Profiling Aquatic Diffusion Process Using Robotic Sensor Networks

Yu Wang, Rui Tan, Guoliang Xing, Jianxun Wang, and Xiaobo Tan

Abstract—Water resources and aquatic ecosystems are facing increasing threats from climate change, improper waste disposal, and oil spill incidents. It is of great interest to deploy mobile sensors to detect and monitor certain diffusion processes (e.g., chemical pollutants) that are harmful to aquatic environments. In this paper, we propose an accuracy-aware diffusion process profiling approach using smart aquatic mobile sensors such as robotic fish. In our approach, the robotic sensors collaboratively profile the characteristics of a diffusion process including source location, discharged substance amount, and its evolution over time. In particular, the robotic sensors reposition themselves to progressively improve the profiling accuracy. We formulate a novel movement scheduling problem that aims to maximize the profiling accuracy subject to the limited sensor mobility and energy budget. We develop an efficient greedy algorithm and a more complex near-optimal radial algorithm to solve the problem. We conduct extensive simulations based on real data traces of GPS localization errors, robotic fish movement, and wireless communication. The results show that our approach can accurately profile dynamic diffusion processes under tight energy budgets. Moreover, a preliminary evaluation based on the implementation on TelosB motes validates the feasibility of deploying our profiling algorithms on mote-class robotic sensor platforms.

Index Terms—Robotic sensor networks, diffusion process, movement scheduling

1 INTRODUCTION

W

ot only have water resources and aquatic ecosystems been facing various physical, chemical, and biological threats from climate change, industrial pollution, and improper waste disposal. For instance, the last four decades witnessed more than a dozen major oil spills with each releasing more than 30 million gallons of oil [1]. Other harmful diffusion processes like chemical leaks could also have disastrous impact on public health and ecosystem sustainability. When such a crisis arises, an immediate requirement is to profile the characteristics of the diffusion process, including the location of source, the amount of discharged substance, and how rapidly it spreads in space and evolves over time.

Manual sampling, via boat/ship or with handheld devices, is still a common practice in the monitoring of aquatic diffusion processes. This approach is labor-intensive and difficult to adapt to the dynamic evolution of diffusion. An alternative is in situ sensing with fixed or buoyed/moored sensors [2]. However, since buoyed sensors cannot move around, it could take a prohibitive-ly large number of them to capture spatially inhomogeneous information. The past couple of decades have seen significant progress in developing robotic technologies for aquatic sensing. Autonomous underwater vehicles (AUVs) [3] and sea gliders [4] are notable examples of such technologies. However, because of their high cost (over 50,000 U.S. dollars per unit [5]), weight (over 100 pounds), and size (1-2 meters long), it is difficult to deploy many AUVs or sea gliders for temporally and spatially resolved measurement of diffusion processes.

Recent advances in computing, communication, sensing, and actuation technologies have made it possible to create untethered robotic fish with onboard power, control, navigation, wireless communication, and sensing modules, which turn these robots into mobile sensing platforms in aquatic environments. Fig. 1(a) shows a prototype of robotic fish swimming in an inland lake. Fig. 1(b) shows the close-up of a robotic fish prototype, equipped with GPS, Zigbee antenna, and a dissolved oxygen (DO) sensor. Due to the low manufacturing cost, these platforms can be massively deployed to form a mobile sensor network that monitors harmful diffusion processes, providing significantly higher spatial and temporal sensing resolution than existing monitoring methods. Moreover, a school of robotic fish can coordinate their sensing and movements through wireless communication enabled by the onboard Zigbee radio, to adapt to the dynamics of evolving diffusion processes.

Despite the aforementioned advantages, low-cost mobile sensing platforms like robotic fish introduce several
challenges for aquatic sensing. First, due to the constraints on size and energy, they are typically equipped with low-end sensors whose measurements are subject to significant biases and noises. They must efficiently collaborate in data processing to achieve satisfactory accuracy in diffusion profiling. Second, practical aquatic mobile platforms are only capable of relatively low-speed movements. Hence the movements of sensors must be efficiently scheduled to achieve real-time profiling of the diffusion processes that may evolve rapidly over time. Third, due to the high power consumption of locomotion, the distance that robotic sensors move in a profiling process should be minimized to extend the network lifetime.

To address the above challenges, we make the following major contributions in this paper:

- We propose a novel accuracy-aware approach for aquatic diffusion profiling based on robotic sensor networks. Our approach leverages the mobility of robotic sensors to iteratively profile the spatiotemporally evolving diffusion process.
- We derive the analytical profiling accuracy of our approach based on the Cramér-Rao bound (CRB). Then we formulate a movement scheduling problem for aquatic diffusion profiling, in which the profiling accuracy is maximized under the constraints on sensor mobility and energy budgets. We develop gradient-ascent-based greedy and dynamic-programming-based radial movement scheduling algorithms to solve the problem.
- We implement the profiling and movement scheduling algorithms on TelosB motes and evaluate the system overhead. Moreover, we conduct extensive simulations based on real data traces of GPS localization errors, robotic fish movement, and wireless communication for evaluation. The results show that our approach can accurately profile the dynamic diffusion process and adapt to its evolution.

The rest of this paper is organized as follows. Section 2 reviews related work. Section 3 introduces the preliminaries and Section 4 provides an overview of our approach. Section 5 derives the analytical profiling accuracy metric and Section 6 formulates the movement scheduling problem. Section 7 presents the two movement scheduling algorithms. Section 8 discusses several extensions. Section 9 presents evaluation results and Section 10 concludes this paper.

2 Related Work

Most previous work on diffusion process monitoring is based on stationary sensor networks. Several different estimation techniques are adopted by these studies, which include state-space filtering, statistical signal processing, and geometric trilateration. The state space approach [7], [8] uses discrete state-space equations to approximate the partial differential equations that govern the diffusion process, and then applies filtering algorithms such as Kalman filters [7], [8] to profile the diffusion process based on noisy measurements. In the statistical signal processing approach, several estimation techniques such as MLE [9], [10] and Bayesian parameter estimation [11] are applied to deal with noisy measurements. For instance, in [10], an MLE-based diffusion characterization algorithm is designed based on binary sensor measurements to reduce the communication cost. In [11], the parametric probability distribution of the diffusion profile parameters is passed among sensors and updated with sensor measurements by a Bayesian estimation algorithm. The passing route is determined according to various estimation performance metrics including CRB.

In geometric trilateration approach [12], [13], the measurement of a sensor is mapped to the distance from the sensor to the diffusion source. The source location can then be estimated by trilateration among multiple sensors. Such an approach incurs low computational complexities, but suffers lower estimation accuracy compared with more advanced approaches such as MLE [13].

Recently, sensor mobility has been exploited to enhance the adaptability and sensing capability of sensor networks. For instance, heuristic movement scheduling algorithms are proposed in [14] to estimate the contours of a physical field. In [15], more complex path planning schemes are proposed for mobile sensors to reconstruct a spatial map of environmental phenomena that do not follow a particular physical model. Our previous works exploit reactive sensor mobility to improve the detection performance of a sensor network [16], [17]. Several studies are focused on using robots to improve the accuracy in profiling diffusion processes. As an extension to [9], the gradient of CRB is used to schedule the movement of a single sensor in [10]. Similarly, a robot motion control algorithm is proposed in [18] to maximize the determinant of the Fisher information matrix. However, as these diffusion profiling approaches [9], [10], [18] adopt complicated numerical optimization, they are only applicable to a small number (e.g., 3 in [18]) of powerful robots. In contrast, we focus on developing movement scheduling algorithms for moderate- or large-scale mobile networks that are composed of inexpensive robotic sensors.

3 Preliminaries

3.1 Diffusion Process Model

A diffusion process in a static aquatic environment, by which molecules spread from areas of higher concentration to areas of lower concentration, follows Fick’s law [19]. In addition to the diffusion, the spread of the discharged substance is also affected by the advection of solvent, e.g., the movement of water caused by the wind. By denoting $t$ as the time elapsed since the discharge of substance and $c$ as the substance concentration, the diffusion-advection model can be described as

$$\frac{\partial c}{\partial t} = D_x \frac{\partial^2 c}{\partial x^2} + D_y \frac{\partial^2 c}{\partial y^2} + D_z \frac{\partial^2 c}{\partial z^2} - u_x \frac{\partial c}{\partial x} - u_y \frac{\partial c}{\partial y}$$  (1)
where $D$ is the diffusion coefficient, $u$ is the advection speed, and the subscripts of $D$ and $u$ denote the directions (i.e., $x$, $y$, and $z$-axis). The diffusion coefficients characterize the speed of diffusion and depend on the species of solvent and discharge substance as well as other environment factors such as temperature. The advection speeds characterize the horizontal solvent movement caused by external forces such as wind and flow. The above Fickian diffusion-advection model has been widely adopted to study the spreading of gaseous substances [10] and buoyant fluid pollutants such as oil slick on the sea [20].

For many buoyant fluid pollutants, the two horizontal diffusion coefficients, i.e., $D_x$ and $D_y$, are identical, while the vertical diffusion coefficient, i.e., $D_z$, is insignificant. For instance, in a field experiment [21], where diesel oil was discharged into the sea, the estimated $D_x$ is 2,000 cm$^2$/s while $D_z$ is only 10 cm$^2$/s. Therefore, the vertical diffusion coefficient can be safely ignored and the diffusion can be well characterized by a 2-dimensional process. In this paper, our study is focused on buoyant fluid pollutants with the diffusion coefficients $D_x = D_y = D$.

Suppose a total of $A$ cm$^3$ of substance is discharged at location $(x_s, y_s)$ and $t = 0$. At time $t > 0$, the original diffusion source is drifted to $(x_0, y_0)$ due to advection, where $x_0 = x_s + u_x t$ and $y_0 = y_s + u_y t$. Hereafter, by source location we refer to the source location that has drifted from the original position due to advection, unless otherwise specified. Denote $d(x, y)$ (abbreviated to $d$) as the distance from any location $(x, y)$ to the source location, i.e., $d = \sqrt{(x - x_0)^2 + (y - y_0)^2}$. In the presence of advection, the diffusion is isotropic with respect to the drifted source location [22]. Therefore, the diffusion profile can be denoted as $c(d, t)$. The initial condition for Eq. (1) is an impulse source, which can be represented by the Dirac delta function, i.e., $c(d, 0) = A \cdot \delta(d)$. The closed-form solution to Eq. (1) is given by [10]:

$$c(d, t) = \alpha \cdot \exp\left(-\beta \cdot d^2\right), \quad d \geq 0, \ t > 0,$$

(2)

where $\alpha = \frac{A}{\pi \beta^{3/2}}$ and $\beta = \frac{1}{4Dt}$. From Eq. (2), for a given time instant $t$, the concentration distribution is described by the Gaussian function that centers at the source location. As time elapses, the concentration distribution becomes flatter. In this paper, the diffusion profile is defined as $\Theta = \{x_0, y_0, \alpha, \beta\}$.

We now validate Eq. (2) with real lab experiments of Rhodamine-B diffusion. We discharge Rhodamine-B solution in saline water, and periodically capture diffusion process using a digital camera. We assume that the grayscale of a pixel in the captured image linearly increases with the concentration at the corresponding physical location [23]. Therefore, the evolution of diffusion process can be characterized by the expansion of a contour given a certain threshold of grayscale in the captured images. With the measured contour areas along the recorded shooting times, we can estimate $D$ by linear regression. Our methodology of model validation is to compare the contour area observed in images with that predicted by the model in Eq. (2). The detailed derivations are available in [24]. Fig. 2 plots the captured images with contours marked in white. Fig. 3 plots the contour areas observed in images and predicted by Eq. (2) versus $t$. We can see that the model in Eq. (2) well characterizes the diffusion process of Rhodamine-B. Although we only verify the Fickian diffusion model in small scale, this model has also been validated using data traces of the large-scale spreading of buoyant fluid pollutants, e.g., the oil slick in [20].

### 3.2 Sensor Model

Our approach leverages mobile nodes (e.g., robotic fish [6]) to collaboratively profile an aquatic diffusion process. The nodes form a cluster, and a cluster head is selected to process the measurements from cluster members. The selection of cluster head will be discussed in Section 7.2. Moreover, we will extend our approach to address multiple clusters in Section 9.2. Many aquatic mobile platforms are battery-powered and hence have limited mobility and energy budget. For instance, the movement speed of the robotic fish designed in [6] was about 1.8 to 6 m/min. We assume that the mobile nodes are equipped with pollutant concentration sensors (e.g., the Cyclops-7 series [25]) that can measure the concentrations of crude oil, harmful algae, etc. Lastly, we assume that the sensors are equipped with low-power wireless interfaces (e.g., 802.15.14 Zigbee radios) and can communicate with each other on water surface.

The measurements of most sensors are subject to biases and additive random noises from the sensor circuitry and the environment (e.g., wave). Specifically, the reading of sensor $i$, denoted by $z_i$, is given by $z_i = c(d_i, t) + b_i + n_i$, where $d_i$ is the distance from sensor $i$ to the diffusion source, $b_i$ and $n_i$ are the bias and
random noise for sensor $i$, respectively. In the presence of constant-speed advection, the source and the sensors will drift with the same speed and therefore they are in the same inertial system. As a result, the concentration at the position of sensor $i$ is given by $c(d_i, t)$. We assume that the noise experienced by sensor $i$ follows the zero-mean normal distribution with variance $\varsigma^2$, i.e., $n_i \sim \mathcal{N}(0, \varsigma^2)$. We assume that the noises, i.e., $\{n_i|\forall i\}$, are independent across sensors. Such a measurement model has been widely adopted for various chemical sensors [9], [10], [11]. We assume that the biases and noise variance (i.e., $\{b_i|\forall i\}$ and $\varsigma$) are known constants. For instance, they are often given in the sensor specification provided by the manufacturer. Moreover, they can be measured in offline lab experiments. Specifically, by placing a sensor in the pollutant-free fluid media, the bias and noise variance can be estimated by the sample mean and variance over a number of readings [26].

In this paper, we adopt a temporal sampling scheme to mitigate the impact of noise. When sensor $i$ measures the concentration, it continuously takes $K$ samples in a short time, and computes the average as its measurement. Hence, the measurement $z_i$ follows the normal distribution, i.e., $z_i \sim \mathcal{N}(c(d_i, t) + b_i, \sigma^2)$, where $\sigma^2 = \varsigma^2/K$.

4 Overview of Approach

In this section, we provide an overview of our approach. Our objective is to profile (i.e., estimate $\Theta$) of an aquatic diffusion-advection process using a robotic sensor network. Our approach is designed to meet two key objectives. First, the noisy measurements of sensors are jointly processed to improve the accuracy in profiling the diffusion. Second, sensors can actively move based on current measurements to maximize the profiling accuracy subject to the energy consumption budget. With the estimated profile $\hat{\Theta}$, we can learn several important characteristics of the diffusion process of interest.$^1$ First, we can compute the current concentration contour maps with Eq. (2). Second, we can estimate the elapsed time since the start of the diffusion and the total amount of discharged substance, with $\tilde{t} = (4D\hat{\beta})^{-1}$ and $A = \pi\hat{\alpha}\hat{\beta}^{-1}$. Third, we can estimate the original source location by $\tilde{x}_s = x_0 - u_x\tilde{t}$ and $\tilde{y}_s = y_0 - u_y\tilde{t}$. Moreover, we can predict the evolution of the diffusion in the future, which is often important for emergency management in the cases of harmful substance discharge.

We assume that the robotic sensors are initially distributed at randomly chosen positions in the deployment region that covers the diffusion source. For instance, the sensors can be dropped off from an unmanned aerial vehicle or placed by an aquatic vessel randomly. Note that random and sometimes uniform deployment of sensors around the source location is a good strategy when the characteristics of diffusion process have yet to be determined. This also avoids the massive locomotion energy required to spread sensors for a satisfactory profiling accuracy when the diffusion process evolves. We assume that all sensors know their positions (e.g., through GPS or an in-network localization service) and are time-synchronized. In Section 9.3.6, we will evaluate the impact of initial sensor deployment on the profiling accuracy and locomotion energy consumption.

After the initial deployment, sensors begin a diffusion profiling process consisting of multiple profiling iterations. The iterative profiling process is illustrated in Fig. 4. In a profiling iteration, sensors first simultaneously take concentration measurements and send them to the cluster head. Various existing data collection protocols [27], [28] can be used to collect the measurements. The cluster head then adopts the maximum likelihood estimation (MLE) to estimate $\Theta$ from the noisy measurements of all sensors. With the estimated diffusion profile, the cluster head schedules sensor movements such that the expected profiling accuracy in the next profiling iteration is maximized, subject to the limited sensor mobility and energy budget. Finally, the movement schedule including moving orientations and distances is sent to sensors to direct their movements.

Our accuracy-aware diffusion profiling approach features the following novelties. First, it starts with little prior knowledge about the diffusion and progressively learns the profile of the diffusion with improved accuracy along the profiling iterations. As sensors resample the concentration in each iteration, such an iterative profiling strategy allows the network to adapt to the dynamics of the diffusion process while reducing energy consumption of robotic sensors. Moreover, although the profiling accuracy in each iteration is affected by errors in sensor localization and movement control, our approach only schedules short-distance movements for sensors in each iteration and updates their positions in the next iteration, which avoids the accumulation of errors in sensor localization and movement control. Second, we analyze the CRB of the MLE-based diffusion profiling algorithm and propose a novel CRB-based profiling accuracy metric, which is used to direct the movement scheduling of robotic sensors. Third, we propose two novel movement scheduling algorithms, which include a gradient-ascent-based greedy algorithm and a dynamic-programming-based radial algorithm. The greedy algorithm only incurs linear complexity, while the radial algorithm can find the near-optimal movement schedule with a higher but still polynomial complexity.

1. For the clarity of presentation, we denote $\bar{x}$ as the estimate of $x$. 

![Fig. 4. The iterative diffusion profiling process.](image-url)
5 Profiling Algorithm and Accuracy Analysis

In this section, we first present our MLE-based diffusion profiling algorithm, which estimates the diffusion profile Θ in each profiling iteration. We then analyze the theoretical profiling accuracy based on CRB. The closed-form relationship between the profiling accuracy and the sensors’ positions will guide the design of our accuracy-aware sensor movement scheduling algorithms.

5.1 MLE-based Diffusion Profiling Algorithm

Suppose that a total of N aquatic sensors are deployed in the region of interest. To simplify the analysis, we define the bias-removed and normalized observation vector z as $z = [\bar{z}_1 - \bar{a}_1, \ldots, \bar{z}_N - \bar{a}_N]^T$. As the sensor biases and noise variance are known, z can be easily calculated by the cluster head based on the noisy sensor measurements. By denoting $H = [\sigma^{-1} e^{-\beta d_1^2}, \ldots, \sigma^{-1} e^{-\beta d_K^2}]^T$, z follows the $N$-dimensional normal distribution, i.e., $z \sim N(\alpha H, I)$, where I is an $N \times N$ identity matrix. The log-likelihood of an observation z given Θ is expressed by [29]:

$$L(z|\Theta) = -(z - \alpha H)^T(z - \alpha H). \tag{3}$$

MLE maximizes the log-likelihood given by Eq. (3). Formally, $\hat{\Theta}(z) = \arg \max_\Theta L(z|\Theta)$. This unconstrained optimization problem can be solved by various numerical methods, e.g., Nelder-Mead’s algorithm [30].

5.2 Cramér-Rao Bound for Diffusion Profiling

CRB provides a theoretical lower bound on the variance of parameter estimators [29], and has been widely adopted to guide the design of estimation algorithms [9], [11]. This section derives the CRB of $\hat{\Theta}(z)$. CRB is given by the inverse of the Fisher information matrix (FIM) [29]. For the diffusion profiling, the FIM is defined by $J = -E \left[ \frac{\partial^2}{\partial \theta^2} \mathcal{L}(z|\Theta) \right] = \alpha^2 \frac{\partial^2 H^T}{\partial \theta^2} \frac{\partial^2 H}{\partial \theta^2}$, where the expectation $E[\cdot]$ is taken over all z. The $k$th diagonal element of the inverse of J (denoted by $J^{-1}_{k,k}$) provides the lower bound on the variance of the $k$th element of $\Theta$ (denoted by $\hat{\Theta}_k$) [29]. Formally, $\text{Var}(\hat{\Theta}_k) \geq J^{-1}_{k,k}$. The number $J^{-1}_{k,k}$ is the CRB corresponding to $\hat{\Theta}_k$, which is denoted as $\text{CRB}(\hat{\Theta}_k)$ in this paper. Although the CRB can be easily computed via numerical methods, in order to guide the movements of sensors, we will derive the closed-form CRB.

Even though J is just a $4 \times 4$ matrix, deriving $J^{-1}$ is challenging, because the $N$-dimensional joint distribution function in Eq. (3) leads to high inter-node dependence. To simplify the discussion, we set up a Cartesian coordinate system with the origin at the source location and let $(x_i, y_i)$ denote the coordinates of sensor $i$. Note that the coordinates of the diffusion source and sensor $i$ in the global coordinate system are $(x_0, y_0)$ and $(x_0 + x_i, y_0 + y_i)$, respectively. The FIM is a diagonal matrix that can be expressed by $J = [J_{11}, J_{12}; J_{12}^T, J_{22}]$. The blocks $J_{11}$, $J_{12}$, and $J_{22}$ are $2 \times 2$ matrices. These blocks can be derived as

$$J_{11} = 4\alpha^2 \beta^2 \frac{\sigma^2}{2} \left[ \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{x_i x_j}{e^{\beta d_i^2} e^{\beta d_j^2}} \sum_{i=1}^{N} \frac{x_i y_j}{e^{\beta d_i^2} e^{\beta d_j^2}} \sum_{i=1}^{N} \frac{y_i y_j}{e^{\beta d_i^2} e^{\beta d_j^2}} \right], \tag{4}$$

$$J_{12} = 2\alpha \beta \frac{\sigma^2}{\sigma^2} \left[ \sum_{i=1}^{N} \alpha_0^2 d_i x_i e^{\beta d_i^2} \sum_{i=1}^{N} \alpha_0^2 d_i y_i e^{\beta d_i^2} \sum_{i=1}^{N} \alpha_0^2 d_i^2 e^{\beta d_i^2} \right], \tag{5}$$

$$J_{22} = \frac{1}{\sigma^2} \left[ \sum_{i=1}^{N} \alpha_0^2 d_i^2 e^{\beta d_i^2} \sum_{i=1}^{N} \alpha_0^2 d_i^2 e^{\beta d_i^2} \sum_{i=1}^{N} \alpha_0^2 d_i^2 e^{\beta d_i^2} \right]. \tag{6}$$

After block manipulations, $J^{-1}$ is given by

$$J^{-1} = \left[ \begin{array}{cc} F^{-1} & -F J_{22} F^{-1} J_{12} \\ J_{22}^{-1} J_{12} F^{-1} & J_{22}^{-1} (I + J_{22} F^{-1} J_{12}) \end{array} \right],$$

where $F = J_{11} - J_{12} J_{22}^{-1} J_{21}$. We define several notations, i.e., $\hat{x}_i, \hat{y}_i, L_{x1}, L_{x2}, L_{y1}, L_{y2}$, and $L_{y3}$, to help the representation of CRB. First, $\hat{x}_i$ is given by

$$\hat{x}_i = \frac{\sum_{j=1}^{N} x_j (d_{ij}^2 - d_{ij}^2 e^{-2\beta d_{ij}^2})}{\sqrt{\sum_{m=1}^{N} \sum_{n=1}^{N} (d_{mn}^2 - d_{mn}^2 e^{-2\beta d_{mn}^2})}}. \tag{7}$$

By replacing $x_j$ in Eq. (4) with $y_j$, we can define $\hat{y}_i$ in a similar manner. Moreover, $L_{x1}, L_{y1}, L_{y2}, L_{y3}$ are $N \times 1$ vectors. The $i$th elements of them are defined as

$$L_{x1}(i) = \sigma^{-1} e^{-\beta d_i^2} (x_i - \hat{x}_i), \quad L_{y1}(i) = \sigma^{-1} e^{-\beta d_i^2} (y_i - \hat{y}_i), \tag{8}$$

$$L_{x2}(i) = \sigma^{-1} e^{-\beta d_i^2} (x_i - \hat{x}_i), \quad L_{y2}(i) = \sigma^{-1} e^{-\beta d_i^2} (y_i - \hat{y}_i). \tag{9}$$

With the above notations, $F$ can be expressed as

$F = 2\alpha^2 \beta^2 \left[ \begin{array}{cc} 2L_{x1} L_{x2} & L_{x1} L_{y2} + L_{y1} L_{x2} + 2L_{y1} L_{y2} \\ L_{x1} L_{y2} + L_{x2} L_{y1} + 2L_{y1} L_{y2} & 2L_{y1} L_{y2} \end{array} \right]. \tag{10}$

The CRBs for the estimates of $x_0$ and $y_0$ are given by $F_{11} = F_{11}^{-1}$, $F_{22} = F_{22}^{-1}$, $F_{11}$, respectively. Specifically,

$$\text{CRB}(x_0) = F_{11}^{-1} = \frac{(4 \alpha^2 \beta^2)^{-1} L_{x1} L_{x2}}{4 L_{y1} L_{y2}}, \tag{11}$$

$$\text{CRB}(y_0) = F_{22}^{-1} = \frac{(4 \alpha^2 \beta^2)^{-1} L_{y1} L_{y2}}{4 L_{x1} L_{x2}}. \tag{12}$$

We now discuss how well the CRBs can characterize the MLE-based profiling algorithm discussed in Section 5.1. As shown in [31], the variance of a parameter estimated by MLE converges to its CRB when the number
of sensors \( N \) approaches infinity. We now evaluate the convergence under realistic settings of \( N \). Fig. 5 shows \( \text{Var}(x_0) \) and \( \text{CRB}(x_0) \) versus \( N \), where the sensors are randomly deployed in a \( 200 \times 200 \text{ m}^2 \) region. We can see that \( \text{Var}(x_0) \) approaches \( \text{CRB}(x_0) \) as \( N \) increases. Moreover, we evaluate the impact of sensor sampling scheme on the convergence. Note that sensor’s signal-to-noise ratio (SNR) increases with the number of samples \( K \). We can see that MLE shows better convergence for larger \( K \). For instance, if \( K = 10 \) and \( N \geq 10 \), the difference between \( \text{Var}(x_0) \) and \( \text{CRB}(x_0) \) is insignificant. These results show that the CRB can well characterize the accuracy of the MLE-based profiling algorithm under realistic setting of network size.

5.3 Profiling Accuracy Metric

In this section, we propose a novel diffusion profiling accuracy metric based on the CRBs derived in Section 5.2, which will be used to guide the movements of sensors in Section 6. Several previous works [18] adopt the determinant of the FIM as the accuracy metric, which jointly considers all the parameters. Such a metric requires the parameters to be properly normalized to avoid biases. However, normalizing the parameters with different physical meanings is highly problem-dependent. Moreover, as the closed-form determinant of the FIM is extremely complicated, the resulted sensor movement scheduling has to rely on the numerical methods with high computational complexities [18], which is not suitable for robotic sensors with limited resources. In this paper, we propose a new profiling accuracy metric, denoted by \( \omega \), which is defined according to the sum of reciprocals of \( \text{CRB}(x_0) \) and \( \text{CRB}(y_0) \). Formally,

\[
\omega = \frac{1}{\text{CRB}(x_0) + \text{CRB}(y_0)} = (1 - \epsilon) \left( L_{x_1, x_2} + L_{y_1, y_2} \right),
\]

where \( \epsilon = \frac{\left(L_{x_1, x_2} + L_{y_1, y_2}\right)^2}{4L_{x_1, x_2}L_{x_1, y_1}L_{y_2}} \). By adopting reciprocals, the accuracy analysis can be greatly simplified. Note that as \( \alpha \) and \( \beta \) are unknown but fixed in a particular profiling iteration, \( 4\alpha^2\beta^2 \) in the denominator of Eq. (8) is a scaling factor. Therefore, optimizing \( \frac{1}{\text{CRB}(x_0)} + \frac{1}{\text{CRB}(y_0)} \) is equivalent to optimizing \( \omega \). As discussed in Section 5.1, we adopt the MLE to estimate \( \Theta \). The variance of the MLE result converges to CRB and hence a larger \( \omega \) indicates more accurate estimation of \( x_0 \) and \( y_0 \). With the metric \( \omega \), the movements of sensors will be directed according to the accuracy of localizing the diffusion source. In the rest of this paper, the term profiling accuracy refers to the metric \( \omega \) defined in Eq. (8). Note that our approach can also be applied to focus on the profiling accuracy of the elapsed time \( t \) and discharged substance amount \( A_t \) by applying the same matrix manipulations to obtain \( \text{CRB}(\alpha) \) and \( \text{CRB}(\beta) \).

5.4 Approximated Profiling Accuracy

The profiling accuracy metric \( \omega \) proposed in Section 5.3 can characterize the profiling performance. However, its expression given by Eq. (8) is still too complex to find efficient movement scheduling algorithms. Hence, we derive the approximation to Eq. (8). If sensors are randomly distributed around the diffusion source, \( \epsilon \) is close to zero. By assuming a random sensor distribution and setting \( \epsilon = 0 \), the \( \omega \) can be approximated as:

\[
\omega \approx \sum_{i=1}^{N} \omega_i, \quad \omega_i = \frac{1}{\sigma^2 - \min_{j \in [1, N], j \neq i} d_i^2 - d_j^2}, \tag{9}
\]

where \( \omega_i \) can be regarded as the contribution of sensor \( i \) to the overall profiling accuracy. As \( \omega_i \) depends on \( d_i \) and the minimum distance to the source from other sensors, Eq. (9) highly reduces the inter-node dependence compared with Eq. (8). Note that the approximation error (i.e., \( \omega - \sum_{i=1}^{N} \omega_i \)) can be easily calculated by the cluster head with the movement schedule.

We now evaluate the above approximation by Monte Carlo method. We define the relative approximation error as \( |\omega - \sum_{i=1}^{N} \omega_i|/\omega \). In the Monte Carlo simulations, we generate a large number of deployments over a disc region with radius of 100 m. In each deployment, sensors are randomly distributed and we then calculate the relative approximation error. Fig. 6 shows the cumulative distribution function (CDF) and the average of the relative approximation error given different number of sensors. We can see that the average relative approximation error is only 10% when 20 sensors are deployed. The approximation in this section assumes that the deployment region is centered at the diffusion source. In Section 9.3.5, we will evaluate the case where the deployment region is biased from the diffusion source.

6 Diffusion Process Profiling Using Robotic Sensors

In this section, we formally formulate the movement scheduling problem. Because of the limited mobility and energy budget of aquatic mobile sensors, the sensor movements must be efficiently scheduled in order to achieve the maximum profiling accuracy. As the power consumption of sensing, computation and radio transmission is significantly less than that of locomotion [6], in this paper we only consider the locomotion energy. Moreover, as the locomotion energy is approximately proportional to the moving distance [32], we will use moving distance to quantify the locomotion energy consumption. To simplify the motion control of sensors, we assume that a sensor moves straight in each profiling iteration and the moving distance is always multiple of
l meters, where l is referred to as step. We note that this model is motivated by the locomotion and computation limitation typically seen for aquatic sensor platforms. First, the locomotion of robotic fish is typically driven by closed-loop motion control algorithms, resulting in constant course-correction during movement. Second, each profiling iteration has short time duration. As a result, the assumption of sensor’s straight movement in an iteration does not introduce significant errors in the movement scheduling. As the estimation and movement are performed in an iterative manner, we will focus on the movement scheduling in one profiling iteration. Denote \( m_i \in \mathbb{Z}^+ \) and \( \phi_i \in [0, 2\pi) \) as the number of steps and movement orientation of sensor \( i \) in a profiling iteration, respectively. Our objective is to maximize the expected profiling accuracy after sensor movements, subject to the constraints on total energy budget and sensor’s individual energy budget. The movement scheduling problem for diffusion profiling is formally formulated as follows:

Movement Scheduling Problem. Suppose that a total of \( M \) steps can be allocated to sensors and sensor \( i \) can move at most \( L_i \) meters in a profiling iteration. Find the allocation of steps and movement orientations for all \( N \) sensors, i.e., \( \{m_i, \phi_i\}_{i \in [1, N]} \), such that the profiling accuracy \( \omega \) (defined by Eq. (8)) after sensor movements is maximized, subject to:

\[
\sum_{i=1}^{N} m_i \leq M, \tag{10}
\]

\[
m_i \cdot l \leq L_i, \quad \forall i. \tag{11}
\]

Eq. (10) upper-bounds the total locomotion energy in a profiling iteration. Eq. (11) can be used to constrain the energy consumption of individual sensors. For instance, \( L_i \) can be specified according to the sensor’s residual energy. Moreover, \( L_i \) can also be specified to ensure the delay of a profiling iteration. If sensors move at a constant speed of \( v \) m/s and a profiling iteration is required to be completed within \( \tau \) seconds to achieve the desired temporal resolution of profiling, \( L_i \) can be set to \( L_i = v \cdot \tau \). As discussed in Section 4, the cluster head adopts MLE to estimate \( \Theta \), and then schedules the movements of sensors such that the expected \( \omega \) in the next profiling iteration is maximized, subject to the constraints in Eqs. (10) and (11). An exhaustive search to the above problem would yield an exponential complexity with respect to \( N \), which is \( O((\phi_0 \cdot L_i) N) \) where \( \phi_0 \) is the granularity in searching for the movement orientation. Such a complexity is prohibitively high as the problem needs to be solved in each profiling iteration by the cluster head. In the next section, we will propose an efficient greedy algorithm and a near-optimal radial algorithm that are feasible to mote-class sensor platforms.

7 Sensor Movement Scheduling Algorithms

In this section, we propose an efficient greedy movement scheduling algorithm based on gradient ascent and a near-optimal radial algorithm based on dynamic programming to solve the problem formulated in Section 6. In both scheduling algorithms, sensors move simultaneously according to a movement schedule. In comparison with the strategy where sensors move sequentially, our scheme adapts to the spatiotemporally evolving process, improving the temporal resolution of profiling.

7.1 Greedy Movement Scheduling

Gradient ascent is a widely adopted approach to find a local maximum of a utility function. In this paper, we propose a greedy movement scheduling algorithm based on the gradient ascent approach. We first discuss how to determine the movement orientations for the sensors. Since the profiling accuracy \( \omega \) given by Eq. (8) is a function of all sensors’ positions, we can compute the gradient of \( \omega \) with respect to the position of sensor \( i \) (denoted by \( \nabla \omega \)), which is formally given by \( \nabla \omega = \begin{bmatrix} \frac{\partial \omega}{\partial x_i} & \frac{\partial \omega}{\partial y_i} \end{bmatrix}^T \). When all sensors except sensor \( i \) remain stationary, the metric \( \omega \) will increase the fastest if sensor \( i \) moves in the orientation given by \( \nabla \omega \). Therefore, in the greedy movement scheduling algorithm, we let \( \phi_i = \angle(\nabla \omega_i) \). Note that sensors will move simultaneously when the movement schedule is executed. We now discuss how to allocate the movement steps. The magnitude of \( \nabla \omega_i \), denoted by \( \| \nabla \omega_i \| \), quantifies the steepness of the metric \( \omega \) when sensor \( i \) moves in the orientation \( \angle(\nabla \omega_i) \) while other sensors remain stationary. Therefore, in the greedy algorithm, we propose to proportionally allocate the movement steps according to sensor’s gradient magnitude. Specifically, \( m_i \) is given by \( m_i = \min \left\{ \left\lfloor \frac{\| \nabla \omega_i \|}{\sum_{i=1}^{N} \| \nabla \omega_i \|} \cdot M \right\rfloor, \lceil \frac{L_i}{l} \rceil \right\} \). Note that the \( \left\lceil \frac{L_i}{l} \right\rceil \) in the min operator satisfies the constraint Eq. (11). This greedy algorithm has linear complexity, i.e., \( O(N) \), which is preferable for the cluster head with limited computational resource.

7.2 Radial Movement Scheduling

In this section, we propose a new movement scheduling algorithm based on the approximations discussed in Section 5.4. From Eq. (9), the contribution of sensor \( i, \omega_i \), depends on the minimum distance between the cluster head and other sensors. Because of such inter-node dependence, it is difficult to derive the optimal distance for each sensor that maximizes the overall profiling accuracy \( \omega \). It can be shown that the problem involves non-linear and non-convex constrained optimization. Several stochastic search algorithms, such as simulated annealing, can find near-optimal solutions. However, these algorithms often have prohibitively high complexities. In our algorithm, we fix the sensor closest to the estimated source location and only schedule the movements of other sensors in each profiling iteration. As the sensor closest to the source receives the highest SNR, moving other sensors will likely yield more performance gain. Moreover, this sensor can serve as the cluster head that
receives measurements from other sensors and computes the movement schedule. It is hence desirable to keep it stationary due to its higher energy consumption in computation and communication. We note that the sensor closest to the source may be different in each iteration after sensor movements, resulting in rotation of cluster head among sensors. By fixing the sensor closest to the source, the distance \( d_i \) that maximizes the expected \( \omega_i \), denoted by \( d^* \), can be directly calculated by

\[
d^*_i = \sqrt{\frac{1}{2\beta} + \min_{j \in [1,N]} d^2_j}, \quad \forall i \neq \arg \min_j d_j. \tag{12}
\]

Note that as \( \beta \) is a time-dependent variable, \( d^* \) also changes with time and hence should be updated in each profiling iteration. Eq. (12) allows us to easily determine the movement orientation of sensor \( i \). Specifically, if \( d_i > d^*_i \), sensor \( i \) will move toward the estimated source location; otherwise, sensor \( i \) will move in the opposite direction. Formally, by defining \( \delta = \text{sgn}(d_i^* - d_i) \), we can express the movement orientation of sensor \( i \) as \( \phi_i = \angle(\mathbf{\delta} \cdot x_i, \mathbf{\delta} \cdot y_i) \).

We now discuss how to allocate the movement steps. In the rest of this section, when we refer to sensor \( i \), we assume sensor \( i \) is not the closest to the estimated source location. After sensor \( i \) moves \( m_i \) steps in the orientation of \( \phi_i \), its contribution to the overall profiling accuracy is

\[
\omega_i(m_i) = (f_i + \beta \cdot m_i \cdot \ell^2 - \min_{j \in [1,N]} d^2_j / \sigma^2) \cdot \text{erf}(\frac{\mathbf{\ell}}{\sqrt{\pi} D t})^2, \tag{13}
\]

where \( \min_{j \in [1,N]} d^2_j \) in Eq. (13) is a constant for sensor \( i \), and \( \beta \) can be predicted based on its current estimate to capture the temporal evolution of the diffusion, i.e., \( \beta = (1/\beta + 4 \cdot D \cdot \tau)^{-1} \). Given the radial movement orientations described earlier, the formulated problem is equivalent to maximizing \( \sum_i \omega_i(m_i) \) subject to the constraints Eqs. (10) and (11), which can be solved by a dynamic programming algorithm as follows.

We number the sensors by \( 1, 2, \ldots, N-1 \), excluding the sensor closest to the estimated source location. Let \( \Omega(i,m) \) be the maximum \( \omega \) when the first \( i \) sensors are allocated with \( m \) steps. The dynamic programming recursion that computes \( \Omega(i,m) \) can be expressed as

\[
\Omega(i,m) = \max_{0 \leq m_i \leq [L_i/m]} \{ \Omega(i-1, m - m_i) + \omega_i(m_i) \}.
\]

The initial condition of the above recursion is \( \Omega(0,0) = 0 \) for \( m \in [0,M] \). According to the above equation, at the \( i \)th iteration of the recursion, the optimal value of \( \Omega(i,m) \) is computed as the maximum value of \([L_i/m]\) cases which have been computed in previous iterations of the recursion. Specifically, for the case where sensor \( i \) moves \( m_i \) steps, the maximum profiling accuracy \( \omega \) of the first \( i \) sensors allocated with \( m \) steps can be computed as \( \Omega(i-1, m - m_i) + \omega_i(m_i) \), where \( \Delta \Omega(i-1, m - m_i) \) is the maximum \( \omega \) of the first \( i-1 \) sensors allocated with \( m-m_i \) steps. The maximum overall profiling accuracy is given by \( \omega^* = \max_{m \in [1,M]} \Omega(N-1, m) \).

We now discuss how to construct the movement schedule using the above dynamic programming recursion. The movement schedule of sensor \( i \) is represented by a pair \( (i, m_i) \). For each \( \Omega(i,m) \), we define a movement schedule \( S(i,m) \) initialized to be an empty set. The set \( S(i,m) \) is filled incrementally in each iteration when \( \Omega(i,m) \) is computed. Specifically, in the \( i \)th iteration of the recursion, if \( \Omega(i-1, m - m_x) + \omega_i(m_x) \) gives the maximum value among all cases, we add a movement schedule \( (i, m_x) \) to \( S(i,m) \). Formally, \( S(i,m) = S(i-1, m - m_x) \cup \{(i,m_x)\} \), where \( m_x = \arg \max_{\Omega(i-1, m - m_x) + \omega_i(m_x)} \{(i,m_x)\} \). The dynamic programming complexity is \( O((N-1)M^2) \), where \( N \) and \( M \) are the number of sensors and allocatable movement steps in a profiling iteration, respectively.

8 Discussions

In this section, we discuss several issues that have not been addressed in the previous sections.

8.1 Diffusion Process with Continuous Source

In previous sections, we assume that the diffusion process is caused by an impulse source. We now extend our approach to address the continuous source, where the discharged substance is released continuously at a certain rate. Formally, the initial condition of continuous source can be expressed by the step function, i.e., \( c(0,t) = \mu \cdot 1(t) \). In this discussion, we assume that there is no advection. The solution to Eq. (1) with this new initial condition can be obtained by integrating Eq. (2) over time \( t \), yielding \( c(d,t) = \frac{\mu}{\pi D} \cdot \text{erf}(\frac{d}{2\sqrt{Dt}}) \), where \( \text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_{-\infty}^{x} \exp(-y^2) dy \). However, the closed-form \( 4 \times 4 \text{ FIM} \) for \( x_0, y_0, t \) and \( \mu \) is too complicated to derive the decomposition of the CRB-based profiling accuracy metric (i.e., \( \omega \)) into the sum of contributions of individual sensors. We note that the decomposition is the key property that allows us to apply dynamic programming to schedule the sensor movements. To make the problem tractable, we only consider the FIM of the source location, which is a \( 2 \times 2 \) diagonal matrix given by \( J = \begin{bmatrix} J_{11} & J_{21} \\ J_{21} & J_{22} \end{bmatrix} \), where

\[
J_{11} = \sum_{i=1}^{N} \left( f^2(d_i) \ell^2 / (4\pi D \sigma^2 \mu^2) \right), \quad J_{21} = \sum_{i=1}^{N} f^2(d_i) y_i, \quad J_{22} = \sum_{i=1}^{N} f^2(d_i) y_i^2 / (4\pi D \sigma^2 / \mu^2),
\]

and \( f(d_i) = d_i^{-2}(\pi Dt)^{-\frac{3}{2}} \exp(-\frac{d^2}{4Dt}) + \text{erfc}(\frac{d}{2\sqrt{2Dt}}) \). We define an \( 1 \times N \) vector denoted by \( \mathbf{L}_X \) where the \( i \)th element of \( \mathbf{L}_X \) is given by \( L_X(i) = f(d_i) x_i \cdot \sigma^{-1} \). We define another \( 1 \times N \) vector denoted by \( \mathbf{L}_Y \) by replacing \( x_i \) in \( \mathbf{L}_X \) with \( y_i \). The CRB \( (x_0) \) and CRB \( (y_0) \), which are given by \( J_{X,i}^{-1} \) and \( J_{Y,i}^{-1} \), can be derived as

\[
\text{CRB}(x_0) = \frac{(\mu/4\pi D)^2}{L_X L_Y T_X T_Y - (L_X L_Y T_X T_Y) \cdot L_X L_Y T_X T_Y}, \quad \text{CRB}(y_0) = \frac{(\mu/4\pi D)^2}{L_Y L_X T_X T_Y - (L_X L_Y T_X T_Y) \cdot L_X L_Y T_X T_Y}.
\]

Similar to Eq. (8), we define the profiling accuracy metric \( \omega \) based on the reciprocals of CRBs:

\[
\omega = \frac{(\mu/4\pi D)^2}{\text{CRB}(x_0)} + \frac{(\mu/4\pi D)^2}{\text{CRB}(y_0)} = (1 - \epsilon) \left( L_X T_X + L_Y T_Y \right),
\]
The network needs to output the quantified profiling decomposition in Eq. (14), the applied as long as the closed-form formula of the profiling accuracy decreases with the noise level, which is consistent with our complexity analysis. The radial algorithm takes about 100 seconds to compute the movement schedule in a profiling iteration when $N = 20$. This overhead is reasonable compared with the movement delay of low-speed mobile sensors. The greedy algorithm is significantly faster, and hence provides an efficient solution when the timeliness is more important than profiling accuracy. For the radial algorithm, 30% execution time is spent on computing a look-up table consisting of each sensor $i$’s contribution, i.e., $\omega_i(m_i)$ in Eq. (13), given all possible values of $m_i$. There are several ways to further reduce the computational overhead. First, our current implementation employs extensive floating-point computation. Our previous experience shows that fixed-point arithmetic is significantly more efficient on TelosB motes. Moreover, we can also adopt more powerful sensor platforms as cluster head in the network. For instance, the projected execution time on Imote2 [33] equipped with a 416 MHz processor is $416$ seconds for computing the movement schedule in a profiling iteration when $N = 20$. This overhead is reasonable compared with the movement delay of low-speed mobile sensors.

9.2 Overhead on Sensor Hardware

We have implemented the MLE and the two movement scheduling algorithms in TinyOS 2.1.1 on TelosB platform [33] equipped with an 8 MHz processor. We note that these algorithms are executed on the cluster head. We ported the C implementation of the Nelder-Mead algorithm [30] in GNU Scientific Library (GSL) [34] to TinyOS to solve the optimization problem in MLE (see Section 5.1). The porting is non-trivial because dynamic memory allocation and function pointer are extensively used in GSL while these features are not available in TinyOS. Our implementation of MLE requires 19 KB ROM and 1 KB RAM. When 10 sensors are to be scheduled, the two movement scheduling algorithms require 1 and 8.8 KB RAM, respectively. Fig. 7 plots the average execution time of the MLE, greedy and radial algorithms versus the number of sensors. We note that the complexity of MLE is $O(N)$. For both movement scheduling algorithms, the execution time linearly increases with $N$, which is consistent with our complexity analysis. The radial algorithm takes about 100 seconds to compute the movement schedule in a profiling iteration when $N = 20$. This overhead is reasonable compared with the movement delay of low-speed mobile sensors. The greedy algorithm is significantly faster, and hence provides an efficient solution when the timeliness is more important than profiling accuracy. For the radial algorithm, 30% execution time is spent on computing a look-up table consisting of each sensor $i$’s contribution, i.e., $\omega_i(m_i)$ in Eq. (13), given all possible values of $m_i$. There are several ways to further reduce the computational overhead. First, our current implementation employs extensive floating-point computation. Our previous experience shows that fixed-point arithmetic is significantly more efficient on TelosB motes. Moreover, we can also adopt more powerful sensor platforms as cluster head in the network. For instance, the projected execution time on Imote2 [33] equipped with a 416 MHz processor is within 2 seconds for computing the movement schedule for 20 sensors.

We note that the complexity of radial algorithm is $O(N^3)$, which can jeopardize the timeliness of periodical profiling when more sensors are deployed. To reduce the computation delay, in this paper we adopt a simple clustering method by randomly assigning $N$ nodes to present the results in Section 9.3, which proceeds as follows.

a) Section 9.3.1 presents how we collect the three sets of data traces, which include GPS localization errors, robotic fish movement, and on-water Zigbee wireless communication.

b) Section 9.3.2 presents the simulation settings and methodology.

c) Section 9.3.3 to Section 9.3.7 present the simulation results where we analyze the impact of several important factors on the profiling accuracy.

9.2 Overhead on Sensor Hardware

We have implemented the MLE and the two movement scheduling algorithms in TinyOS 2.1.1 on TelosB platform [33] equipped with an 8 MHz processor. We note that these algorithms are executed on the cluster head. We ported the C implementation of the Nelder-Mead algorithm [30] in GNU Scientific Library (GSL) [34] to TinyOS to solve the optimization problem in MLE (see Section 5.1). The porting is non-trivial because dynamic memory allocation and function pointer are extensively used in GSL while these features are not available in TinyOS. Our implementation of MLE requires 19 KB ROM and 1 KB RAM. When 10 sensors are to be scheduled, the two movement scheduling algorithms require 1 and 8.8 KB RAM, respectively. Fig. 7 plots the average execution time of the MLE, greedy and radial algorithms versus the number of sensors. We note that the complexity of MLE is $O(N)$. For both movement scheduling algorithms, the execution time linearly increases with $N$, which is consistent with our complexity analysis. The radial algorithm takes about 100 seconds to compute the movement schedule in a profiling iteration when $N = 20$. This overhead is reasonable compared with the movement delay of low-speed mobile sensors. The greedy algorithm is significantly faster, and hence provides an efficient solution when the timeliness is more important than profiling accuracy. For the radial algorithm, 30% execution time is spent on computing a look-up table consisting of each sensor $i$’s contribution, i.e., $\omega_i(m_i)$ in Eq. (13), given all possible values of $m_i$. There are several ways to further reduce the computational overhead. First, our current implementation employs extensive floating-point computation. Our previous experience shows that fixed-point arithmetic is significantly more efficient on TelosB motes. Moreover, we can also adopt more powerful sensor platforms as cluster head in the network. For instance, the projected execution time on Imote2 [33] equipped with a 416 MHz processor is within 2 seconds for computing the movement schedule for 20 sensors.

We note that the complexity of radial algorithm is $O(N^3)$, which can jeopardize the timeliness of periodical profiling when more sensors are deployed. To reduce the computation delay, in this paper we adopt a simple clustering method by randomly assigning $N$ nodes to

9.2 Overhead on Sensor Hardware

We have implemented the MLE and the two movement scheduling algorithms in TinyOS 2.1.1 on TelosB platform [33] equipped with an 8 MHz processor. We note that these algorithms are executed on the cluster head. We ported the C implementation of the Nelder-Mead algorithm [30] in GNU Scientific Library (GSL) [34] to TinyOS to solve the optimization problem in MLE (see Section 5.1). The porting is non-trivial because dynamic memory allocation and function pointer are extensively used in GSL while these features are not available in TinyOS. Our implementation of MLE requires 19 KB ROM and 1 KB RAM. When 10 sensors are to be scheduled, the two movement scheduling algorithms require 1 and 8.8 KB RAM, respectively. Fig. 7 plots the average execution time of the MLE, greedy and radial algorithms versus the number of sensors. We note that the complexity of MLE is $O(N)$. For both movement scheduling algorithms, the execution time linearly increases with $N$, which is consistent with our complexity analysis. The radial algorithm takes about 100 seconds to compute the movement schedule in a profiling iteration when $N = 20$. This overhead is reasonable compared with the movement delay of low-speed mobile sensors. The greedy algorithm is significantly faster, and hence provides an efficient solution when the timeliness is more important than profiling accuracy. For the radial algorithm, 30% execution time is spent on computing a look-up table consisting of each sensor $i$’s contribution, i.e., $\omega_i(m_i)$ in Eq. (13), given all possible values of $m_i$. There are several ways to further reduce the computational overhead. First, our current implementation employs extensive floating-point computation. Our previous experience shows that fixed-point arithmetic is significantly more efficient on TelosB motes. Moreover, we can also adopt more powerful sensor platforms as cluster head in the network. For instance, the projected execution time on Imote2 [33] equipped with a 416 MHz processor is within 2 seconds for computing the movement schedule for 20 sensors.

We note that the complexity of radial algorithm is $O(N^3)$, which can jeopardize the timeliness of periodical profiling when more sensors are deployed. To reduce the computation delay, in this paper we adopt a simple clustering method by randomly assigning $N$ nodes to
The average of the profiling results of all clusters is yielded as the final result. The complexity for a cluster reduces to $O(N^3/p^3)$. Due to space limitation, the details of the clustering method and its performance analysis are omitted here and can be found in [24].

Fig. 8 plots the execution time and profiling accuracy of the radial approach when the network is divided into $p$ clusters. The left Y-axis is the ratio of execution time for $p$ clusters with respect to the case of a single cluster. We can see that both the execution time and profiling accuracy decrease with $p$. This is because simply averaging results from all clusters does not fully account for the inter-cluster dependence in the accuracy of dynamic programming. Nevertheless, the radial algorithm of 2 and 3 clusters still outperforms the greedy algorithm of a single cluster in terms of profiling accuracy.

9.3 Trace-Driven Simulations

9.3.1 Trace Collection

We collect three sets of data traces, which include GPS localization errors, robotic fish movement, and on-water Zigbee wireless communication. First, the data traces of GPS error are collected using two Linx GPS modules [35] in outdoor open space. We extract the GPS error by comparing the distance measured by GPS modules with the groundtruth distance. The average GPS error is 2.29 meters. Second, the data traces of movement control are collected with a robotic fish developed in our lab [6] (see Fig. 1). The movement of robotic fish is driven by a servo motor that is controlled by continuous pulse-width modulation waves. By setting the fish tail beating amplitude and frequency to $23^\circ$ and 0.9 Hz, the movement speed is 2.5 m/min. We then have the fish swim along a fixed direction in an experimental water tank, and derive the real speed by dividing the moving distance by elapsed time. Third, the data traces of Zigbee communication are collected with two IRIS motes [33] by measuring the packet reception rate (PRR) of a single link on the wavy water surface of Lake Lansing, Michigan, on a windy day. We note that the PRR in such wavy water environment is more dynamic than that in calm water environment, due to multipathing [36]. Specifically, we place the two motes about 12 centimeters above the water surface, and measure the PRR versus their distance. The results are plotted in Fig. 9. We note that the two IRIS motes achieve an average PRR of 0.8, when they are 37 meters apart. According to our experience, the communication range of IRIS on water surface drops by about 50% compared to that on land.

9.3.2 Simulation Settings and Methodologies

We conduct extensive simulations based on collected data traces to evaluate the effectiveness of our approach. The simulation programs are written in Matlab. As discussed in Section 3.2, the effect of constant-speed advection is canceled because the sensors and source location are in the same inertial system. Therefore, we only simulate the diffusion process without advection. The diffusion source is at the origin of the coordinate system, i.e., $x_0 = y_0 = 0$. The sensors are randomly deployed in the square region of $200 \times 200$ m$^2$ centered at the origin. The reading of a sensor is set to be the sum of the concentration calculated from Eq. (2), the bias $b_i$, and a random number sampled from the normal distribution $N(0, \varsigma^2)$. Without loss of generality, we assume that the bias $b_i$ is zero in the simulations. As discussed in Section 3.2, in each profiling iteration, a sensor samples $K$ readings and outputs the average of them as the measurement. The amount of discharged substance is set to be $A = 0.70 \times 10^6$ cm$^3$ (i.e., $0.7$ m$^3$) unless otherwise specified. The diffusion coefficient is set to be $D = 5.000$ cm$^2$/s. Note that the settings of $A$ and $D$ are comparable to the real field experiments reported in [21] where 2 to 5 m$^3$ of diesel oil were discharged into the sea and the estimated diffusion coefficient ranged from 2,000 cm$^2$/s to 7,000 cm$^2$/s. The noise standard deviation is set to be $\varsigma = 1$ cm$/m^2$, i.e., 1 cm$^3$ discharged substance per unit area. To easily compare various movement scheduling algorithms, we let the first profiling iteration always start at $t = 1800$ s, i.e., half an hour after the discharge. At $t = 1800$ s, the average received SNR is around 10/1. The rationale of this setting is that moving sensors too early (i.e., at low SNRs) leads to little

As we adopt a 2-dimensional model to characterize the diffusion process, the physical unit of concentration is cm$^3$/m$^2$. As observed in the field experiments [21], diesel oil can penetrate down to several meters from the water surface. As a result, the equivalent $\varsigma$ that accounts for the depth dimension ranges from 0.1 cm$^3$/m$^3$ to 1 cm$^3$/m$^3$. Our setting is consistent with the noise standard derivation of the crude oil sensor Cyclops-7 [25], which is 0.1 cm$^3$/m$^3$.

3. As we adopt a 2-dimensional model to characterize the diffusion process, the physical unit of concentration is cm$^3$/m$^2$. As observed in the field experiments [21], diesel oil can penetrate down to several meters from the water surface. As a result, the equivalent $\varsigma$ that accounts for the depth dimension ranges from 0.1 cm$^3$/m$^3$ to 1 cm$^3$/m$^3$. Our setting is consistent with the noise standard derivation of the crude oil sensor Cyclops-7 [25], which is 0.1 cm$^3$/m$^3$. 

---

**Fig. 7.** Execution time on TelosB vs. the number of sensors $N$. **Fig. 8.** Execution time and profiling accuracy vs. the number of clusters $p$. **Fig. 9.** PRR (90% confidence) vs. distance.
complexity with respect to the number of sensors. It then employs force search to find the optimal step allocation under the constraints in Eqs. (10) and (11). The first baseline (referred to as "trace-driven") in the evaluation. The first baseline (referred to as "trace-driven") schedules the movements based on the SNRs received by sensors. The SNR received by sensor $i$ (denoted by $SNR_i$) is defined as $c(d_i, t)/\sigma$, where $d_i$ and $t$ can be computed from the estimated profile $\Theta$. In the SNR-based scheduling algorithm, the sensors always move toward the estimated source location to increase the received SNRs. The movement steps are proportionally allocated according to sensors’ SNRs. The rationale behind this heuristic is that the accuracy of MLE increases with SNR. The second baseline (referred to as "annealing") is based on the simulated annealing. For given movement orientations $\{\phi_i | \forall i\}$, it uses the brutal-force search to find the optimal step allocation under the constraints in Eqs. (10) and (11). It then employs a simulated annealing algorithm to search for the optimal movement orientations. However, it has exponential complexity with respect to the number of sensors.

We now discuss how the real data traces are used in the simulations. We compare the performance of the greedy and radial algorithms with and without the data traces of movement and localization errors. If the movement/localization traces are used, the movement speed of a robotic sensor in the simulation is set to be the real speed that is randomly selected from the movement data traces, and the position reading sent to the cluster head is corrupted by a localization error that is randomly selected from the GPS error traces. The simulation results with movement/localization traces presented in Section 9.3.3 to 9.3.6 are labeled with the prefix “trace-driven”. In Section 9.3.7, to evaluate the communication overhead of our approach, the PRR of a link in the simulation is set to be the distance-based interpolation of real PRR measurements shown in Fig. 9.

We compare our approach with two additional baseline algorithms in the evaluation. The first baseline (referred to as SNR-based) schedules the movements based on the SNRs received by sensors. The SNR received by sensor $i$ (denoted by $SNR_i$) is defined as $c(d_i, t)/\sigma$, where $d_i$ and $t$ can be computed from the estimated profile $\Theta$. In the SNR-based scheduling algorithm, the sensors always move toward the estimated source location to increase the received SNRs. The movement steps are proportionally allocated according to sensors’ SNRs. The rationale behind this heuristic is that the accuracy of MLE increases with SNR. The second baseline (referred to as "annealing") is based on the simulated annealing. For given movement orientations $\{\phi_i | \forall i\}$, it uses the brutal-force search to find the optimal step allocation under the constraints in Eqs. (10) and (11). It then employs a simulated annealing algorithm to search for the optimal movement orientations. However, it has exponential complexity with respect to the number of sensors.

We first visually compare the sensor movement trajectories computed by the greedy and radial movement scheduling algorithms. A total of 20 sensors are deployed. Fig. 10 shows the movement trajectories of sensors in the first 15 profiling iterations. For a particular sensor, the circle denotes its initial position, the segments represent its movement trajectory of 15 profiling iterations, and the arrow indicates its movement orientation in the 15th iteration. The sensor with no segments remains stationary during all 15 profiling iterations. We can see that, with the greedy algorithm, several sensors (e.g., sensor 15, 16, and 17) have bent trajectories. This is because the movement orientation of each sensor is to maximize the gradient ascent of $\omega$, hence not necessarily to be aligned along the iterations. In the radial algorithm, sensors’ trajectories are more straight. This is because the movement orientation is along the direction determined by the current sensor position and the estimated source location that is close to the true source location. Moreover, we find that the radial algorithm outperforms the greedy algorithm in terms of profiling accuracy after the first 15 iterations. In the greedy algorithm, the orientation assignment and movement step allocation are based on the gradient derived from the current positions of sensors. Besides, the greedy algorithm does not account for the interdependence of sensors in providing the overall profiling accuracy. As a result, its solution may not lead to the maximum $\omega$ after the sensor movements and the temporal evolution of diffusion.

We now discuss how the real data traces are used in the simulations. We compare the performance of the greedy and radial algorithms with and without the data traces of movement and localization errors. If the movement/localization traces are used, the movement speed of a robotic sensor in the simulation is set to be the real speed that is randomly selected from the movement data traces, and the position reading sent to the cluster head is corrupted by a localization error that is randomly selected from the GPS error traces. The simulation results with movement/localization traces presented in Section 9.3.3 to 9.3.6 are labeled with the prefix “trace-driven”. In Section 9.3.7, to evaluate the communication overhead of our approach, the PRR of a link in the simulation is set to be the distance-based interpolation of real PRR measurements shown in Fig. 9.

We compare our approach with two additional baseline algorithms in the evaluation. The first baseline (referred to as SNR-based) schedules the movements based on the SNRs received by sensors. The SNR received by sensor $i$ (denoted by $SNR_i$) is defined as $c(d_i, t)/\sigma$, where $d_i$ and $t$ can be computed from the estimated profile $\Theta$. In the SNR-based scheduling algorithm, the sensors always move toward the estimated source location to increase the received SNRs. The movement steps are proportionally allocated according to sensors’ SNRs. The rationale behind this heuristic is that the accuracy of MLE increases with SNR. The second baseline (referred to as "annealing") is based on the simulated annealing. For given movement orientations $\{\phi_i | \forall i\}$, it uses the brutal-force search to find the optimal step allocation under the constraints in Eqs. (10) and (11). It then employs a simulated annealing algorithm to search for the optimal movement orientations. However, it has exponential complexity with respect to the number of sensors.

9.3.3 Sensor Movement Trajectories

We now discuss how the real data traces are used in the simulations. We compare the performance of the greedy and radial algorithms with and without the data traces of movement and localization errors. If the movement/localization traces are used, the movement speed of a robotic sensor in the simulation is set to be the real speed that is randomly selected from the movement data traces, and the position reading sent to the cluster head is corrupted by a localization error that is randomly selected from the GPS error traces. The simulation results with movement/localization traces presented in Section 9.3.3 to 9.3.6 are labeled with the prefix “trace-driven”. In Section 9.3.7, to evaluate the communication overhead of our approach, the PRR of a link in the simulation is set to be the distance-based interpolation of real PRR measurements shown in Fig. 9.

We now discuss how the real data traces are used in the simulations. We compare the performance of the greedy and radial algorithms with and without the data traces of movement and localization errors. If the movement/localization traces are used, the movement speed of a robotic sensor in the simulation is set to be the real speed that is randomly selected from the movement data traces, and the position reading sent to the cluster head is corrupted by a localization error that is randomly selected from the GPS error traces. The simulation results with movement/localization traces presented in Section 9.3.3 to 9.3.6 are labeled with the prefix “trace-driven”. In Section 9.3.7, to evaluate the communication overhead of our approach, the PRR of a link in the simulation is set to be the distance-based interpolation of real PRR measurements shown in Fig. 9.

We now discuss how the real data traces are used in the simulations. We compare the performance of the greedy and radial algorithms with and without the data traces of movement and localization errors. If the movement/localization traces are used, the movement speed of a robotic sensor in the simulation is set to be the real speed that is randomly selected from the movement data traces, and the position reading sent to the cluster head is corrupted by a localization error that is randomly selected from the GPS error traces. The simulation results with movement/localization traces presented in Section 9.3.3 to 9.3.6 are labeled with the prefix “trace-driven”. In Section 9.3.7, to evaluate the communication overhead of our approach, the PRR of a link in the simulation is set to be the distance-based interpolation of real PRR measurements shown in Fig. 9.

We now discuss how the real data traces are used in the simulations. We compare the performance of the greedy and radial algorithms with and without the data traces of movement and localization errors. If the movement/localization traces are used, the movement speed of a robotic sensor in the simulation is set to be the real speed that is randomly selected from the movement data traces, and the position reading sent to the cluster head is corrupted by a localization error that is randomly selected from the GPS error traces. The simulation results with movement/localization traces presented in Section 9.3.3 to 9.3.6 are labeled with the prefix “trace-driven”. In Section 9.3.7, to evaluate the communication overhead of our approach, the PRR of a link in the simulation is set to be the distance-based interpolation of real PRR measurements shown in Fig. 9.
have little impact on our approach. The radial algorithm outperforms the greedy and SNR-based algorithms by 16\% and 50\% in terms of \( \omega \) at the 15\textsuperscript{th} profiling iteration, respectively. And the accuracy performance of the radial algorithm is very close to the annealing algorithm that can find the near-optimal solution. However, we note that in each iteration of the annealing algorithm, a new look-up table needs to be computed due to changed movement orientations. Hence its execution time highly depends on the number of iterations that can be very large. Therefore, the annealing algorithm is infeasible on mote-class platforms.

Fig. 12 plots the average of \( \text{Var}(\tilde{x}_0) \) and \( \text{Var}(\tilde{y}_0) \) in each profiling iteration under various settings of the discharged substance amount \( A \). In order to evaluate the variances in each profiling iteration, the sensors perform many rounds of MLE, where each round yields a pair of \((\tilde{x}_0, \tilde{y}_0)\). The \( \text{Var}(\tilde{x}_0) \) and \( \text{Var}(\tilde{y}_0) \) are calculated from all rounds. From Fig. 12, we find that the variances may increase (for the SNR-based scheduling algorithm) or fluctuate (for other approaches) after several iterations. This is because the variances are time-dependent due to the involving of \( \alpha \) and \( \beta \) in CRB\((\tilde{x}_0)\) and CRB\((\tilde{y}_0)\). Moreover, we can see that the variances decrease with \( A \). As sensors receive higher SNRs in the case of higher \( A \), our result consists with the intuition that the estimation error decreases with SNR. Compared with the SNR-based algorithm, the radial algorithm reduces the variance in estimating diffusion source location by 36\% for \( A = 0.7 \times 10^6 \) cm\(^3\). Compared with the greedy algorithm, the reductions are 12\% and 18\% for \( A = 0.7 \times 10^6 \) and \( 1.4 \times 10^6 \) cm\(^3\), respectively.

The MLE algorithm also estimates the initial substance amount \( A \) and elapsed time \( t \). Therefore, we additionally evaluate the performance of our movement scheduling algorithms on profiling \( A \) and \( t \). Fig. 13 plots \( t \) versus the groundtruth time. We can see that the elapsed time can be accurately estimated. For instance, the relative error of \( t \) for radial algorithm is only 1.0\%. Fig. 14 plots the relative error of \( A \), which is calculated as \( |(\tilde{A} - A)/A| \). We can see that both the radial and greedy algorithms give comparable profiling performance with the annealing algorithm. The relative error is less than 1.4\%. Moreover, the result shows that the greedy algorithm achieves a slightly better profiling performance than the radial algorithm. As discussed in Section 5.3, the accuracy metric adopted in this paper is defined to characterize the accuracy of localizing the diffusion source. Therefore, it is possible that radial algorithm yields larger error than greedy algorithm in profiling \( A \).

9.3.5 Sampling Scheme, Source Bias, Network Density

We characterize the profiling error after 15 iterations by the average of \( \text{Var}(\tilde{x}_0) \) and \( \text{Var}(\tilde{y}_0) \). Except for the evaluations on network density, a total of 10 sensors are deployed. In the temporal sampling scheme presented in Section 3.2, a sensor yields the average of \( K \) continuous samples as the measurement to reduce noise variance. Fig. 15 plots the profiling error versus \( K \). We can see that the profiling error decreases with \( K \). The relative reductions of profiling error by the radial algorithm with respect to the greedy and SNR-based algorithms are about 18\% and 30\%, respectively, when \( K \) ranges from 2 to 20.

The approximations discussed in Section 5.3 assume that the sensors are randomly deployed around the diffusion source. In this set of simulations, we evaluate the impact of source location bias on profiling accuracy. Specifically, the diffusion source appears at \((\delta, 0)\), where \( \delta \) is referred to as the source location bias. Fig. 16 plots the profiling error versus \( \delta \). To jointly account for the impact of random sensor deployment, for each setting of \( \delta \), we deploy a number of networks and show the error bars. We find that the radial algorithm is robust to the source location bias. Moreover, we note that the radial algorithm is consistently better than other algorithms.

Fig. 17 plots the profiling error versus the number of sensors. When more sensors are deployed, the profiling error can be reduced. The radial algorithm is consistently better than the other algorithms. For all algorithms, the profiling error is reduced by about 40\% when the number of sensors increases from 10 to 15. Moreover, the relative reduction of profiling error decreases with the number of sensors.
9.3.6 Impact of Sensor Deployment

In this section, we evaluate the impact of initial sensor deployment on the profiling accuracy and energy consumption in locomotion. We fix each $d_i$ and randomly deploy sensors in one, two adjacent, three and four quadrants of the plane originated at the source location, resulting in four sensor deployments. We compute the upper bound of $\omega$, in which sensors’ angles with respect to the source location are exhaustively searched to maximize the profiling accuracy. Note that the sensor deployment with maximized profiling accuracy is still an open issue. Fig. 18 plots the upper bound of $\omega$ as well as the profiling accuracy of four sensor deployments. We can see that the profiling accuracy of the four-quadrant deployment is the closest to the upper bound. Fig. 18 also plots the minimum total distance that the sensors in a deployment have to move to achieve the upper bound $\omega$. We can observe that if sensors are not deployed around the source location, spreading sensors first can significantly improve the profiling accuracy. However, if sensors have limited energy for locomotion, it is more beneficial to deploy sensors around the source to avoid energy-consuming spreading movements.

9.3.7 Communication Overhead

In each profiling iteration, the communication overhead is mainly affected by the packet loss caused by the unreliable on-water wireless communication. Hence, we employ the total number of transmissions in collecting all sensor measurements as the evaluation metric. Specifically, we choose the shortest distance path as the routing path from a sensor to the cluster head, where the distance metric of each hop is $\text{PRR}^{-1}$, i.e., the expected number of (re-)transmissions on the hop. The PRR is set to be the distance-based interpolation of real measurements shown in Fig. 9. When a node transmits packet to the next hop, the packet is delivered with a success probability equal to the PRR. The node re-transmits the packet up to 10 times before it is dropped. In the simulations, 30 sensors are randomly deployed. The packet to the cluster head includes sensor ID, current position and measurement. The packet to sensor contains the movement schedule that includes movement orientation and distance. Our simulation results show that the number of (re-)transmissions in a profiling iteration has a mean of 158 and a standard deviation of 28. Even if all these transmissions happen sequentially, the delay will be within seconds, because transmitting a TinyOS packet only takes about 10 ms on typical mote-class platforms. This result shows that our approach has low communication overhead under realistic settings.

10 Conclusion and Future Work

In this paper we propose an accuracy-aware profiling approach for aquatic diffusion processes using robotic sensor networks. Our approach features an iterative profiling process where the sensors reposition themselves to progressively improve the profiling accuracy along the iterations. We develop two movement scheduling algorithms, including an efficient greedy algorithm and a near-optimal radial algorithm. We implement our algorithms on TelosB motes and evaluate their overhead. We also conduct extensive simulations based on real traces of GPS localization errors, robotic fish movement, and wireless communication. Our results show that our approach can accurately profile dynamic diffusion processes with low overhead.

The movement scheduling approach described in this paper is targeted at robotic sensors with limited sensing and motion capabilities in relatively calm water environment. We are developing the next generation of our robotic fish platforms that are capable of more complex sensing and motion control. In our future work, we will investigate distributed control algorithms that allow such robotic sensors to autonomously plan their motion paths, which reduce the overhead of cross-sensor coordination in collaborative sensing tasks.

Acknowledgment

This research was supported in part by the U.S. National Science Foundation under grants ECCS-1029683, CNS-0954039 (CAREER), IIS-0916720, and ECCS-1050236. We thank Ruogu Zhou for his contribution in traces collection and hardware design.

References
