Mobile Sensor Coordination for Optimal Coverage of Weather Data Sources

Irené Tematelewo
College of Engineering
Oregon State University
Corvallis, USA
temateli@oregonstate.edu

Khushal Brahmbhatt
College of Engineering
Oregon State University
Corvallis, USA
brahmbhk@oregonstate.edu

***Abstract*—Weather forecasting requires high quality weather data which requires robust sensor networks that are inexpensive to maintain and can adapt to changing weather patterns. Previous works have shown how multiagent approaches can be used to achieve optimal coverage in distributed sensor networks. We aim to learn control policies that allow multiple mobile sensor agents to coordinate and move to optimal locations in an environment such that they can achieve maximum coverage of data sources in the most efficient way. We use a neuroevolutionary algorithm that is evaluated using a density-based coverage function to find policies that allow mobile sensor agents to find the optimal path around obstacles to an optimal location that achieves efficient coverage of data sources. We show that using these methods helps our agents generalize to changing environments.**

I. Introduction

Many data driven applications like weather forecasting rely on high quality data to make accurate prediction. Such accurate prediction is critical in domains like air traffic planning, agriculture, disaster management, generating economic models and much more. Area coverage with sensor networks at a reasonable resolution is therefore important to gather fine-grained information needed to build reliable models. However, sensor networks are very expensive to deploy and maintain. It's important to find optimal locations that will limit the cost while assuring good coverage.

Optimal placement of weather sensors needs to take into account factors such as climate zone, geography of the area, and population density where more accurate measurements are needed. Changing weather patterns and sensor failures also pose additional challenges to the placement and coverage problem. Most works try to treat the sensor placement problem as a static problem where the sensor agents are modeled at a fixed position. In case of sensor failure for instance, it can take hours or days to bring a weather station back up, leading to loss of critical data. In order to be reliable, weather data collected needs to be timely and accurate, as well as comprehensive in the different types of data to be collected (such as temperature, precipitation, wind speeds) as well as the geographic coverage. Having hybrid types of sensors (stationary and mobile) helps overcome these challenges by allowing mobile ones to move to a different location and reorganize to assure continuous coverage or to gather information in areas with limited accessibility. The sensors can thus be viewed as agents acting in the environment.

By modeling the weather sensors as mobile agents in a dynamic environment, we aim to identify the optimal location they should move to that will maximize coverage of high-interest zones. Modeling the problem as a Multiagent System (MAS) allows for cooperation and coordination between sensor agents such as reorganizing in case of changing weather patterns or a sensor failure (off service). But multiagent coverage is difficult in dynamic environments, and even more challenging when the data sources and the agents have heterogeneity. We aim to use a neuroevolutionary algorithm to learn a multiagent system using a coverage function that allows the system adapt to these conditions and determine optimal paths the agent should take when traversing to a new location so as to incur the least energy cost, and also be able to navigate around obstacles in the environment.

In this paper, we present a neuroevolutionary multiagent approach for optimal coverage of high-interest zones using mobile weather sensors. We improve on [1] by modeling a dynamic environment where weather patterns and points of interest (POIs) are constantly changing. We also model obstacles in the environment that the agents have to avoid and plan optimal paths around to traverse to the optimal location. Our main contribution is the design of an objective function based on the concept of information density. We show that our proposed method is able to learn near-optimal policy with limited training.

The rest of the paper is organized as follows: In section II we introduce key concepts in evolutionary algorithms. In section III we talk about previous works in the sensor placement and coverage optimization problem. We present our approach in section IV. The experiments and results are discussed in sections V and VI respectively. The conclusion and future work are discussed lastly in section VII.

II. Background

Neuroevolution is an unsupervised learning method that combines neural networks and genetic algorithms. The core concept is to enable learning of a given objective or target by exploring the action space. Neuroevolutionary algorithms

can fall within the paradigm of reinforcement learning where agents learn from their interactions with the environment.

In the neuroevolutionary algorithm, we initialize a population of neural networks and evaluate their fitness using a fitness function. Based on their fitness scores, an assigned number of networks are selected and mutated to replace the worst performing networks in the pool of neural networks. This leads to a new generation of neural networks. This process is continued until a target objective is met or until a set number of generations of evolution.

III. Related Work

Static coverage control consist of positioning sensors without any mobility, and the approach used to achieve this goal is an off-line optimization strategy aimed to minimizing the sum of weighted Euclidean distances to sensors [2]. Dynamic coverage control on the other hand, aims at deploying agents in a mission space subject to changing conditions, such that a defined objective function is optimized [3]. It mainly depends on the defined objective function, the density function and the dynamics of the agents [3]. When the environment is dynamically changing and large to the extent that it cannot be covered by a stationary team of agents, the coverage problem is called Persistent coverage control [4].

The general approach to solving a coverage control problems is to partition the mission space such that agents' subspace (local environment) don't overlap with other's, and then find the closed route for each agent in its sub-space [5].

Coverage control has received great interest in the recent years. For instance, [2] proposed a gradient-based distributed coverage control approach for coverage control, data source detection and data collection of mobile sensors. The approach allows coverage while handling polygonal obstacles in the mission space. In [3], a spatial estimation algorithm is proposed to perform coverage control in domains with unknown density of the information of interest, then a consensus mechanism is applied to improve the control strategy of the control system. [5] address the problem of persistent coverage in environments with complicated constraints. The authors address the environment partitioning limitations of methods like graph-based, spanning tree-based and clustering-based by performing sensors deployment before decomposing the mission space using a new clustering approach based on k-means.

Modern approaches to the coverage optimization problem make increasing use of Reinforcement Learning (RL) methods. RL algorithms can be classed into either value-based, policy-based or actor-critic. In value-based methods, the goal is to learn a value function while policy-based methods aim to optimize a policy directly. Actor-critic methods combine the two and try to learn both a value function and an optimal policy. The problems in this domain are usually formulated as Markov Decision Processes (MDPs).

[6] applied an actor-critic method to coordinate a team of mobile sensors to maximize reward (data) collection. Value-based methods are however more popular when studying the coverage problem. Various works have looked at optimizing the sensor coverage problem while minimizing energy consumption as a MAS using Q-learning and Distributed Value Functions [7, 8, 9, 10, 11]. [7] proposes a spatial objective function using a temporal difference algorithm to learn optimal policies for sensor placement in a spatial domain that is used to model distributed parameter systems. [12] studies game-theoretic approaches to coverage optimization by modeling mobile sensors in a multiplayer game. [1] aims to get optimal coverage by modeling the sensors as mobile agents with different types of data they can sense and different levels of sensitivity modeled as a gaussian. They use a neuroevolutionary algorithm to get the Q-values that are used to learn the value function. This allows the method to generalize to dynamic environments and sensor failures. Additionally, difference functions are used to improve the global objective function. Dynamic environments with obstacles have however not been studied in relation to the sensor coverage problem.

IV. System Modeling

A. Environment

To develop control policies that allow mobile sensor agents to coordinate and move to cover data sources optimally in an environment, we consider a continuous 2D plane as shown in Figure 1. We consider all agents (shown in red) and data sources (shown in blue) as point objects with a position (x_0, y_0), and obstacles as polygons in the environment.

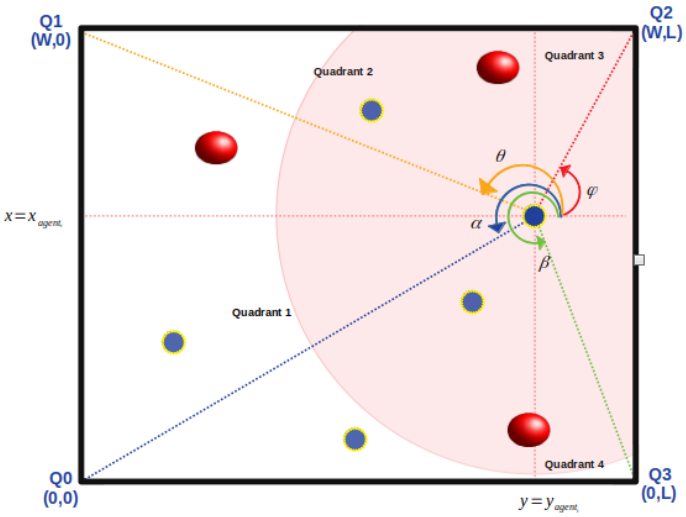

Fig. 1. Environment presentation with 3 sources and 5 agents. The sources are the blue spheres and the agents are the red circles.

B. Sources

Data sources are fixed Points Of Interest (POIs) with a Gaussian distribution representing prevalent weather patterns in the environment and the decay of their importance as you move away from them. The sources can be of different types such as temperature or precipitation, or both, and each type will have a strength representing the range to which it can be sensed. The strength of a data source can also be

affected by the area it is in. For example, sources in residential areas (represented by obstacles) have higher importance as POIs and thus would have higher strength. We represent the strength of a source as $v_{j,t}$ where j is the source index and t is the type of data. Each information type generated by a source is modeled as a two dimensional Gaussian with the shape controlled by $v_{j,t}$. The likelihood under each Gaussian, defined by $P_{j,t}(A_i|S_j)$ whose expression is given in Equation 1 represents how an agents located at position I senses source j for information type t.

$$P_{j,t}(A_i|S_j) = \frac{1}{\sqrt{(2\pi)^2|\Sigma|}} e^{-\frac{1}{2}(I-J)^T\Sigma^{-1}(I-J)} \tag{1}$$

Where $I = (x_i, y_i)$ and $J = (x_j, y_j)$ are the coordinates of agent i and source j respectively and Σ is the covariance matrix controlling the shape of the Gaussian.
$\Sigma = \begin{bmatrix} v_{j,t} & 0 \\ 0 & v_{j,t} \end{bmatrix}$

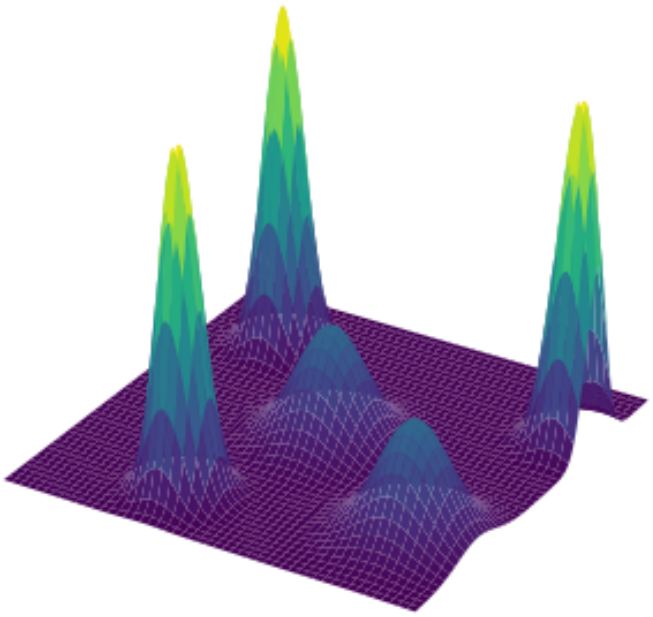

Fig. 2. Example of data generation distribution by 5 data sources (one information type).

C. Agents

An agent can have one or more types of sensors to measure different types of weather data, with different levels of efficiency. The efficiency determines how much of the source the agent can cover. Since an agent will typically never have 100% efficiency, it means multiple agents would be required to fully cover a source. The sensing efficiency of an agent to the sources is represented as $\mu_{i,t}$ where i is the agent index, and t is the type of sensor.

Agents sensing of a particular source is determined by the likelihood function in Equation 1. The closer an agent is to the center of a source the higher the likelihood for that particular source is. Since agents are equipped with different sensors, the overall coverage of source j by agent i is the mixture of all the Gaussian (one for each information type) of j.

D. States and Actions

An agent is able to observe both the data sources as well as other sensor agents. To get the state of an agent at a given time, we split the environment around the agent into four quadrants as shown in Figure 1.

The state of an agent, i, is comprised of sensing of sources, j, and other agents, a, in each quadrant, q. The sensing of sources in a given quadrant is denoted by $S_{i,t}(q)$, and the sensing of other agents in the given quadrant is denoted by $A_{i,t}(q)$, where t is the type of data or the type of sensor. Since we model the sources as Gaussian generating objects, we compute the sensing of the sources in a quadrant by agent i as the likelihood of i under each source, weighted by the information strength of agent i. And we express it as in Equation 2 below.
Given that the likelihood of a Gaussian is maximized at the position of the mean, all the agents under the current modeling approach will tend to converge to the center of the sources and will not achieve the expected coverage objective. To mitigate this, we model the sensing of the agents in each quadrant such that agent i is tricked to pick a quadrant in which the agents are far apart from it while maximizing the likelihood of the sources. We express this in Equation 3 with the assumption that the agents are equipped with GPS and know the position of all the other agents in the system.

$$S_{i,t}(q) = \sum_{j \in q} \mu_{i,t} P_{j,t}(A_i|S_j) \tag{2}$$

$$\begin{aligned} \delta_{i,a} &= \|i - a\| \\ A_{i,t}(q) &= \sum_{a \in q} \frac{\delta_{i,a}}{\max(\delta)} \end{aligned} \tag{3}$$

where $\delta_{i,a}$ is the euclidean distance between agent i and another agent a in quadrant q.

Given these states, an agent can take a discrete action to the center of the best surrounding four quadrants. To determine the best quadrant to step into, we calculate Q values for all the quadrants using a neuro-evolutionary approach. Each agent maintains its own set of neural networks. The input to the neural networks is the agent states. Since an agent's state consists of sensing of sources as well as other agents for each of the four quadrants, the input layer of the neural network will have $(2 \cdot 4 \cdot t)$ nodes, where t is the information or sensor type. The output layer will have 4 nodes, giving the Q value for each of the four quadrants. Once the best quadrant is selected, the angle ϕ is computed as illustrated on Fig. 1, and a constant move is then made by agent i in the direction of ϕ using the Equation 4 from [1]

$$x_{k+1} = x_k + step\_size \cdot \cos\left(\frac{\phi \cdot \pi}{180^\circ}\right)$$
$$y_{k+1} = y_k + step\_size \cdot \sin\left(\frac{\phi \cdot \pi}{180^\circ}\right) \quad (4)$$

Where (x_{k+1}, y_{k+1}) is the position of the agent after update, (x_k, y_k) is the old position (before update), ϕ is the direction in which to move and $step\_size$ is a constant parameter controlling the speed of the agent.

E. Fitness functions

We evaluate the performance of the system by calculating the total coverage of all the source by the sensor agents. We define the coverage of a source j by all the agents for a given information type t in equation 5 and we show that this formulation is equivalent to maximizing the likelihood of the agents under the mixture of Gaussians modeling the data generating sources.

$$C_{j,t} = \sum_{i=1}^{N_a} \mu_{i,t} P_{j,t}(A_i|S_j) \quad (5)$$

Where N_a is the number of agents in the system.

Proof. The ultimate goal is to maximize the likelihood under the mixture of Guassians distribution shown in Fig. 1. Maximizing the likelihood is equivalent to maximizing the negative log-likelihood. However, the negative log-likelihood is not bounded since $0 \leq P_{j,t}(A_i|S_j) \leq 1$. By adopting the exponential coppula to map the negative log-likelihood to $[0, 1]$, this will be equivalent to computing the exponential of log and therefore leads to identity. □

We use three reward functions to evaluate the neuro-evolutionary algorithm and the performance of our system. The overall system performance is given by the global objective function, and is a measure of the coverage of all the data sources and their information types. It is done by dividing the sum of the coverages of all the sources and types by the product of the number of sources and types, giving the average coverage of each source:

$$G(z) = \frac{\sum_j \sum_t C_{j,t}}{j \cdot t} \quad (6)$$

To calculate the local objective function, $L_i(z)$ of an agent, i, we estimate the coverage of the sources achieved with only agent i in the system using (5). $L_i(z)$ is then obtained using the same objective function in (6). For the difference objective function, $D_i(z)$, the same process is repeated but without the agent i.

$$L_i(z) = G(z_{+i}) \quad (7)$$

$$D_i(z) = G(z) - G(z_{-i}) \quad (8)$$

F. Neuro-evolutionary algorithm

The neural network that predicts the best direction to move by agent i, is trained using a neuro-evolutionary algorithm. Each agent holds a population of N neural networks randomly initialised, evaluated and progressively mutated until the n^{th} generation is reached [13]. The training process is given by Algorithm 1.

Algorithm 1: Neuro-Evolutionary Algorithm

```
while generation < G do
    for each agent i do
        Initialize a population of N neural networks;
        Select a network from the population to build a
          team with other agents' selection;
        Evaluate the team and assign the team's fitness
          to each team member;
        Rank the population according to their fitness;
        Select K networks from the population with
          epsilon-greedy;
        Mutate the K networks (parent networks) to
          create K new networks (offspring) ;
        Replace the K poorly performing networks
          from the population with the K offspring;
    end
end
```

V. Simulations

The system configuration for our simulations is the following:

A. Environment

The environment is a two dimensional 100×100 plane where the agents can move within the boundaries while avoiding the obstacles. The sources and the agents are placed uniformly at random within the environment. The sources are static while the agents act to cover the sources.

B. Sources and Agents

We place the sources in the environment at fixed location, by making them far apart from each other. The agents are placed at random in the bottom left quadrant, with the objective of achieving the optimal coverage as shown in Fig. 3 (bottom right). Each source generates two types of data (air temperature and precipitation) with different information strength (20 for temperature and 10 for precipitation). The agents' information efficiency is different for each of the sensors, and we set them to 5 for temperature and 3 for precipitation.

C. Learning

Each agent has a population of 50 neural networks with randomly generated initial weights, with a mutation probability of 0.1. Each network is evaluated after 50 steps (move) by the agent, using one of the fitness functions (global, local and difference reward) described above. In the experiments, the

number of information types is set to 2, which induces a neural network with 16 input units, each representing the ways agent i senses both the other agents and the sources in each of the 4 quadrants towards which it can move. The neural networks architecture is composed with one hidden layer of 10 neurons and 4 output nodes representing the value of each quadrant for any given input state. The hidden layer is ReLu activated and the softmax function is applied to the logit to compute the probability distribution over all the quadrants.

VI. Results

Fig.3 shows the trajectory of the agents with a policy learned after 1500 mutations. The black rectangles represent the obstacles, while the red circles are the sources and the greenish/blue circles are the agents' initial/final positions. The gray lines represent the movement of the agent from their initial position to the final stage after 50 steps.

From these results (Fig. 3), it can be seen that the coverage function learned under global and difference reward (bottom left and top left plots) are very similar and the agents are more widespread (better coverage) than with local reward (top right plot) where the agents seem to have learned to go to the closest source. Additionally, compared to the optimal coverage, the learned policies under global and difference reward still make the agents go to the center of the sources where the density is higher. This suggests that the agent-agent position constraint added with $A_{i,t}(q)$ defined above is not strong enough.

The system performance under the different types of reward defined above is given on Fig. 4. It can be seen that even though the difference is not statistically significant, the difference evaluation reward performs better than global reward signal which in turn is better than local reward. Even though we have tested the scalability of the proposed approach due to time constraint, we believe that our method is scalable and as the size of the system grows, our hypothesis is that difference evaluation will significantly outperform global and local reward by the nature of noise reduction property that it carries.

VII. Conclusion

In this paper, we used a neuroevolutionary algorithm to achieve near-optimal coverage of distributed data sources in an environment using mobile weather sensors by treating the problem as a multiagent system. We presented a density-based coverage function to evaluate the fitness functions in the neuroevolutionary algorithm using difference rewards. Our results show that this method can generalize to environments with changing POIs. Based on this, we speculate that it would also be able to generalize when simulated with sensor failures. We also show that by constraining agent-agent position, we limit trivial solution that leads the agents to converge to the sources with the highest density.

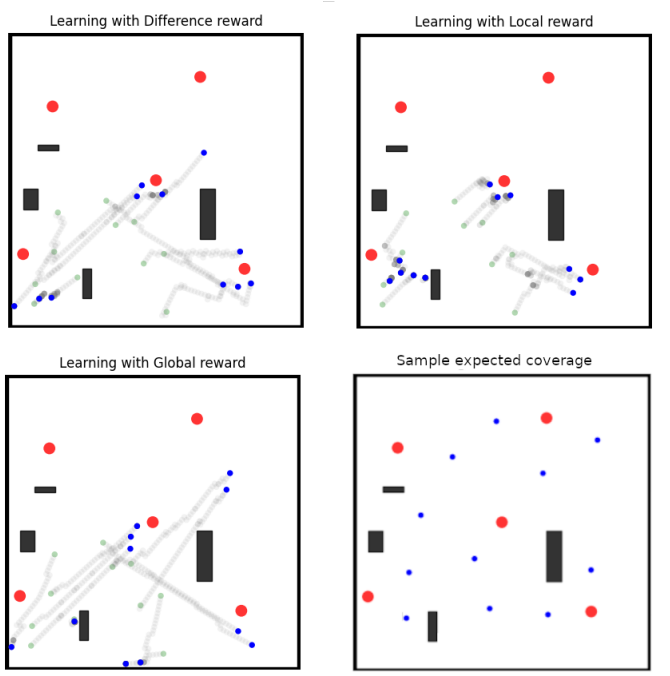

Fig. 3. Policy learned by 11 agents using global reward in an environment with 5 sources.

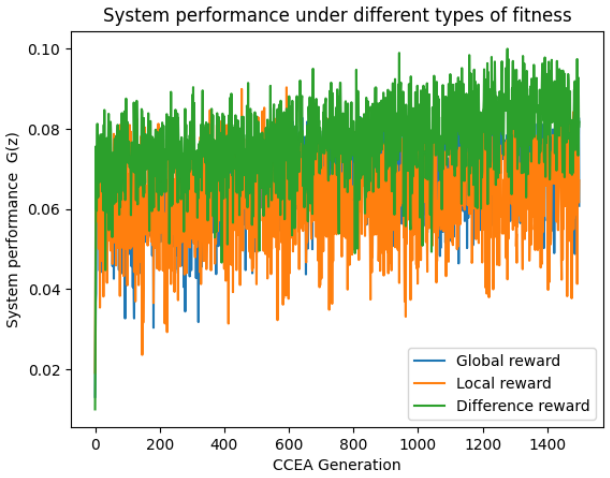

Fig. 4. System performance after 1500 episodes with 5 sources and 11 agents.

Our current implementation treats the source distribution model as constant. We aim to improve this model to make the data generation of the sources decrease as more agents cover it. This could improve an agent's ability to learn a more optimal location. We also seek to improve the constraints on agent positioning so that they do not all converge to a location and are able to find more spread out optimal locations in the environment as expected. Lastly, we aim to experiment with Deep Q-networks since they would be able to provide immediate feedback as compared to neuroevolutionary algorithms where we have to wait till the end of an episode

for an evaluation.

REFERENCES

[1] Christian Roth, Matt Knudson, and Kagan Tumer. "Agent fitness functions for evolving coordinated sensor networks". In: *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. 2011, pp. 275–282.

[2] Minyi Zhong and Christos G Cassandras. "Distributed coverage control and data collection with mobile sensor networks". In: *IEEE Transactions on Automatic Control* 56.10 (2011), pp. 2445–2455.

[3] Lei Zuo, Weisheng Yan, and Maode Yan. "Efficient coverage algorithm for mobile sensor network with unknown density function". In: *IET Control Theory & Applications* 11.6 (2017), pp. 791–798.

[4] Xuchao Lin and Christos G Cassandras. "An optimal control approach to the multi-agent persistent monitoring problem in two-dimensional spaces". In: *IEEE Transactions on Automatic Control* 60.6 (2014), pp. 1659–1664.

[5] Yuan Tang et al. "A Novel Cooperative Path Planning for Multirobot Persistent Coverage in Complex Environments". In: *IEEE Sensors Journal* 20.8 (2020), pp. 4485–4495.

[6] Paris Pennesi and Ioannis Ch Paschalidis. "A distributed actor-critic algorithm and applications to mobile sensor network coordination problems". In: *IEEE Transactions on Automatic Control* 55.2 (2010), pp. 492–497.

[7] Zhi Wang, Han-Xiong Li, and Chunlin Chen. "Reinforcement learning-based optimal sensor placement for spatiotemporal modeling". In: *IEEE transactions on cybernetics* 50.6 (2019), pp. 2861–2871.

[8] Mark Wei Ming Seah et al. "Achieving coverage through distributed reinforcement learning in wireless sensor networks". In: *2007 3rd International Conference on Intelligent Sensors, Sensor Networks and Information*. IEEE. 2007, pp. 425–430.

[9] Hongbin Chen, Xueyan Li, and Feng Zhao. "A reinforcement learning-based sleep scheduling algorithm for desired area coverage in solar-powered wireless sensor networks". In: *IEEE Sensors Journal* 16.8 (2016), pp. 2763–2774.

[10] Jean-christophe Renaud and Chen-khong Tham. "Coordinated sensing coverage in sensor networks using distributed reinforcement learning". In: *2006 14th IEEE International Conference on Networks*. Vol. 1. IEEE. 2006, pp. 1–6.

[11] Santosh Soni and Manish Shrivastava. "Novel learning algorithms for efficient mobile sink data collection using reinforcement learning in wireless sensor network". In: *Wireless Communications and Mobile Computing* 2018 (2018).

[12] Minghui Zhu and Sonia Martınez. "Distributed coverage games for energy-aware mobile sensor networks". In: *SIAM Journal on Control and Optimization* 51.1 (2013), pp. 1–27.

[13] Nicholas Zerbel. *RoverDomain*. URL: https://github.com/zerbeln/RoverDomain.

TABLE I
WORK DISTRIBUTION

Work	*Irené Tematelewo*	*Khushal Brahmbhatt*
Organization	50%	50%
Technical	70%	30%
Writing	50%	50%
Coding	100%	0%