Ker+22; HV20; YCH14]. We conclude the literature review by discussing studies similar to ours [AEE20; AEE18; Zan+20] and studies we employ to build the dataset for simulation [Car+08; Bou+19].

2.1 Recent Work in UWSNs

In [Jou+19], the authors comprehensively review challenges facing UWSNs, including energy harvesting and underwater localization. They delve into key enabling technologies such as radio frequency, optical, and acoustic communication, and discuss state-of-the-art localization protocols. The concept of the Internet of Underwater Things (IoUT) is introduced, providing insights into integration with terrestrial networks and cloud computing. Open issues and future research directions for UWSNs and IoUT are identified. Similarly, [Awa+19] focuses on recent issues and challenges in UWSNs, covering network architecture, medium access control, routing protocols, data aggregation and fusion, security, privacy, and quality of service. The paper also presents existing solutions and applications, offering insights into the current state of UWSN research.

Additionally, Shovon et al. [SS22] provide a survey of multi-path routing protocols for UWSNs, categorizing them into energy, geographical, and data centered routing protocols. The study in [XHW20] introduces a clustering routing protocol tailored for Underwater Wireless Sensor Networks (UWSNs). Emphasizing energy efficiency, the protocol utilizes data fusion and genetic algorithms to optimize communication and data aggregation. Through the formation of clusters and the application of genetic algorithms, the proposed protocol strives to improve the overall energy efficiency of UWSNs, addressing the specific challenges presented by the underwater environment. We include the work in [Zhu+22] to enrich the discussion on recent advancements in UWSN routing protocols.

2.2 Node-Disjoint Paths in Networks and Communications

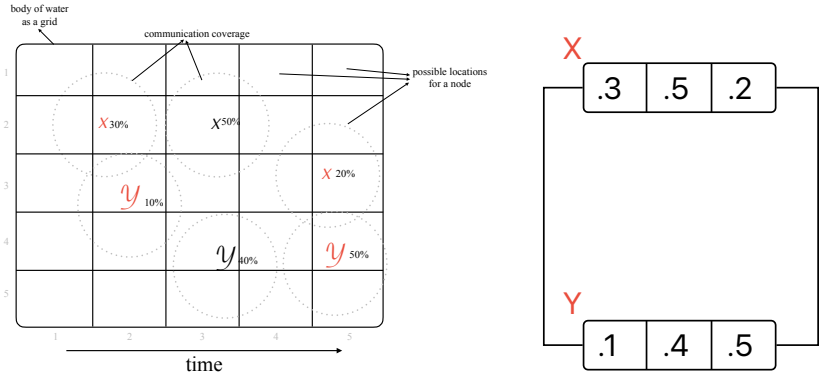
Since this paper discusses node-disjoint paths, we explore related work in networks and communications that employ node-disjoint paths. Lu et al. [Lu+22] design algorithms for BCube Connected Crossbars (BCCC) networks, returning fault-free paths between two nodes using node-disjoint paths. Wu et al. [Wu+22] develop an algorithm for node-disjoint paths in dragonfly networks. Kern et al. [Ker+22] propose algorithms for node-disjoint paths on H-free graphs with multiple nodes and edges. Additionally, Hadid et al. [HV20] devise a self-stabilizing node-disjoint paths routing algorithm in star networks. Finally, the work in [YCH14] proposes a robust node-disjoint multipath routing scheme tailored for wireless sensor networks. Emphasizing resilience, the protocol aims to enhance reliability by establishing node-disjoint paths. According to the authors, the work contributes to improved fault tolerance and communication robustness in wireless sensor deployments.

2.3 Studies Similar to Ours

Next, we explore recent studies that share similarities with our work. In [AEE20], the authors propose an approach to finding the likelihood that a Wireless Sensor Network (WSN) is in a connected state, considering networks with multiple states or locations, akin to UWSNs where nodes float freely with water currents. In [AEE18], the largest connected component problem in UWSNs is studied, assuming a probabilistic communication model. The proposed dynamic programming algorithm addresses network complexities and compares favorably with existing methods. We integrate the insights from [Zan+20] to contribute to the discussion on probabilistic communication models. Concluding this section, we turn our attention to the pivotal work in [Car+08; Bou+19], essential for building the dataset to evaluate the performance of our solution in solving the Two-Nodes connectivity ($2Nodes$ connectivity) problem. The Meandering Current Mobility Model (MCM) proposed in [Car+08] captures realistic movement patterns of underwater sensors influenced by ocean currents and water turbulence. Evaluating the impact of MCM on routing protocols, the study concludes that MCM provides more realistic and challenging scenarios for underwater network simulation. An extension of this work, [Bou+19], considers additional forces affecting underwater sensor node movement, crucial for creating a comprehensive dataset for our study. Both works are fundamental to building our dataset, as detailed in §4.1.

3 Proposed Model

In this section, we discuss the methodology behind our algorithm to solve the Two-Nodes connectivity ($2Nodes$ connectivity) problem. But, first, we introduce the notion of *probabilistic graphs*. Earlier in §1, we mentioned that nodes



(a) (b)

Figure 1: A hypothetical scenario where the body of water is visualized as a grid (figure 1a). Nodes x and y reside in different locations (cells) with different probabilities (figure 1b). The nodes in red are the connected nodes.

move uncontrollably with the water currents. By product, the location of any node at any given time is probabilistic. In other words, the location of a node x at a time t can only be defined with a certain probability (p). Accordingly, node x can reside in different locations at different times with different probabilities. Thus, x has a set of possible locations, which we refer to as a *locality* set (i.e., Loc_x).

Each node $x \in V$ in an UWSN has a locality set, where V is the collection of all nodes. Those locality sets dictate the communication links between the nodes. To illustrate, nodes x and y can reside in multiple locations at different times, as explained earlier. In some of those locations, x and y can reach each other (i.e., connected). On the other locations, however, x and y cannot reach each other (i.e., unconnected). Based on those connected and unconnected states, we can establish a connection table ($E_{\{x,y\}}$) between x and y. For example, $E_{\{x,y\}}[i,j] = 1$ if x at location i reaches y at location j. However, if x at location i cannot reach y at location j, then $E_{\{x,y\}}[i,j] = 0$. Thus, the total number of entries in $E_{\{x,y\}} = |Loc_x| * |Loc_y|$ where $|Loc_x|$ and $|Loc_y|$ are the number of locations for nodes x and y, respectively. Figure 1a shows a hypothetical scenario where nodes x and y both reside in three different locations. In the figure, x is connected to y if x is at location [2, 2] and y is at location [2, 3]. Also, when x is at location [5, 3] and y is at location [5, 4],. Thus, we populate table $E_{\{x,y\}}$ as in table 1.

To create the locality set for each node, we visualize the body of water as a grid. The cells of the grid represent all possible locations for nodes. For example, figure 1a assumes that node x occupies locations [2, 2] 30% of the time, [3, 2] 50% of the time, and [5, 3] the rest of the time. Whereas, node y occupies

x	y	connected
[2,2]	[2,3]	1
[2,2]	[3,4]	0
[2,2]	[5,4]	0
[3,2]	[2,3]	0
[3,2]	[3,4]	0
[3,2]	[5,4]	0
[5,3]	[2,3]	0
[5,3]	[3,4]	0
[5,3]	[5,4]	1

Table 1: $E_{x,y}$ table

locations [2, 3] 10% of the time, [3, 4] 40% of the time, and [5, 4] the rest of the time. By time, we mean the time we simulate or observe the nodes. Therefore, $p(Loc_x) = \{[2,2] : 0.3, [3,2] : 0.5, [5,3] : 0.2\}$ and $p(Loc_y) = \{[2,3] : 0.1, [3,4] : 0.4, [5,4] : 0.5\}$. For simplicity, we write $p(Loc_x) = \{i : 0.3, j : 0.5, k : 0.2\}$ and $p(Loc_y) = \{i : 0.1, j : 0.4, k : 0.5\}$ where $\{i, j, k\}$ refer to the different cells in the grid.

Because of the locality sets, an UWSN has multiple *states* w.r.t. time. By a state, we mean an UWSN where each node resides in one location of their respective locality set at a specific time. The occurrence of a state (S) is probabilistic since locality sets are probabilistic by nature. Thus, the multiple states and their probabilistic nature give rise to the notion of *probabilistic graphs* (G). To represent the example in figure 1a as a probabilistic graph, we refer to figure 1b. The links in the figure represent the connection between x and y as explained earlier in table 1. Figures 2,3a, and 3b further illustrate the notion of probabilistic graphs and states. The figure shows a probabilistic graph (figure 2) and two of its states (figures 3a and 3b). Figure 3a illustrates a working state because there exist a path from s to t. Figure 3b, on the other hand, does not include a path between x and y, which illustrates a failed state.

Now, we formally define a probabilistic graph as $G = \{V, Loc, E, P\}$ where V is the collection of all nodes in an UWSN, Loc represents the locality sets of those nodes, and P is the probabilities associated with those locality sets. E represents the communication tables between the nodes and is equal to $\frac{V*(V-1)}{2}$ tables. To solve the $2Nodes$ connectivity problem, we need to compute the occurrence probability of each state $S_i \in G$ (i.e., Eq.1) and then add the results (i.e., Eq.2). In Eq.1, x is a node that belongs to state S_i and resides at some location $[i]$. In Eq.2, N_s is the total number of states in G.

$$P(S_i) = \prod_{x \in S_i} Loc_x[i] \tag{1}$$

$$P(G) = \sum_{S_i \in G}^{N_s} P(S_i) \tag{2}$$

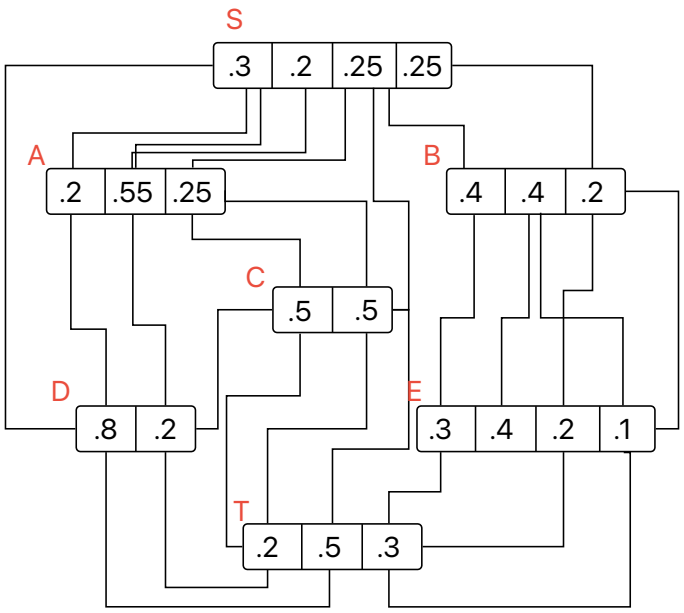
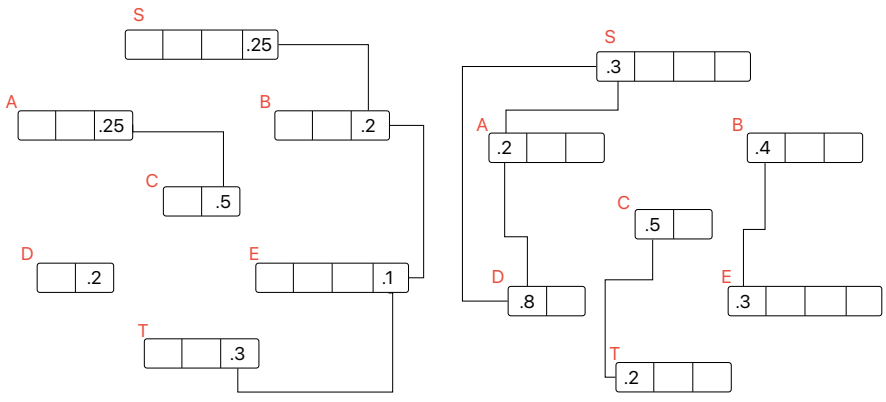


Figure 2: An example of a probabilistic graph.



(a) A working state

(b) A failed state

Figure 3: Working and failed states from the probabilistic graph in figure 2.

The total number of states (N_s) is calculated as in Eq.3 where $|Loc_x|$ is the total number of locations for a node x. For instance, in figure 2, the total number of states is 1728 according to Eq.3.

$$N_s = \prod_{x \in G=V} |Loc_x| \tag{3}$$

When Loc_x grows for $x \in V$, we can argue that N_s grows exponentially, as does the running time for computing $P(G)$. To explain, we refer to the *2-terminal network reliability* problem discussed in [Col91]. In [Col91], the authors discuss the reliability of a network by examining the operation state of the nodes between two target nodes. They adopt a similar notion to probabilistic graphs, where nodes can be in a failed or operating state. This resembles our concept of locality sets, where, in this case, each node has a locality set with only two entries: failed and operating. The authors showed that the 2-terminal network reliability problem is $\#P-hard$. The Two-Nodes connectivity ($2Nodes$ connectivity) problem we discuss here shares the same complexity as the 2-terminal network reliability problem since both problems have similar concepts of locality sets and, by extension, probabilistic graphs. Furthermore, the locality sets in our problem could even have more entries than two. Thus, the Two-Nodes connectivity ($2Nodes$ connectivity) problem is $\#P - hard$.

As we shall explain, we can solve the $2Nodes$ connectivity problem in polynomial time if the input graph is composed of node-disjoint paths. A node-disjoint path is a path where each node between the two target nodes (e.g., s and t) is connected to exactly two nodes, the nodes before and after. Figure 4 shows three node-disjoint paths between two target nodes (s, t): the first one (in red) is {S,A,D,T}, the second one (in blue) is {S,C,T}, and the last one (in green) is {S,B,E,T}. We can employ any number of algorithms to retrieve the node-disjoint paths from an input graph. In this study we employ the Edmonds-Karp algorithm. The Edmonds-Karp algorithm is used to find the maximum flow through a graph from a source to a destination, subject to the capacity constraints of the graph links [FF56]. Here, we set the capacity constraints of the links to one, as we are only interested in finding node-disjoint paths between the two terminal nodes. Figure 4 shows the resulted node-disjoint paths using the Edmonds-Karp algorithm. To avoid any misunderstanding, we refer to a graph of node-disjoint paths as H instead of G, where $H \subseteq G$. Thus, an exact solution for H can also serve as a lower bound solution for G. The next section discusses how we compute an exact solution for $2Nodes$ connectivity when the input graph is H.

3.1 An exact solution for the $2Nodes$ connectivity problem

To solve the $2Nodes$ connectivity problem for a graph H, we assume that s and t are the two target nodes in the graph. The node-disjoint paths in H are $dp \in \text{DP}$, where DP is the maximum number of node-disjoint paths. We compute the node-disjoint path connectivity table (T^{dp}_{prob}) for each $dp \in \text{DP}$ where the number of $T_{\text{prob}} = \text{DP}$. To do so, we assume that each dp is composed

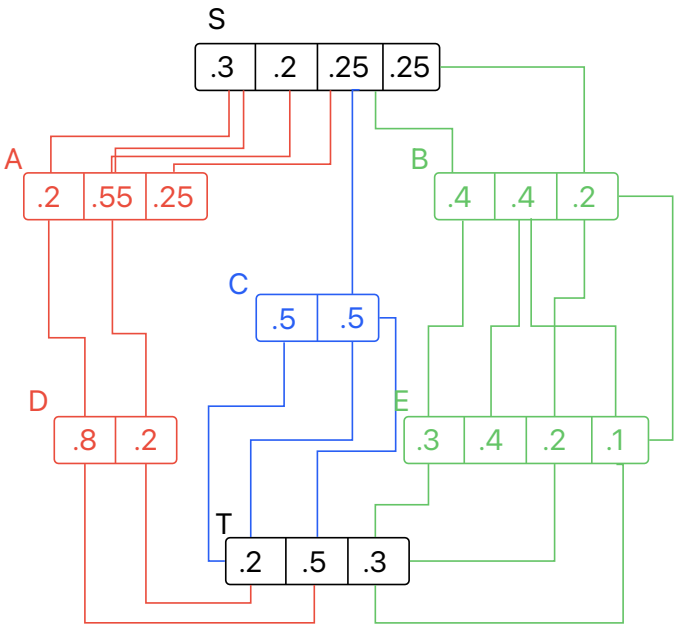


Figure 4: The node-disjoint paths graph H from the probabilistic graph G (figure 2)

of the nodes $\{s, 1st, 2nd, ..., lst, t\}$. For now, we ignore s and t and focus on the other nodes. For any neighboring nodes x and y in dp, we compute the connectivity between the nodes as in Eq. 4 and 5. In Eq. 5, we replace Loc_y with the newly computed Loc_z and use that in the connectivity computation of subsequent nodes.

$$p(Loc_z[i,j]) = \sum_{i \in Loc_x, j \in Loc_y} p(Loc_x[i]).p(Loc_y[j]).E_{\{x,y\}}[i,j] \tag{4}$$

$$Loc_y = Loc_z \tag{5}$$

Then, for each dp, $T^{dp}_{\text{prob}} = Loc_z[i,j]$ from Eq.5, where $i \in 1st$ and $j \in lst$ nodes. Afterwards, we compute the probability ($prob^{dp}$) that s at location i ($Loc_s[i]$) reaches t at location j ($Loc_t[j]$) through a path $dp \in$ DP as in Eq.6. $1st$ and lst nodes are the first and last nodes in a node-disjoint path (dp), respectively.

$$p\Big(prob^{dp}(Loc_s[i], Loc_t[j])\Big) = \sum \left(p\Big(T^{dp}_{\text{prob}}(Loc^{dp}_{1st}[a], Loc^{dp}_{lst}[b])\Big).E_{\{s,1st\}}[i,a].E_{\{lst,t\}}[b,j] \right) \tag{6}$$

From Eq.6, we compute the probability ($prob$) that $Loc_s[i]$ reaches $Loc_t[j]$ through at least one $dp \in$ DP as in Eq.7

$$p\Big(prob(Loc_s[i], Loc_t[j])\Big) = p(Loc_s[i]).p(Loc_t[j]).\left(1 - \prod_{dp}^{\text{DP}} \Big(1 - p\Big(prob^{dp}(Loc_s[i], Loc_t[j])\Big)\Big)\right) \tag{7}$$

Finally, we solve the $2Nodes$ connectivity problem by rewriting Eq.2 to become Eq.8.

$$P(H) = \sum_{i \in Loc_s, j \in Loc_t} p\Big(prob(Loc_s[i], Loc_t[j])\Big) \tag{8}$$

In the next section, we prove that Eq.8 is an exact solution for H and a lower bound solution for G with a polynomial running time.

3.2 The Two-Nodes connectivity solution's running time

Algorithm 1 returns the node-disjoint path connectivity table T^{dp}_{prob} for each $dp \in DP$. Using T^{dp}_{prob}, algorithm 2 returns the $prob^{dp}$ table for each $dp \in DP$. Then, algorithm 3 returns $prob$ table, from which the $2Nodes$ connectivity problem is solved as in algorithm 4. Now, we show the running time for each algorithm to prove that the exact solution for the $2Nodes$ connectivity problem runs on polynomial time if the input graph is composed of node-disjoint paths (H).

Algorithm 1 running time is $O\left[DP.\left(\Big(r-1\Big).(LOC^3_{max})\right) + LOC^2_{max}\right]$. Specifically, the running time for the *for* loop on lines 5 to 9 is $|Loc_{1st}| \times |Loc_{2nd}|$.

Algorithm 1 The node-disjoint path connectivity table (T_{prob}). Eq.4,5

Require: DP

for $dp \in DP$ **do**
  for two subsequent nodes x and $y \in dp$ **do**
    $Loc_z \leftarrow \{\}$ ▷ where the number of entries in $Loc_z = |Loc_x|.|Loc_y|$
    if $x = 1st$ and $y = 2nd$ **then** ▷ the first two nodes in dp
      for $i \in Loc_x$ and $j \in Loc_y$ **do**
        if $E_{\{x,y\}}[i,j] == 1$ **then**
          $p(Loc_z[i,j])$ $+=\Big(p(Loc_x[i]).p(Loc_y[j])\Big)$
        end if
      end for
    else if $x \neq 1st$ and $y \neq 2nd$ **then**
      for $[i,j] \in Loc_x$ and $k \in Loc_y$ **do**
        if $E_{\{x,y\}}[j,k] == 1$ **then**
          $p(Loc_z[i,k])$ $+=\Big(p(Loc_x[i,j]).p(Loc_y[k])\Big)$
        end if
      end for
    end if
    $Loc_y \leftarrow Loc_z$
  end for
  $T^{dp}_{\text{prob}} \leftarrow Loc_z$
end for
Return T_{prob} tables

If we denote the maximum locality set for any given node as LOC_{max}, then the running time for this for loop is $O(LOC^2_{max})$. For lines 11 to 15, the for loop has a running time of $|LOC_{max}| \times |LOC^2_{max}|$ where $|LOC^2_{max}|$ is the result of combining the locality sets of two nodes as explained earlier. Thus, the running time is $O(LOC^3_{max})$. The running time for lines 2 to 18 is $O\Bigg(\Big((r-1).LOC^3_{max}\Big) + LOC^2_{max}\Bigg)$ where r is the number of nodes in a node-disjoint path excluding s and t. Finally, for lines 1 to 20 and the algorithm, the running time is $O\Bigg[DP.\Bigg(\Big(r-1\Big).(LOC^3_{max})\Bigg) + LOC^2_{max}\Bigg]$ where DP is the number of node-disjoint paths.

Algorithm 2 The probability ($prob^{dp}$) tables. Eq.6

Require: Loc_s, Loc_t, and T_{prob} tables
1: **for** $dp \in DP$ **do**
2: **for** $i \in Loc_s$ and $j \in Loc_t$ **do**
3: $p\Big(prob^{dp}(Loc_s[i], Loc_t[j])\Big) \leftarrow 0$
4: **for** $[a, b] \in T^{dp}_{\text{prob}}$ **do** ▷ where $a \in Loc_{1st}$ and $b \in Loc_{lst}$
5: **if** $E_{\{s,T^{dp}_{\text{prob}}\}}[i, a] == 1$ and $E_{\{T^{dp}_{\text{prob}},t\}}[b, j] == 1$ **then**
6: $p\Big(prob^{dp}(Loc_s[i], Loc_t[j])\Big)$ $+= p\Big(T^{dp}_{\text{prob}}(Loc^{dp}_{1st}[a], Loc^{dp}_{lst}[b])\Big)$
7: **end if**
8: **end for**
9: **end for**
10: **end for**
11: **Return** $prob^{dp}$ tables

For algorithm 2, the running time is $O(DP.LOC^4_{max})$. In details, the running time for the loop on lines 4 to 8 is $O(LOC^2_{max})$ because $|T^{dp}_{\text{prob}}| = |Loc_{1st}| \times |Loc_{lst}|$. The running time for the loop on lines 2 to 9 is $|Loc_s| \times |Loc_t| \times LOC^2_{max}$, in other words, $O(LOC^4_{max})$. Thus, the running time for lines 1 to 10 and the algorithm is $O(DP.LOC^4_{max})$.

The running time for algorithm 3 is $O(DP.LOC^2_{max})$. The running time for the loop on lines 3 to 5 is $O(DP)$, whereas the running time for the loop on lines 1 to 7 is $O(LOC^2_{max})$ because $|Loc_s| \times |Loc_t|$. Thus, the running time for the algorithm is $O(DP.LOC^2_{max})$.

Finally, algorithm 4 combines the three algorithm introduced earlier and returns an exact solution for the $2Nodes$ connectivity problem. As we discussed earlier, the running time for the three algorithms is $O\Bigg[DP.\Bigg(\Big(r-1\Big).(LOC^3_{max})\Bigg) +$

Algorithm 3 The probability (*prob*) table. Eq.7

Require: Loc_s, Loc_t, and $prob^{dp}$ tables

for $i \in Loc_s$ and $j \in Loc_t$ **do**
    temp $\leftarrow 1$
    for $dp \in DP$ **do**
        temp $\times = 1 - p\Big(prob^{dp}(Loc_s[i], Loc_t[j])\Big)$
    end for
    $p\Big(prob(Loc_s[i], Loc_t[j])\Big) = p(Loc_s[i]).p(Loc_t[j]).(1 - \text{temp})$
end for
Return *prob* table

Algorithm 4 The $2Nodes$ connectivity solution. Eq.8

Require: Loc_s, Loc_t, and DP

T_{prob} tables $\leftarrow$ Algorithm1(DP) ▷ returns T_{prob} tables
$prob^{dp}$ tables $\leftarrow$ Algorithm2($Loc_s, Loc_t, T_{\text{prob}}$ tables) ▷ returns $prob^{dp}$ tables
prob tables $\leftarrow$ Algorithm3($Loc_s, Loc_t, prob^{dp}$ tables) ▷ returns *prob* tables
$2Nodes$ connectivity $\leftarrow 0$
for $i \in Loc_s$ and $j \in Loc_t$ **do**
    $2Nodes$ connectivity $+= p\Big(prob(Loc_s[i], Loc_t[j])\Big)$
end for
Return $2Nodes$ connectivity

$LOC^2_{max}\Big]$, $O(DP.LOC^4_{max})$, and $O(DP.LOC^2_{max})$, respectively. Besides those algorithms, algorithm 4 has one for loop (i.e., lines 5 to 7) that has a running time of $O(LOC^2_{max})$ because $|Loc_s| \times |Loc_t|$. Thus, the running time for the algorithm is $O\Bigg[DP.\Bigg(\Big(r-1\Big).(LOC^3_{max})\Bigg) + LOC^2_{max} + (DP.LOC^4_{max}) + (DP.LOC^2_{max}) + LOC^2_{max}\Bigg]$. Alternatively, $O\Bigg(DP.\Big(((r-1).LOC^3_{max}) + LOC^4_{max} + LOC^2_{max}\Big) + 2LOC^2_{max}\Bigg)$. For simplicity, we denote the running time for algorithm 4 as $O(DP.LOC^4_{max})$. Thus, the solution for the $2Nodes$ connectivity problem has a polynomial running time if the input graph is composed of node-disjoint paths (H).

Next, we discuss, in details, how we simulate and build the dataset to evaluate the performance of the $2Nodes$ connectivity solution.

4 Results and Discussion

Here, we implement the proposed model in §3 to compute the solution for the Two-Nodes connectivity ($2Nodes$ connectivity) problem for a graph composed of node-disjoint paths. We evaluate our model using graphs obtained from datasets, which we build from the work presented in [Car+08; Bou+19]. In the next section, we discuss the dataset.

4.1 The Dataset

As mentioned in §2, the work in [Car+08; Bou+19] present mobility models of underwater sensor nodes by studying different forces that affect that mobility. In building the dataset, the initial positions of nodes are chosen randomly, and the constant parameters in the mobility model are chosen according to the work in [Bou+19]. Based on the initial positions and the constant values, the movements of nodes are simulated according to the mobility model. Table 2 shows the chosen initial positions as well as the constant parameters from [Bou+19]. The explanation of the parameters are found in [Bou+19; Zho+11; NKK14]. We implement a python script to build the dataset. The script populates the dataset by running simulations for different number of nodes and locality sets. The simulations are run on an iMac Pro with a 3.2 GHz 8-Core Intel Xeon W processor and 32 GB of RAM. The simulation inputs are the number of nodes (V) and the maximum locality set for any node in the graph (LOC_{max}). The simulation results, on the other hand, are the running time and the connectivity.

We compute the connectivity for our solution and the exact solution. Recall that our solution for the $2Nodes$ connectivity problem generates an exact solution for a graph composed of node-disjoint paths (H). It is also a lower

Parameter	Value	Parameter	Value
k1	0.3π	v	0.3
k2	π	V_s	0.5
k3	2π	P_w	1025
k4	1	x initial location	-1×10^3 to 1×10^3
k5	1	y initial location	-0.5×10^3 to 0.5×10^3
λ	1	z initial location	-500 to -100

Table 2: Simulation parameters and initial x,y,z locations

bound solution for a graph G where $H \subseteq G$. To get the exact solution for G, we generate every possible state of G as mentioned in §3. This results in a solution with a running time that grows exponentially with the number of nodes. Specifically, the time complexity of the exact solution is $O(LOC_{max}^{V})$. The time complexity of our algorithm, as mentioned in §3, is $O(DP.LOC_{max}^{4})$ where DP is the number of node-disjoint paths. Because of the exponential nature of the exact solution, we were able to compute the exact connectivity for simulations up to $\{V = 14, LOC_{max} = 3\}$. In fact, this results in $3^{14} = 4,782,969$ states. Algorithm 5 shows our method to compute the exact solution.

Algorithm 5 Exact Solution. Eq1

Require: *node_id*, *path*, *prob*
1: *node* ← get the node from the nodes list using *node_id*
2: *loc* ← get the location of the node from the locations dictionary
3: *paths* ← empty list
4: **for** each *location* in *locations* **do**
5: add *location* to *path*
6: multiply *prob* with the location probability
7: **if** *node* is not the destination **then**
8: *path*, *prob* ← call *exact_solution* with the next *node_id*, current *path*, and updated *prob*
9: remove the last *location* from *path*
10: divide *prob* by the location probability
11: **else**
12: *new_path* ← copy of *path* and add *prob* to it
13: add *new_path* to list of paths
14: remove the last *location* from *path*
15: divide *prob* by the location probability
16: **end if**
17: **end for**
18: **Return** *path* and *prob*

Figure 5 shows the number of simulations with respect to V and LOC_{max}. Figure 6 shows the distribution of the data points in the dataset. With

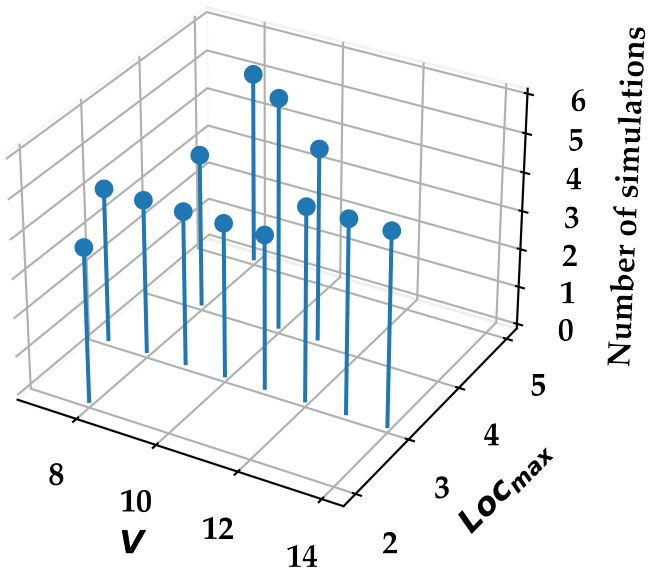
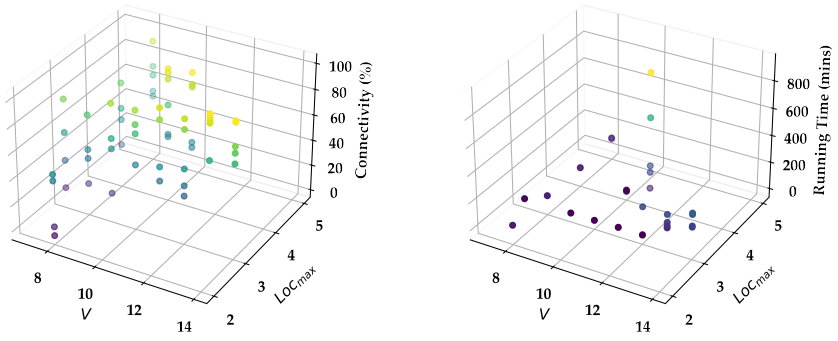


Figure 5: The number of simulations with respect to V and LOC_{max}

.



(a) (b)

Figure 6: The data points for both (a) the connectivity (%) and (b) the running time (in minutes) with respect to V and LOC_{max}.

respect to V and LOC_{max}, **(a)** shows the connectivity (%) and **(b)** shows the running time in minutes.

4.2 The Results

From the dataset, we evaluate the performance of our algorithm from §3 in comparison to the exact solution. We refer to our algorithm as the *lower bound* solution, and the exact algorithm as the *exact* solution. In comparing the two solutions, we discuss the following categories:

(i) The connectivity of the exact and lower bound solutions w.r.t. V (category 4.3).

(ii) The running time of the exact and lower bound solutions w.r.t. V (category 4.4).

(iii) The connectivity of the exact and lower bound solutions w.r.t. LOC_{max} (category 4.5).

(iv) The running time of the exact and lower bound solutions w.r.t. LOC_{max} (category 4.6).

Due to the random nature of initial node positions, as mentioned earlier, we repeat the simulation for each category multiple times and report the average value and the standard deviation. The standard deviation values are represented in the figures as error bars.

4.3 Connectivity of the exact and lower bound solutions w.r.t. V.

In this category, we examine the connectivity (%) when $V = \{7, 8, ..., 14\}$ and $LOC_{max} = 3$. Figure 7 reports the results. From the figure, we notice that the lower bound solution has a very similar pattern to the exact solution. When the exact solution reports high connectivity, the lower bound solution also reports high connectivity. This resemblance in pattern is an important feature to have in a lower bound solution as it can provide a useful indication of network connectivity without the expensive computation associated with the exact solution. In fact, recall that 14 is chosen as the maximum number of nodes in a graph because it becomes infeasible to run the exact solution on our machine when $V > 14$. We further discuss the running time in category 4.4.

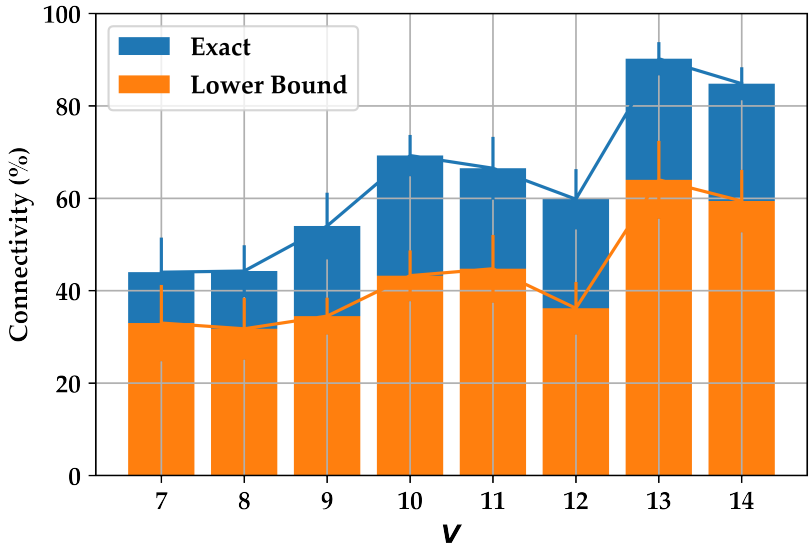


Figure 7: The connectivity of the exact and lower bound solutions when $V = \{7, 8, ..., 14\}$ and $LOC_{max} = 3$

With respect to connectivity, the results in figure 7 indicate an upward trajectory in connectivity associated with the increase of V. This, we believe, is the expected behaviour as increasing V usually increase the chances of S reaching t.

4.4 Running time of the exact and lower bound solutions w.r.t. V.

This category discusses the running time difference (in minutes) between the exact and lower bound solution with the same V and LOC_{max} as category 4.3. Figure 8 proves that the exact solution grows exponentially as V grows, which is the expected behaviour as explained earlier. The figure also proves another assumption we made earlier, which is the polynomial running time of the lower bound solution.

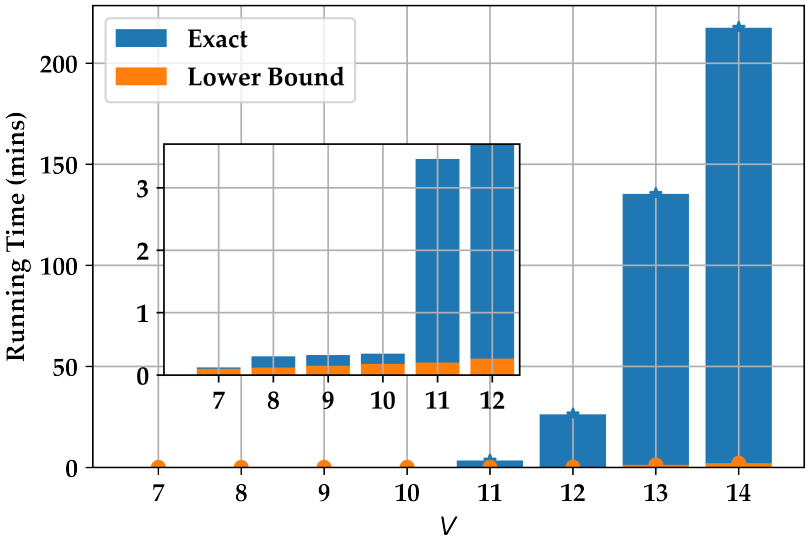


Figure 8: The running time (in minutes) of the exact and lower bound solutions when $V = \{7, 8, ..., 14\}$ and $LOC_{max} = 3$. The smaller figure shows a zoomed in part of the original figure.

In fact, the lower bound solution has a running time of less than three minute, in spite of the number of nodes. However, the exact solution takes approximately two hours to be returned when $V = 13$, and two more hours (a total of four hours) when the number of nodes is only increased by one (a total of 14 nodes). Put differently, the former represents a 125x increase in the running time compared to the lower bound solution, and the latter represents a 222x increase! This emphasizes the efficiency and usefulness of the lower bound solution. Furthermore, the significant increase in running time only ensures less than 30% increase in accuracy, according to figure 9. Therefore, the lower bound solution serves as an optimal candidate for applications where a fast and reasonably accurate solution is more important than an accurate but extremely slow solution.

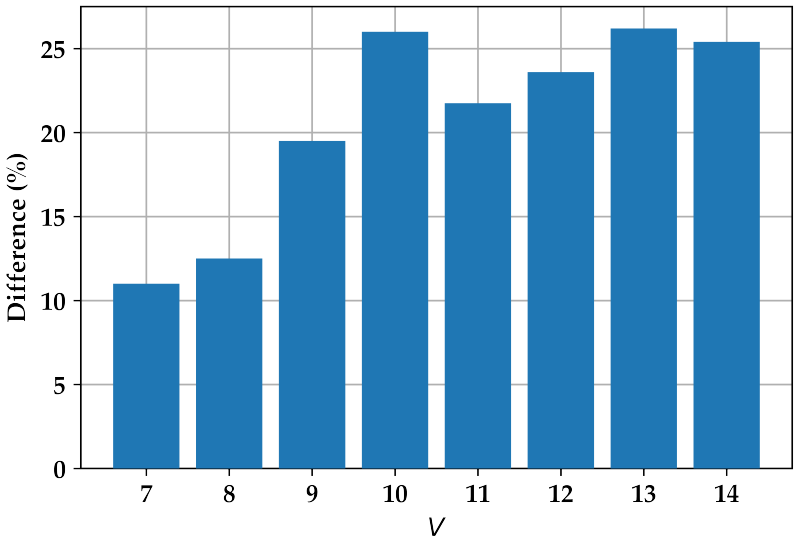


Figure 9: The connectivity difference (%) between the exact and lower bound solutions when $V = \{7, 8, ..., 14\}$ and $LOC_{max} = 3$

4.5 Connectivity of the exact and lower bound solutions w.r.t. LOC_{max}.

This category discusses the connectivity (%) when $V = 8$ and $LOC_{max} = \{2, 3, ..., 5\}$. Similar to the results in figure 7, the lower bound solution in figure 10 shows a highly similar pattern to the exact solution. Also, the connectivity increases as we increase the simulation period, which by product increases LOC_{max}. This behaviour is expected because increasing LOC_{max} should improve the chances of two nodes being connected, which includes s and t.

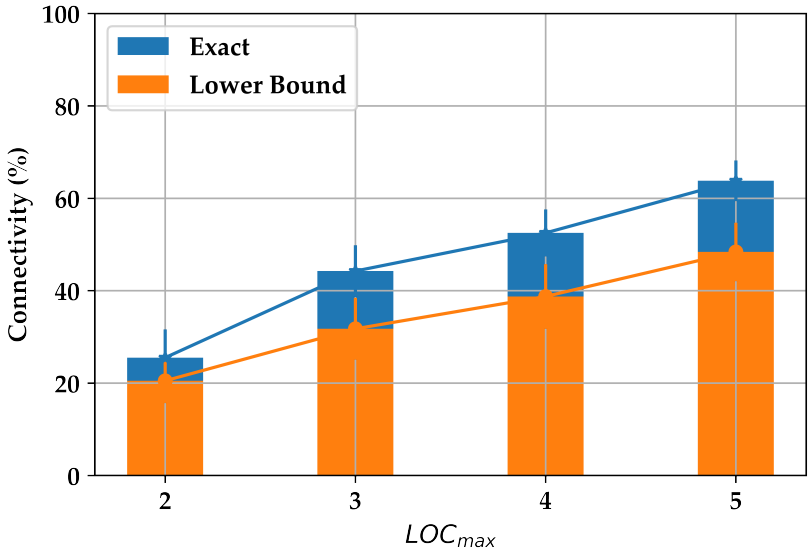


Figure 10: The connectivity of the exact and lower bound solutions when $LOC_{max} = \{2, 3, ..., 5\}$ and $V = 8$

4.6 Running time of the exact and lower bound solutions w.r.t. LOC_{max}.

In this category, we discuss the running time (in minutes) when $V = 8$ and $LOC_{max} = \{2, 3, ..., 5\}$. Again, we notice the significant difference between the lower bound solution and the exact solution as reported in figure 11. Even for the lowest reported LOC_{max}, the lower bound solution returns a solution $\approx 80x$ faster than the exact solution. We can also observe the start of an exponential growth in running time as LOC_{max} increases from four to five. This is in line with the results in figure 8, which further emphasize the efficiency of the lower bound solution.

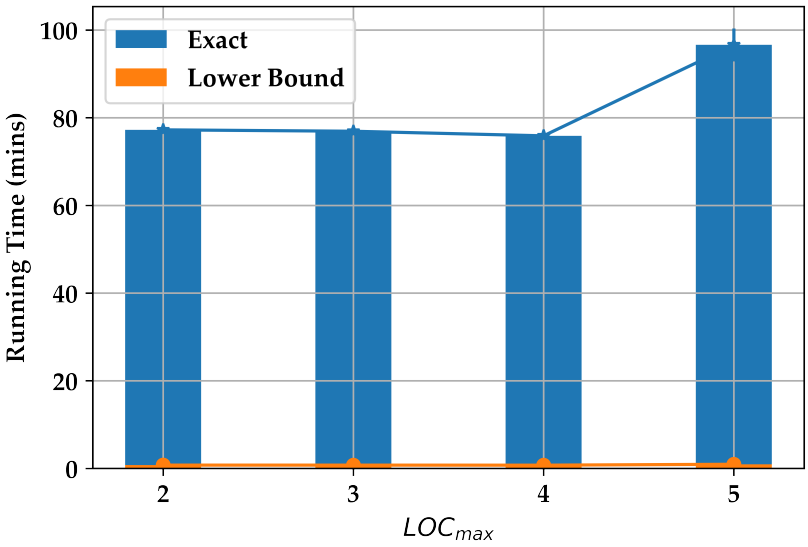


Figure 11: The running time (in minutes) of the exact and lower bound solutions when $V = 8$ and $LOC_{max} = \{2, 3, ..., 5\}$

All the presented results shed light on the capabilities of our solution in generating a lower bound solution that serves as a valuable alternative to an expensive exact solution. We have shown that our solution can generate a solution that, in the worst-case scenario, is only 30% less accurate but 222x faster than the exact solution. We have also shown that our solution performs consistently in spite of the number of nodes and/or the number of locality sets. Unlike the exact solution, where a change (sometime by one) in the number of nodes and/or the number of locality sets is accompanied by an exponential growth in the running time. Thus, as we mentioned earlier, our lower bound solution represents an interesting choice for applications that require fast outcomes (i.e., time-sensitive) without sacrificing accuracy.

5 Conclusion and Future Scope

In this work, we discussed the problem of free floating nodes communicating with one another in an Underwater Wireless Sensor Network (UWSN) environment. We formalized the problem and labeled it as the Two-Nodes connectivity ($2Nodes$ connectivity) problem. We showed that this problem is similar to the *2-terminal network reliability* problem discussed in [Col91]. Both problems belong to the classes of problem known as $\#P - Hard$. Thus, we devised an algorithm that solves the $2Nodes$ connectivity problem in polynomial time given that the input graph is composed of node-disjoint paths. We explained the

node-disjoint paths and how to obtain them from any input graph using the Edmond-Karp algorithm. We noted that our algorithm returns an exact solution for a graph composed of node-disjoint paths, which is also a lower bound solution for any graph where the node-disjoint paths are extracted. To evaluate the performance of our algorithm, we built a dataset using the simulation models proposed in [Car+08; Bou+19]. Using the dataset, we evaluated the performance of our algorithm against an exact solution across four different categories. We showed that our algorithm returns solutions that follow the connectivity "behaviour" of the exact solution and are overwhelmingly faster than the exact solution. Those traits of our solution promote its usefulness for applications where a quick analysis of the network connectivity is more important than an exact report of the connectivity. Nevertheless, our solution can still deliver a reasonably accurate estimation of the network connectivity because the difference in some tested scenarios between our solution and the exact solution is as low as 5%.

Regarding future work, we are planning to transition from simulation to empirical experiment. We will deploy sensor nodes in the ocean and examine the usefulness of our solution against different factors such as the number of nodes and the communication range of each node. We are also planning to implement and compare different communication protocols such as Bluetooth Mesh, Zigbee, and Thread. We believe the transition from theory to practise has its own challenges and limitations that it deserve its own body of work.

References

[FF56] L R Ford Jr and D R Fulkerson. "Maximal flow through a network". In: *Canadian journal of Mathematics* 8.3 (1956), pp. 399–404.

[Col91] Charles J Colbourn. "Combinatorial aspects of network reliability". In: *Annals of Operations Research* 33.1 (1991), pp. 1–15.

[Car+08] A. Caruso et al. "The Meandering Current Mobility Model and its Impact on Underwater Mobile Sensor Networks". In: *IEEE INFOCOM 2008 - The 27th Conference on Computer Communications*. 2008, pp. 221–225. DOI: 10.1109/INFOCOM.2008.53.

[Zho+11] Zhong Zhou et al. "Scalable Localization with Mobility Prediction for Underwater Sensor Networks". In: *IEEE Transactions on Mobile Computing* 10.3 (2011), pp. 335–348. DOI: 10.1109/TMC.2010.158.

[HSZ12] John Heidemann, Milica Stojanovic, and Michele Zorzi. "Underwater sensor networks: applications, advances and challenges". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 370.1958 (2012), pp. 158–175.

[NKK14] Nusrat Nowsheen, Gour Karmakar, and Joarder Kamruzzaman. "MPDF: Movement predicted data forwarding protocol for Underwater Acoustic Sensor Networks". In: *The 20th Asia-Pacific Conference on Communication (APCC2014)*. 2014, pp. 100–105. DOI: 10.1109/APCC.2014.7091613.

[YCH14] Leilei Yu, Dongyan Chen, and Xu Huang. "Robust node-disjoint multipath routing for wireless sensor networks". In: *International Journal of Sensor Networks* 15.2 (2014), pp. 112–120.

[AEE18] Salwa Abougamila, Mohammed Elmorsy, and Ehab S. Elmallah. "On Probabilistic Connected Components in Underwater Sensor Networks". In: *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*. 2018, pp. 200–207. DOI: 10.1109/LCN.2018.8638137.

[Awa+19] Khalid Mahmood Awan et al. "Underwater Wireless Sensor Networks: A Review of Recent Issues and Challenges". In: 2019 (2019). ISSN: 1530-8669. DOI: 10.1155/2019/6470359. URL: https://doi.org/10.1155/2019/6470359.

[Bou+19] Fatma Bouabdallah et al. "Time evolution of underwater sensor networks coverage and connectivity using physically based mobility model". In: *Wireless Communications and Mobile Computing* 2019 (2019).

[Jou+19] Mohammed Jouhari et al. "Underwater Wireless Sensor Networks: A Survey on Enabling Technologies, Localization Protocols, and Internet of Underwater Things". In: *IEEE Access* 7 (2019), pp. 96879–96899. DOI: 10.1109/ACCESS.2019.2928876.

[AEE20] Salwa Abougamila, Mohammed Elmorsy, and Ehab S. Elmallah. "On Connected Components in Multistate Wireless Sensor Network Probabilistic Models". In: *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*. 2020, pp. 1–7. DOI: 10.1109/ICC40277.2020.9148670.

[HV20] Rachid Hadid and Vincent Villain. "A Self-stabilizing One-To-Many Node Disjoint Paths Routing Algorithm in Star Networks". In: *IFIP International Conference on Distributed Applications and Interoperable Systems*. Springer. 2020, pp. 186–203.

[HKA20] Khandaker Foysal Haque, K Habibul Kabir, and Ahmed Abdelgawad. "Advancement of routing protocols and applications of underwater wireless sensor network (UWSN)—A survey". In: *Journal of Sensor and Actuator Networks* 9.2 (2020), p. 19.

[XH20] Xingxing Xiao and Haining Huang. "A Clustering Routing Algorithm Based on Improved Ant Colony Optimization Algorithms for Underwater Wireless Sensor Networks". In: *Algorithms* 13.10 (2020). ISSN: 1999-4893. DOI: 10.3390/a13100250. URL: https://www.mdpi.com/1999-4893/13/10/250.

[XHW20] Xingxing Xiao, Haining Huang, and Wei Wang. “Underwater wireless sensor networks: An energy-efficient clustering routing protocol based on data fusion and genetic algorithms”. In: *Applied Sciences* 11.1 (2020), p. 312.

[Zan+20] Elma Zanaj et al. “Underwater wireless sensor networks: Estimation of acoustic channel in shallow water”. In: *Applied Sciences* 10.18 (2020), p. 6393.

[Ker+22] Walter Kern et al. “Disjoint paths and connected subgraphs for H-free graphs”. In: *Theoretical Computer Science* 898 (2022), pp. 59–68.

[Lu+22] Jialiang Lu et al. “BCCC Disjoint Path Construction Algorithm and Fault-Tolerant Routing Algorithm under Restricted Connectivity”. In: *Algorithms* 15.12 (2022), p. 481.

[SS22] Iftekharul Islam Shovon and Seokjoo Shin. “Survey on Multi-Path Routing Protocols of Underwater Wireless Sensor Networks: Advancement and Applications”. In: *Electronics* 11.21 (2022), p. 3467.

[Wu+22] Suying Wu et al. “Connectivity and constructive algorithms of disjoint paths in dragonfly networks”. In: *Theoretical Computer Science* 922 (2022), pp. 257–270.

[Zhu+22] Rongxin Zhu et al. “An On-Site-Based Opportunistic Routing Protocol for Scalable and Energy-Efficient Underwater Acoustic Sensor Networks”. In: *Applied Sciences* 12.23 (2022), p. 12482.