Queen's University

MTHE 493 Winter 2015

Distributed Triggering Strategies for Deployment of Autonomous Mobile Networks with Outdated Information

Authors:

Ryan Farrell 10004993
Mikhail Hayhoe 10013003
Justin Ma 10006954
Mark Mahony 10012980

Supervisor:

Professor Bahman Gharesifard

April 6, 2015

Abstract

Consider a number of agents with limited communicating and data collecting capabilities that are deployed over a bounded region with an underlying information density distribution. The goal of this project is to design distributed triggering strategies for communication to balance the trade-off between collective information gathered and frequency of communication within the network. Specifically, these strategies were applied to aquatic robots tasked with gathering oceanographic data. To better model this environment, a number of assumptions were made, such as a maximum speed and a drift force to simulate ocean currents. The collective amount of information gathered was maximized by having the agents perform the discretized version of the continuous-time Lloyd's algorithm. Two event-triggering strategies were developed: constant triggering and non-constant triggering. The constant triggering strategy had each agent communicate their current position when they exceeded a constant distance r from their last communicated position. The non-constant triggering strategy expanded upon this by making the radius agent-specific as a function of the average distance between the agent and its neighbours. Both triggering strategies resulted in a significant decrease in the number of communications while sacrificing a relatively small amount of information gathered.

Acknowledgements

We would like to extend our sincerest gratitude to Professor Bahman Gharesifard for his guidance and support as a supervisor, teacher, and mentor throughout this project and during our time at Queen's.

Thank you to the Department of Mathematics and Engineering for providing us with the opportunity and knowledge necessary to pursue this project.

Thanks to our parents, families, and peers for their continued support.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Project Scope

While still a relatively futuristic technology in terms of commercial implementation, research in the areas of swarm intelligence and distributed network control have been thriving over the past decade with significant progress being made in both theoretical and applied fields. There is a plethora of theoretical topics available to study in this field, but it is important to consider the engineering trade-offs faced by these problems. When developing control algorithms for these systems, there are many real world applications that must be accounted and designed for.

A number of applications could be derived from the basis of swarm theory and network control such as a set of driverless cars which need to manoeuvre around a parking lot without colliding with one another or stationary objects. While any number of these complex examples could be imagined, in this project an application was selected which could be more easily modelled so that results and findings would be fit for real world implementation.

Research in this area [5] has detailed the functionality of a set of robotic agents whose purpose was to collect oceanographic data over long periods of time while trying to minimize movements and communications in order to preserve limited power supplies and extend the time the agents could be in operation without being recalled. While trying to preserve power, the agents were tasked with maximizing the amount of information they were gathering as a network while only having the ability to communicate with neighbouring agents. The trade-off between maximizing the amount of information gathered by the network and minimizing the amount of communication within the network was the chosen focus of this project.

The application in [5] was to model agents in an aquatic environment, so an attempt was made to replicate some of the environmental effects the agents were subject to in the model. This was accomplished through adding simulated ocean currents and wave-action to the region in which the agents were operating. Before proceeding to the development of strategies for agent behaviour to optimize the communication-performance trade-off, the communication capabilities of the agents needed to be established. Since the goal of the project was to explore distributed algorithms, which are strategies which rely solely on local information, the agents were only granted the ability to communicate with agents who were their immediate neighbours.

Several additional assumptions were made in an attempt to make the simulation platform more realistic with respect to the chosen application. Namely, the movement capabilities of the agents were limited and the initial deployment was assumed to be clustered as to simulate a redeployment of all agents.

After establishing a simulation environment, the goal of this project was to create localized strategies to optimize the aforementioned trade-off between communications and network coverage. This was the design component of this project; at the

time of implementation, optimizing this trade-off was very much an open problem with no established solutions.

After developing a number of strategies and comparing their performance, the objective was to determine the engineering implications of these findings. This included how these results might yield cost savings in the oceanographic data collection application, as well as other benefits such as reduced environmental impact or improved data collection efficiency of the network.

1.2 Problem Statement

Given a set of agents with limited communicating and sensing capabilities, the objective is to design a distributed control algorithm to maximize the information gathered while minimizing the energy consumed by lowering the number of communications between agents.

The approach considered was to design a number of distributed triggering strategies to dictate when agents would communicate, as their movement behaviour would be based on the last instance when they communicated with surrounding agents.

2 Mathematical Background

There are numerous mathematical definitions and concepts that were used throughout the project. Unless otherwise stated, the content in this chapter is based on the text *Distributed Control of Robotic Networks* written by F. Bullo, J. Cortés, and S. Martínez [1].

Suppose we have a bounded region $S \subset \mathbb{R}^d$ which is the area of interest. Assume there is a density function $\phi : S \to \mathbb{R}_{\geq 0}$ which represents the distribution of information over S. In other words, at any point $x \in S$, the value of $\phi(x)$ indicates the importance of that point. Finally, define a sensing function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}$ as a non-increasing function which represents the quality of the information that an agent receives from a location $x \in S$.

2.1 The Expected-Value Multicenter Function

The goal is to capture the maximum amount of information in S, given a set of physical constraints, by optimally placing the sensing agents. In order to quantify how much information is being captured, we introduce the expected-value multicenter function $\mathcal{H}_{exp}$.

Definition 2.1. (Expected-Value Multicenter Function): *Suppose there are n sensing agents each with position p_i. Given a density function ϕ and sensing function f, define the expected-value multicenter function (EVMF) $\mathcal{H}_{exp} : S^n \to \mathbb{R}$ as*

$$\mathcal{H}_{exp}(p_1, \ldots, p_n) = \int_S \max_{i \in \{1,\ldots,n\}} f(\|q - p_i\|_2)\phi(q)dq \tag{1}$$

Therefore the goal of maximizing information is equivalent to maximizing the EVMF or equivalently $\mathcal{H}_{exp}$. We now discuss the components that are used to construct $\mathcal{H}_{exp}$.

2.1.1 Voronoi Partitions

Definition 2.2. (Partition): *A partition of a set S is a collection of closed connected sets $W_1, \ldots, W_n \subset S$ which satisfy they following properties:*

1. $S = \bigcup_{i=1}^{n} W_i$
2. $(\mathring{W}_j) \cap (\mathring{W}_k) = \emptyset, for j \neq k$

Where $\mathring{W}_i$ denotes the interior of W_i.

Given a set of agent positions $(p_1, \ldots, p_n)$ distributed over a space S, each assigned to a partition W_i, we can rewrite the EVMF function as

$$\mathcal{H}_{exp}(p_1, \ldots, p_n, W_1, \ldots, W_n) = \sum_{i=1}^{n} \int_{W_i} \max_{i \in \{1,\ldots,n\}} f(\|q - p_i\|_2)\phi(q)dq \tag{2}$$

Definition 2.3. (Voronoi Partitions): *Given a set $S \in \mathbb{R}^d$ and n distinct points $\mathcal{P} = \{p_1, \ldots, p_n\} \subset S$, the Voronoi partition of S with $\mathcal{P}$ is the collection of sets $\mathcal{V}(\mathcal{P}) = \{\mathcal{V}_1(\mathcal{P}), \ldots, \mathcal{V}_n(\mathcal{P})\}$ where*

$$\mathcal{V}_i(\mathcal{P}) = \{q \in S \mid \|q - p_i\|_2 \leq \|q - p_j\|_2, \forall p_j \in \mathcal{P}\} \tag{3}$$

Theorem 2.1. (Optimality of Voronoi Partitions): *Since the sensing function $f(\|q - p_i\|_2)$ is non-increasing with $\|q - p_i\|$, the optimal set of partitions which maximize the value of the $\mathcal{H}_{exp}$ will be the Voronoi partitions.*

Proof: See Proposition 2.13 in [1]. Intuitively, for a given point $x \in S$, the agent which is closest to it should be the agent that collects data from x. This arrangement exactly defines the Voronoi partition where each agent is assigned all space which is closer to that agent than any other.

2.1.2 Sensing Function

Sensing functions $f : \mathbb{R}_{\geq 0} \to \mathbb{R}$ are non-increasing piecewise continuously differentiable functions with a finite number of jump discontinuities. These functions determine the quality of the information received by the agent based on the distance to the point it is collecting data from. A few special cases are described below.

Distortion Problem
The sensing function $f(x) = -x^2$ results in a case called the distortion problem. The justification for this sensing function is the inverse square law which states that a specified physical quantity is inversely proportional to the square of the distance from the source of that physical quantity. The distortion problem is relevant in disciplines involving vector quantization, signal compression, and numerical integration.

Area Problem
The sensing function $f(x) = 1$ results in a case called the area problem since, when $\phi(x)$ is uniform, $\mathcal{H}_{exp}$ becomes the total area covered by the agents.

In this project, we consider a mix of the distortion problem and the area problem. The distortion problem is considered because agents are acquiring information from a distance away from them. However, an assumption made is that agents can perfectly acquire data within a distance α, and beyond that their sensing abilities quickly decay to zero. This is more in line with the area problem. We call α the sensing radius. The sensing function used for the agents is shown below.

$$f(x) = \begin{cases} 1 & \text{if } x \leq \alpha \\ \mathrm{e}^{\frac{-\psi}{\alpha}(x-\alpha)^2} & \text{if } x > \alpha \end{cases} \quad (4)$$

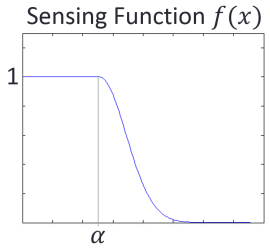


Figure 1: The sensing function $f(x)$

$f(x)$ was defined to be a function that is unity until α, at which point it decays smoothly to zero. The values of α and ψ would depend on the capabilities of the sensor the agents are using, but in this case they were fixed at $\alpha = 0.05$ and $\psi = 80$. These values result in approximately a 68% disjoint network-wide coverage of S when using 50 agents.

2.1.3 Density Function

The density function $\phi(x)$ describes the distribution of information over S. Two density functions were used in the simulations, both being time invariant and normalized so that

$$\int_S \phi(q)dq = 1$$

The first distribution is symmetric and was named "Volcano":

$$\phi(x,y) = 1 + \sin(8\sqrt{(x-0.5)^2 + (y-0.5)^2}) \tag{5}$$

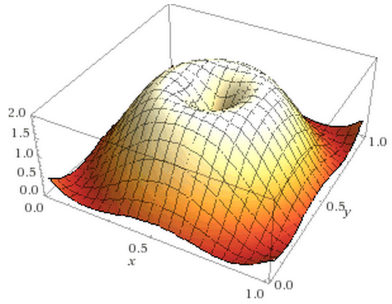


Figure 2: The Volcano distribution

The second distribution is asymmetric and was named "Islands":

$$\phi(x,y) = \frac{1}{0.13883}(0.7xe^{-49(x-0.8)^2} - 25(y-0.6)^2 + 1.5ye^{-16(x-0.3)^2} \\ -36(y-0.2)^2 + 1.8xye^{-16(x-0.3)^2} - 36(y-0.75)^2) \tag{6}$$

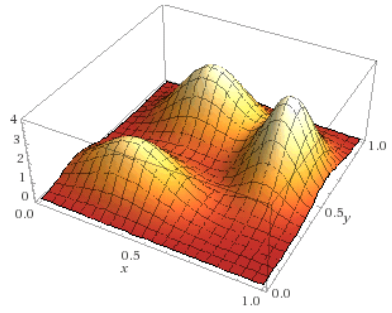


Figure 3: The Islands distribution

2.2 Maximizing The Expected-Value Multicenter Function

One of the main objectives is to maximize $\mathcal{H}_{exp}$ while considering the trade-offs that arise. It is shown below that with the current parameters and assumptions, a local maximum can be achieved. There are two considerations for this local maximum: the positioning of the agents to achieve a local maximum, and the method by which they reach those positions.

Definition 2.4. (Local and Global Maximums): *Suppose S is a metric space. A real-valued function $g : S \to \mathbb{R}$ has a:*

1. ***Local maximum*** *at $x_{lmax} \in S$ if $g(x_{lmax}) \geq g(x), \forall x \in [x_{lmax} - \epsilon, x_{lmax} + \epsilon]$ for some $\epsilon \in \mathbb{R}$*
2. ***Global maximum*** *at $x_{gmax} \in S$ if $g(x_{gmax}) \geq g(x), \forall x \in S$*

Note that it is true that if x_{max} is a global maximum then it is a local maximum, but the converse is not necessarily true. For this report, it is sufficient to find a local maximum of $\mathcal{H}_{exp}$. Since the density functions used will be smooth with a relatively small number of local maximums, often a solution resulting in a local maximum will achieve a global maximum.

2.2.1 Center of Mass

The idea of the center of mass is important because it will provide an algorithm to reach a local maximum. For the following definitions, let $S \subset \mathbb{R}^d$ be the bounded region of interest and let $\phi : \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ be the density function associated with the region S.

Definition 2.5. (Mass): *The mass of a bounded set $B \subset S$ with a density function ϕ is denoted $M_\phi(B)$ and can be computed by*

$$M_\phi(B) = \int_S \phi(q)dq \tag{7}$$

Definition 2.6. (Center of Mass[1]): *The center of mass of a bounded set $B \subset S$ with a density function ϕ is denoted $CM_\phi(B)$ and can be computed by*

$$CM_\phi(B) = \frac{1}{M_\phi(B)} \int_S q \cdot \phi(q)dq \tag{8}$$

[1]The centroid, which is often confused with the center of mass, is the point corresponding to the center of mass when the distribution is uniform.

2.2.2 Differentiability of $\mathcal{H}_{exp}$

In order to determine the desired position for an agent to maximize $\mathcal{H}_{exp}$, it is required that $\mathcal{H}_{exp}$ be differentiable.

Definition 2.7. (Globally Lipschitz): *A function $f : S \to \mathbb{R}^d$ is Globally Lipschitz if there exists $K \in \mathbb{R}_{>0}$ such that*

$$\|f(x-y)\|_2 \leq K\|x-y\|_2 \qquad \forall x, y \in S \tag{9}$$

Theorem 2.2. (Differentiability of $\mathcal{H}_{exp}$) : *Given a set $S \subset \mathbb{R}^d$ that is bounded and measurable, a density $\phi : \mathbb{R} \to \mathbb{R}_{\geq 0}$, and a sensing function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}$, the expected-value multicenter function $\mathcal{H}_{exp} : S^n \to \mathbb{R}$ is*

1. *Globally Lipschitz on S*
2. *Continuously differentiable on S where for $i \in \{1, \ldots, n\}$*

$$\begin{aligned}\frac{\partial \mathcal{H}_{exp}}{\partial p_i}(\mathcal{P}) &= \int_{V_i(\mathcal{P})} \frac{\partial}{\partial p_i} f(\|q-p_i\|_2)\phi(q)dq \\ &+ \sum_{a \in Dscn(f)} (f_-(a) + f_+(a)) \int_{V_i(\mathcal{P}) \cap \partial\overline{B}(p_i,a)} n_{\text{out}}(q)\phi(q)dq\end{aligned} \tag{10}$$

where $Dscn(f)$ is the set of all discontinuities of f and n_{out} is the outward normal vector to the closed ball $\overline{B}(p_i, a)$.

Proof: See Proposition 2.1.6 in [1].

Now we can show how with n agents at positions $p_1, \ldots, p_n$, sensing function $f(x) = -x^2$, and a Voronoi partition on the space, a locally optimal solution will be obtained if each agent moves in the direction of the center of mass of its Voronoi partition.

$$\frac{\partial \mathcal{H}_{exp}}{\partial p_i}(\mathcal{P}) = \int_{V_i(\mathcal{P})} \frac{\partial}{\partial p_i} f(\|q-p_i\|_2)\phi(q)dq \tag{11}$$

$$= \int_{V_i(\mathcal{P})} \frac{\partial}{\partial p_i} (-\|q-p_i\|_2^2)\phi(q)dq \tag{12}$$

$$= 2M_\phi(V_i(\mathcal{P}))(CM_\phi(V_i(\mathcal{P})) - p_i) \tag{13}$$

The intermediate steps between lines (12) and (13) are a result of the Parallel Axis Theorem. What this is saying is that a local maximum is reached when $\frac{\partial \mathcal{H}_{exp}}{\partial p_i}(\mathcal{P}) = 0$ which is only the case when $CM_\phi(V_i(\mathcal{P})) = p_i$. In other words, to obtain a local maximum each agent should move so that its position p_i is equivalent to the center of mass of its Voronoi partition.

2.2.3 Optimization Methods

Optimization problems can be phrased as maximizing or minimizing an objective function within a set of constraints. There are many techniques and algorithms to find a local maximum of a function.

Definition 2.8. (Gradient): *The gradient of a scalar function $f(x_1, x_2, \ldots, x_n)$ is defined as*

$$\nabla f = \frac{\partial f}{\partial x_1} e_1 + \ldots + \frac{\partial f}{\partial x_n} e_n \tag{14}$$

where $(e_1, \ldots, e_n)$ is the canonical basis of $\mathbb{R}^n$.

As stated previously, we have determined that the optimal location for an agent to maximize $\mathcal{H}_{exp}$ is at the center of mass of its Voronoi partition. We now look at methods for getting the agent to that position. One of the simplest methods is the method of steepest ascent.

Definition 2.9. (The Method of Steepest Ascent): *Consider a starting point $x_0 \in S$ with an underlying density function ϕ. The method of steepest ascent is an iterative procedure where at each iteration the choice of direction is where ϕ increases most quickly. In other words,*

$$x_{k+1} = x_k + \lambda_k \nabla f(x_k) \tag{15}$$

where λ_k dictates the step size in that direction.

For this project, λ_k can be interpreted as the velocity of each agent which will be bounded by physical constraints. The method of steepest ascent is a first order optimization method. The rate of convergence to a local maximum is relatively slow because the agent will tend to "zig-zag" to the optimum solution, especially when the function is non-concave.

The procedure of computing the centroid for each agent's Voronoi and moving linearly towards that point at each iteration is called Lloyd's Algorithm. For this project, because the density function is not uniform, Lloyd's Algorithm is modified to use the center of mass of each Voronoi.

Definition 2.10. (Lloyd's Algorithm): *Suppose there is an initial placement of n agents in a domain S. Lloyd's Algorithm dictates the n agents perform the following procedure:*

1. *The Voronoi of the n agents is computed*
2. *Integrate each Voronoi partition to find the centroid*
3. *Each agent moves to the centroid of its Voronoi partition*
4. *If convergence has not been achieved, repeat this process*

In summary, $\mathcal{H}_{exp}$ will be locally maximized when all agents are at their center of mass of their Voronoi partition. In order to reach this local maximum, agents should move in a straight line towards their center of mass.

2.3 Stability

Stability theory analyzes the stability of solutions and trajectories of dynamical systems under small perturbations of initial conditions.

2.3.1 Lyapunov Direct Method

Consider a dynamical system given by

$$\dot{x} = f(x,t) \qquad x(t_0) = x_0 \qquad x \in \mathbb{R}^n \tag{16}$$

Lyapunov's direct method[2] [6] allows us to determine the stability of the differential equation (16) without explicit integration. The idea is that if there is a measure of the "energy" in the system, the stability can be determined by analyzing the rate of change of the energy in that system. The equation that measures the energy will be denoted V. For the definitions that follow, let $B_\epsilon = \{x \in \mathbb{R}^n : \|x\| < \epsilon\}$.

Definition 2.11. (Equilibrium Point): *A point $x^* \in \mathbb{R}^n$ is an equilibrium point of the system* (16) *if $f(x^*, t) = 0$.*

Definition 2.12. (Locally Positive Definite Functions): *A continuous function $V : \mathbb{R}^n \times \mathbb{R}_{\geq 0} \to \mathbb{R}$ is a locally positive definite function if for some $\epsilon > 0$ and some continuous, strictly increasing function $h : \mathbb{R}_{\geq 0} \to \mathbb{R}$,*

$$V(0,t) = 0 \quad \text{and} \quad V(x,t) \geq h(\|x\|) \qquad \forall x \in B_\epsilon, \forall t \geq 0 \tag{17}$$

[2]The indirect method uses the linearization of the system to determine the local stability of the original system

Locally positive definite functions behave similarly to energy functions in a local sense. Functions that are globally similar to energy functions are called positive definite functions.

Definition 2.13. (Positive Definite Functions): *A continuous function* $V : \mathbb{R}^n \times \mathbb{R}_{\geq 0} \to \mathbb{R}$ *is a positive definite function if it satisfies the definitions of being a locally positive definite function and if* $h(p) \to \infty$ *as* $p \to \infty$

In order to bound the energy functions, we define decrescence.

Definition 2.14. (Decrescent Functions): *A continuous function* $V : \mathbb{R}^n \times \mathbb{R}_{\geq 0} \to \mathbb{R}$ *is decrescent if for some* $\epsilon > 0$ *and some continuous, strictly increasing function* $g : \mathbb{R}_{\geq 0} \to \mathbb{R}$

$$V(x,t) = 0 \leq g(\|x\|) \qquad \forall x \in B_\epsilon, \forall t \geq 0 \tag{18}$$

The time derivative of V, denoted $\dot{V}$, is taken along the trajectories of the system given by

$$\dot{V} = \dot{V}|_{\dot{x}=f(x,t)} = \frac{\partial V}{\partial t} + \frac{\partial V}{\partial x} \cdot f \tag{19}$$

Using the previous definitions and the definition of the time derivative of V, the theorem below allows us to determine the stability of the system by studying an appropriate energy function [6].

Theorem 2.3. (Basic Theorem of Lyapunov): *Let* $V(x,t)$ *be a non-negative function with derivative* $\dot{V}$ *along the trajectories of the system.*

1. *If* $V(x,t)$ *is locally positive definite and* $\dot{V}(x,t) \leq 0$ *locally in* x *and for all* t*, then the origin of the system is locally stable (in the sense of Lyapunov)*
2. *If* $V(x,t)$ *is locally positive definite and decrescent, and* $\dot{V}(x,t) \leq 0$ *locally in* x *and for all* t*, then the origin of the system is uniformly locally stable (in the sense of Lyapunov)*
3. *If* $V(x,t)$ *is locally positive definite and decrescent, and* $-\dot{V}(x,t)$ *is locally positive definite, then the origin of the system is uniformly locally asymptotically stable.*
4. *If* $V(x,t)$ *is locally positive definite and crescent, and* $-\dot{V}(x,t)$ *is positive definite, then the origin of the system is globally uniformly asymptotically stable.*

As a rough summary, Theorem 2.3 is stating that when $V(x,t)$ is a locally positive definite function and $\dot{V}(x,t) \leq 0$ then we can conclude the stability of the equilibrium point.

2.4 Triggered Control

In many control applications involving sensors and wireless communications, control tasks are executed periodically. However, choosing a constant period to execute tasks often leads to over-utilization of resources. In the context of this project, if an agent is close to its desired optimal position then it may no longer need to continuously communicate with other agents because they would not have moved significantly since the last communication. Two common control schemes to mitigate the wasting of resources are event-triggering and self-triggering. Triggering refers to a situation where the system actuates its control; in this context, the it would communicate. Both methods use feedback of the system as opposed to the classical time-triggering which is done in an open loop because it does not consider the status of the system. As a result, the focus shifts from classical periodic control to aperiodic control.

2.4.1 Event-Triggering

Event-triggering, which is the triggering strategy used in this project, is reactive and generates control actuation when the state deviates more than a certain threshold which is designed a priori [4]. Shown below is the main mathematical idea behind event-triggering.

To simplify the ideas behind event-triggering, a linear model is used but the result holds for non-linear systems as well [8]. Consider the linear system given by

$$\frac{d}{dt}x_p = A_p x_p + B_p u \qquad x_p \in \mathbb{R}^n, u \in \mathbb{R}^n \tag{20}$$

and assume there is a linear feedback control law

$$u = Kx_p \tag{21}$$

that makes the system (20) asymptotically stable. With this control law, the system becomes

$$\frac{d}{dt}x_p = A_p x_p + B_p K x_p \tag{22}$$

The problem of triggering strategies is how to implement the control law (21) such that it avoids the inefficiencies described earlier by using the method of event-triggering. The idea is to recompute (21) not at a periodic time frequency but rather when the performance is unsatisfactory. An example of quantifying performance can be seen by using a Lyapunov function $V(x_p) = x_p^T P x_p$ where P is a symmetric positive-definite matrix and $V(x_p)$ satisfies

$$\frac{d}{dt}V(x_p(t)) = \frac{\partial V}{\partial x_p}(A_p + B_p K)x_p = -x_p^T Q x_p \tag{23}$$

and where Q will be positive-definite. Since equation (23) is negative, the time derivative of V is negative and hence along the solution of the closed-loop system, V decreases. The rate at which V decreases is determined by Q. We are thus guaranteed the inequality

$$\frac{d}{dt}V(x_p(t)) \leq -\sigma x_p^T Q x_p \qquad \sigma \in (0,1) \tag{24}$$

Here σ is a design parameter because a smaller value means the user is willing to tolerate a slower rate of decrease before triggering, and so its exact value is dependant on the application. The idea of event-triggering is that whenever (24) is violated, the control is triggered. This event-triggering strategy has made the periodicity of the controller independent of time. With the proper choice of σ, this will result in a more efficient use of some system resources.

2.4.2 Self-Triggering

Self-triggering is proactive in the sense that the next control actuation is forecasted using previously known data. Up until the forecasted time, the system does not need to check any conditions to determine whether it should communicate.

Self-triggering is much more difficult to implement and design, which is why event-triggering was the triggering method used for this project.

3 Simulation Design

3.1 Algorithm Progression

The model was simulated in MATLAB; throughout the course of the project, many different versions of the simulation were created. Their main functionality and implementation are described below. Their full descriptions can be found in Section 3.2.

3.1.1 Voronoi Partitions

The simulation began by displaying a set of points along with their Voronoi partitions. To test the implementation of the partitioning algorithm, agents were randomly deployed across the space S. The first problem encountered was that MATLAB's built-in Voronoi function did not limit the partitions; that is, they would extend out of the space S off to infinity. To solve this problem, the VoronoiLimit function (Section 3.2.4) was used and modified so that it would calculate the Voronoi partitions of the agents limited to the space $S = [0,1] \times [0,1]$.

3.1.2 Agent Movement

After the placement of the agents and the calculation of their Voronoi partitions were in place, the next step was to implement movement. To make the problem more realistic, as well as to challenge the robustness of all algorithms, the agents were randomly deployed in a cluster within the region $[0, 0.25] \times [0, 0.25]$.

The most important consideration when deciding on an algorithm for movement was that it be distributed; in other words, agents could decide where they would move based only on information that was available locally. To this end, the continuous-time Lloyd's algorithm with discretized time-steps was implemented. Using this algorithm allowed for the discretization of agent movement into uniform time-steps while also guaranteeing locally maximum coverage of S. In this application, it was deemed sufficient for agents to reach a locally optimal coverage because finding a global optimum was an incredibly complex and open problem at the time of implementation. At this stage the information density was set to be uniform, and thus Lloyd's algorithm simply required agents to follow the gradient flow; in this case, they followed the path towards the centroid of their Voronoi partition. After the implementation, it was observed that the agents moved to a locally optimal coverage of the space regardless of their initial positions.

3.1.3 Information Density

After agent movement was in place, the information density was changed to be non-uniform, as it would be in a real application. This proved to be challenging for a number of reasons. The obvious change in the algorithm was that instead of an agent moving toward its centroid, it now had to move towards its center of mass. In this case, the mass is considered to be the amount of information in the cell. Finding the center of mass proved to be incredibly difficult because a precise calculation would involve the integration of the information density over an agent's Voronoi cell. Instead, the integral was approximated using a weighted sum calculation. The main idea was to triangulate each cell using its vertices and centroid, weight each triangle's centroid using its mass, and then average these values to find the center of mass of the cell. To start, each triangle was described as a set of vertices

$$T_i = \{(x_{1_i}, y_{1_i}), (x_{2_i}, y_{2_i}), (x_c, y_c)\} \tag{25}$$

where (x_c, y_c) is the centroid and $(x_{1_i}, y_{1_i}), (x_{2_i}, y_{2_i})$ are adjacent vertices of the cell. Next, a weighting factor was calculated for each triangle as

$$W(T_i) = Area(T_i) \cdot \frac{\phi(x_{1_i}, y_{1_i}) + \phi(x_{2_i}, y_{2_i}) + \phi(x_c, y_c)}{3} \tag{26}$$

Then the centroid of each individual triangle is found and denoted by

$C(T_i) = (x_{c_i}, y_{c_i})$. The center of mass of the Voronoi cell was then calculated as

$$V_{CM} \approx \frac{\sum_{i=1}^{n} C(T_i)W(T_i)}{\sum_{i=1}^{n} W(T_i)} \tag{27}$$

With this calculation in place, each agent then followed the gradient flow towards their center of mass. Once again Lloyd's algorithm guaranteed that they would reach a locally optimal coverage of the information density. An important remark is that the center of mass is computed using only local information. As long as each agent knows the borders of its Voronoi partition, as well as the information density inside their Voronoi cell, the center of mass can be calculated. The agents can then find the displacement vector, denoted $\vec{d}$, between their current position and center of mass and move as far along it as their max speed would allow.

3.1.4 Drift

In an effort to root the problem in a real world situation, the simulation was modified to emulate a scenario where the agents were aquatic robots tasked with gathering information in the ocean. To this end, a current was introduced that randomly influenced the movement of the agents. The current was assumed to be uniform across all agents, but changed with time in both magnitude and direction.

After the initial implementation of the current, henceforth referred to as drift, the agents continued to move along $\vec{d}$ and let the drift push them as they went. It was soon realized that this was not the most efficient way for the agents to move, because they were acting as though the drift did not exist. This led us to the next iteration of the algorithm which attempted to fight the drift.

Moving to counteract drift

Imposing the drift proved to be much more difficult than initially imagined. To fight it, agents required knowledge of the drift at the previous time-step. To this end, it was assumed that each agent had a sensor that could measure the drift it had previously been subjected to, denoted $drift_{past}$. Using this knowledge, along with the fact that the drift did not change drastically between time-steps, the agents could estimate the effect it would have on their movement.

Defining $move_{agent}$ as the direction that the agent would move in the absence of drift and $move \triangleq move_{agent} + drift$ as the net movement of the agent, simple vector addition is used to determine how our agent should move:

$$move_{agent} = v_{max}\frac{\vec{d}}{\|\vec{d}\|} - drift_{past} \tag{28}$$

The resulting net movement of the agent is:

$$move = v_{max}\frac{\vec{d}}{\|\vec{d}\|} - drift_{past} + drift \tag{29}$$

The past drift vector was subtracted because the drift experienced minimal change between time-steps and so $\|drift - drift_{past}\|$ was small relative to v_{max}. This method worked well, but it allowed agents to violate the maximum speed condition when the $\vec{d}$ and $drift_{past}$ acted in similar directions.

Advanced Drift Cancellation

To ensure the agents obeyed the maximum speed limitation when moving, the problem of movement became a complicated geometric one. First they had to determine whether it was possible to reach their ideal position given their knowledge of the past drift. If it was, they attempted to move directly to their center of mass. If not, a complicated calculation was performed to make their net movement vector as straight as possible given their knowledge of the past drift. First, the angle of the agent's movement θ was calculated as

$$\theta = \arcsin\left(\frac{1}{v_{max}}\sqrt{\frac{\|drift\|^2}{1+\frac{1}{\gamma^2}}}\right) \tag{30}$$

where

$$\gamma \triangleq \tan\left(\arccos\left(\frac{\langle\vec{d}, drift_{past}\rangle}{\|drift_{past}\|\|\vec{d}\|}\right)\right) \tag{31}$$

and $\langle\cdot,\cdot\rangle$ is the Euclidean inner product. Trigonometry was then used to determine $move$ after applying $move_{agent}$. It was assumed $\|drift_{past}\|$ was never identically zero. Again, the best estimate was to assume the current drift was identical to the past drift so the components of $move$ were calculated as

$$move_x = v_{max} \cdot \cos(\arctan(\vec{d}_y, \vec{d}_x + \theta))$$
$$move_y = v_{max} \cdot \sin(\arctan(\vec{d}_y, \vec{d}_x + \theta))$$

where the arctan function used here is four-quadrant inverse tangent which allows the calculation of the arctangent in any quadrant. With this, the movement vector becomes

$$move = (move_x, move_y) - drift_{past} + drift \tag{32}$$

After implementing this movement, strange jumps in the movement of the agents appeared when the $move_{agent}$ vector pointed to the left. It turned out that the erratic behaviour was being caused by the inverse tangent function atan in MATLAB.

Since this operation is usually defined on $[\frac{-\pi}{2}, \frac{\pi}{2}]$, the agents were not properly calculating the direction of $move_{agent}$ when it pointed to the left, which is the region $[\frac{\pi}{2}, \frac{3\pi}{2}]$. To solve this problem, the function atan2 was used. This function extends the region on which the inverse tangent is defined to be from $[-\pi, \pi]$. After this fix the agents efficiently moved to counteract the drift by trying to force $move$ to be directly along $\vec{d}$ using their knowledge of $drift_{past}$.

3.1.5 Sensing Function

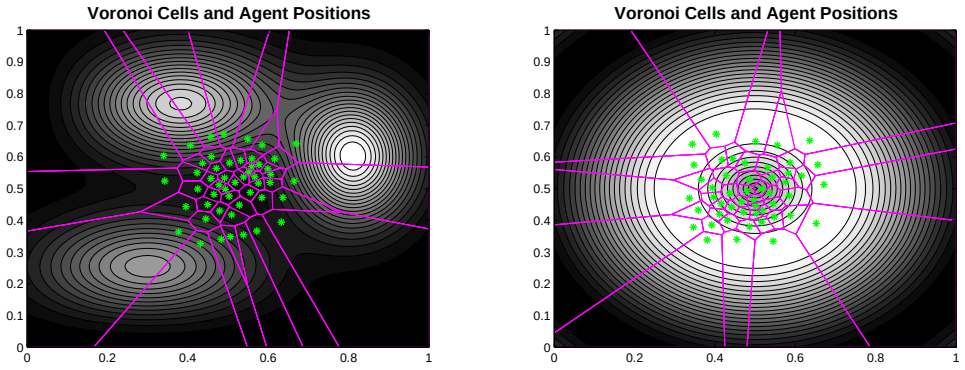
The next step in the algorithm was to implement the sensing function $f(x)$. This involved limiting the radius within which the agents could collect data as described in Section 2.1.2, and thus the region over which the center of mass was calculated was reduced. The range was limited by introducing $f(x)$ into the calculations for the weighting of the triangles used to calculate mass. However, if the vertices of the cell were past the radius α of $f(x)$ they contributed little to the weighting factor, and thus the center of mass would be biased to ignore that region.

To fix the mass bias problem, all of the cell's initial triangulations were sub-triangulated recursively. This process could be done as many times as necessary to achieve a desired precision in mass; however, it introduced significant computational complexity and increased the run-time of the simulation. It was determined after several trials that two sub-triangulations (for a total of 3 triangulation operations) was sufficient to alleviate the issue created by $f(x)$ while keeping the computational time at a minimum. This sub-triangulation also made the mass calculations more accurate, which meant the center of mass approximations were closer to their actual value.

3.1.6 Triggering Strategies

The last step in the algorithm was to implement the triggering strategies. This involved several small modifications to the algorithm. First the number of communications, distance travelled, and value of $\mathcal{H}_{exp}$ were recorded in every time-step. This allowed the performance of the strategies to be tracked in order to ensure they met our stated objectives of lowering the number of communications while sacrificing minimal amounts of information gathered. The exact implementation of these algorithms, as well as their full descriptions, can be found in Section 4.

The graphical visualization of the model, as output by MATLAB, is shown in Figure 4 below during the initial deployment stage for the two distributions used.



(a) Islands distribution

(b) Volcano distribution

Figure 4: Example of the random initial deployment of the agents

3.2 Functions

The functions moveVoronoi, voronoiPlot, and subTriangulate were coded specifically for this project. The functions VoronoiLimit[3] and polygeom[4] are open-source functions for MATLAB.

3.2.1 moveVoronoi()

moveVoronoi is the main function for this simulation which is executed to begin the simulation. Its main purpose is to determine how the agents should move in each time-step. It runs over 100 time-steps, as this provides more than enough time for all of the algorithms to reach their final resting positions while keeping the run-time of the simulation relatively low. The number of time-steps can be adjusted as desired to see the operation of the simulation over different lengths of time.

To begin with, several constants based on the real world assumptions must be defined. These values can be modified depending on the system that is being modelled, but all methodologies and procedures that have been conducted in this analysis are completely independent of them. Therefore, the same processes can be repeated to find different results based on a change in these constants.

Set Model Parameters

Firstly, $n = 50$ agents are randomly deployed over the space $S = [0, 1] \times [0, 1]$. Their deployment is restricted to the space $[0.375, 0.625] \times [0.375, 0.625]$, which represents the middle quarter of the space. This restriction is in place to mimic a real world

[3] VoronoiLimit can be downloaded at
http://www.mathworks.com/matlabcentral/fileexchange/34428-voronoilimit

[4] polygeom can be downloaded at
http://www.mathworks.com/matlabcentral/fileexchange/319-polygeom-m

scenario where all agents would be dropped off together in a central area, and then need to move to their desired positions. Next, the maximum speed of the agents is defined to be $v_{max} = 0.025$; this means that agents can move at most 2.5% of the length of $[0, 1]$ in one time-step. Then the α value for our sensing function is defined to be $\alpha = 0.05$. Lastly, constants are defined to restrict the values of drift. The drift vector is randomly initialized to be any angle between 0° and 360°, and its magnitude is restricted to be greater than 0 but less than or equal to 0.005, 20% of the agents' max speed, at all times. Between time-steps, the drift's magnitude can only change by 0.0005, or 10% of the maximum drift. The change in angle is restricted to be no more than 1°. It is assumed that the agents can sense the drift after it has influenced them, and so they have access to the drift vector from the previous time-step.

Deploy Agents and Compute Initial Values

The first step of the algorithm is to randomly deploy the agents as described above, and randomly initialize the drift. An example of the random deployment of the agents for both distributions can be seen in Figure 4, with agent positions in green and Voronoi borders in magenta. Each time-step is treated independently, and so a loop is run until a desired number of iterations have been processed. First, the contour plot of the density function is drawn. The S-limited Voronoi partitions are then found, which are the Voronoi partitions intersected with the boundary of S, using voronoiPlot. This function returns the coordinates of every agent's Voronoi cell, their center of mass, and the $\mathcal{H}_{exp}$ in the current time-step. Next, the drift force that will occur in this time-step is determined.

Determine Agent Movement

Next, a loop runs through all agents to determine their movement. First, the displacement $\vec{d}$ is calculated from the agent to its center of mass, and then the agent calculates how it should move to counteract the drift.

If $\|\vec{d} - drift_{past}\| \leq v_{max}$, then the agent can move to its desired position without violating the maximum speed condition. The idea is that after the drift is applied, the net movement $move$ of the agent becomes

$$move = \vec{d} - drift_{past} + drift$$

If $\|\vec{d} - drift_{past}\| > v_{max}$ and the vectors are not collinear (to avoid division by zero), then we employ a much more complicated algorithm which is described in detail in Section 3.1.4.

Finally if $drift_{past}$ and $\vec{d}$ are collinear, then the agent simply wishes to move along $\vec{d}$ and thus the knowledge of $drift_{past}$ is irrelevant. In this case, $move$ is calculated as

$$move = v_{max} \cdot \frac{\vec{d}}{\|\vec{d}\|} + drift$$

Final Computations and Metrics
The next step is to calculate the number of Voronoi neighbours of the agent. If an agent chooses to communicate in the current time-step, then the count for the number of communications is increased by its number of neighbours. After each agent has been moved to its next location, $drift_{past}$ is updated by storing $drift$. There are several other calculations done in this loop, but they are only performed for specific triggering strategies and not in the general case. After the above loop has concluded, the value of $\mathcal{H}_{exp}$ in each time-step is examined in reverse order until they drop below 98% of the final value. The time-step after this occurs (i.e. the first time it is within 2% of the final value of $\mathcal{H}_{exp}$) is defined as the stopping time.

3.2.2 voronoiPlot()

voronoiPlot calculates and plots the Voronoi partitions and returns the calculated values for use in moveVoronoi.

It first finds and plots the Voronoi partitions, as well as the positions of the agents, by calling VoronoiLimit. The called function returns the Voronoi cells for each agent. Then a loop is run through each cell to find the center of mass. The sub-triangulation process performed when it calls subTriangulate is described in detail in Sections 3.1.3 and 3.1.5. Lastly, the overall $\mathcal{H}_{exp}$ is found by summing the values of $\mathcal{H}_{exp}$ over the individual triangles.

3.2.3 subTriangulate()

subTriangulate recursively triangulates a given triangle until a desired number of iterations have been performed.

Recursively Triangulate the Area
First, the function finds the centroid of the given triangle using polygeom. It then creates three new triangles from the centroid and adjacent vertices of the original triangle in the same way that it is done in voronoiPlot. Then, if it has not met the desired number of sub-triangulations, it calls itself again on each of the three new sub-triangles. It then loops through each of the triangles that it has made and calculates their centers of mass using (26) and (27).

Numerically Integrate to Calculate $\mathcal{H}_{exp}$
The last step is to calculate the $\mathcal{H}_{exp}$ of the overall triangle that was given to the function. Due to the complexity of calculating a number of integrals, a weighted sum approximation is used instead. Similar to finding the weighted centroid, the information density is evaluated at the vertices and then weighted by the sensing function. These values are then averaged for all three vertices to get a weighting factor for the overall triangle. This weighting factor is then multiplied by the area of the triangle, found using polygeom, to approximate the integral over the triangle.

This information is passed back to voronoiPlot to be summed along with all other triangles to find the overall value of $\mathcal{H}_{exp}$ in the current iteration.

3.2.4 VoronoiLimit()

VoronoiLimit is an open-source function that was modified slightly for use in this simulation. It finds the Voronoi partition for a given set of agent positions, and then intersects the partitions with the boundary of the space S. The end result is that the Voronoi partitions are limited to the space S. After calculating these values, it then plots the boundaries of the Voronoi cells as well as the positions of the individual agents.

3.2.5 polygeom()

This open-source function calculates the area and centroid of a given polygon. In this context, it was used to find the centroid and area of triangles at multiple points in the triangulation process.

4 Design of Triggering Strategies

With the MATLAB simulation described previously, agents communicate at every time-step. We call this triggering strategy the full communication base case. Now, using metrics to analyze the base case, triggering strategies can be designed to make better use of the agents' communication abilities. In what follows, the strategies are evaluated using the Islands distribution, given by equation (6) and shown in Figure 3, as its asymmetric nature would lead to more robust strategies relative to the symmetric Volcano distribution, given by equation (5) and Figure 2.

4.1 Metrics of Comparison

Metrics of comparison must be defined to determine the benefits and drawbacks of the control policies. In this work, three metrics of comparison were considered: the value of $\mathcal{H}_{exp}$ collected in the final step, the total number of communications over a fixed number of time-steps, and the settling time of $\mathcal{H}_{exp}$. The methods used to calculate these values can be found in Section 3.2.

Cumulative distance travelled was considered as an additional metric, but was not used in determining the optimal policy. This is an important consideration as both communication and motor use require the consumption of power. Thus, a control policy which reduces total communications but requires additional movement may incur a higher net consumption of power and so such policies are not considered. In this work, all control policies that result in a cumulative distance travelled that is less than the base case are considered equal by this metric.

4.2 Control Strategies

4.2.1 Full Communication

The first strategy considered is the full communication case, which is analogous to applying no communication control to the system. In this strategy, each agent communicates their current position to all of their neighbours at every time-step.

This case is of particular interest since, given the constraints imposed by the simulation, the largest expected $\mathcal{H}_{exp}$ is achieved using this strategy. This is a logical conclusion as each control policy imposes restrictions on the frequency of communication, and hence reduces the accuracy and amount of available information. This fact makes the full communication strategy a convenient base case for comparison.

4.2.2 Constant Triggering Radius

By imposing restrictions on the agents' ability to communicate, uncertainty in the positions of their neighbours is introduced. Without the precise location of its neighbours, an agent cannot accurately calculate its Voronoi partition, requiring the use of a different partitioning scheme. Nowzari and Cortés proposed a solution [7] to this issue using the notion of guaranteed and dual-guaranteed Voronoi partitions, as defined by Sember and Evans [2]. Unfortunately, determining these partitions and thus the optimal position for an agent to move towards is computationally expensive and diminishes the savings from communication. An alternative approach was taken by Heemels, Johansson, and Tabuada [4], which was the inspiration behind the creation of the constant triggering radius policy described below.

Definition 4.1. (Constant Triggering Radius Communication Condition): *Agent i communicates its position if*

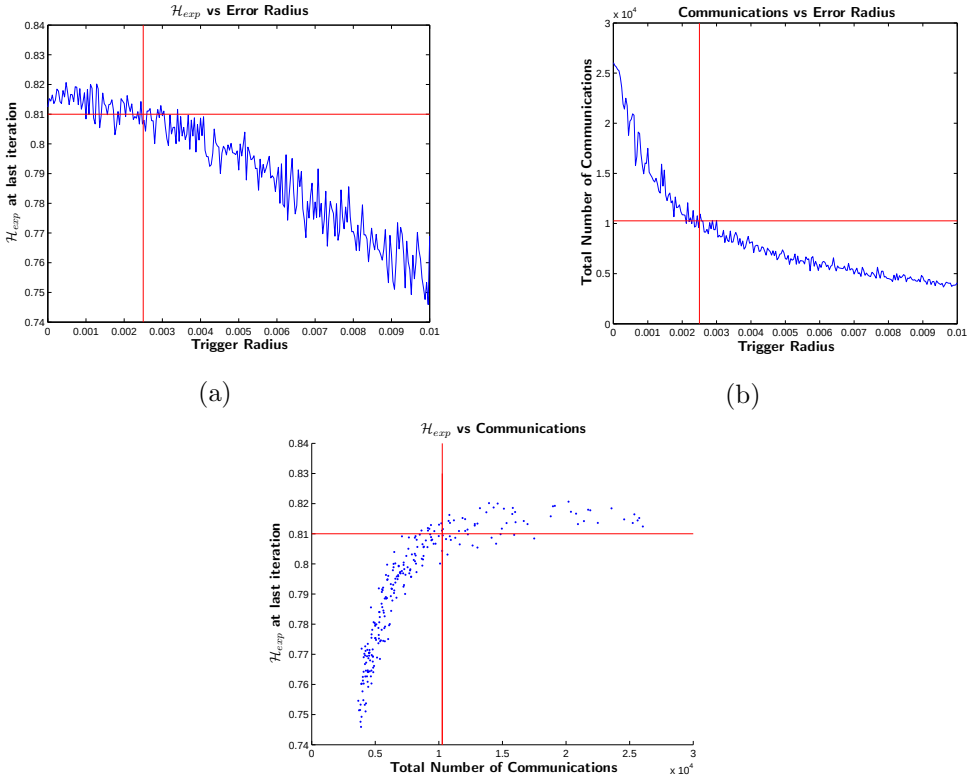
$$\|p_i^t - \tilde{p}_i\| \geq r \tag{33}$$

Where p_i^t is the position of agent i at time t, $\tilde{p}_i$ is the last communicated position of agent i, and r is the triggering radius.

In this strategy, each agent is responsible for communicating their position to their neighbours instead of measuring their neighbours' positions. The agents use an event triggering policy and communicate when the inequality in (33) is satisfied.

One can interpret this bound as a promise each agent makes to their neighbours; specifically, that they will remain within a radius r of their last communicated position. Furthermore, since r is chosen to be small, the Voronoi partitions of each agent can be well approximated using the last communicated positions of an agent and its neighbours, instead of their current positions. An implementation of this strategy can be found in Section 3.2.1.

One can easily see that the performance of this strategy is heavily dependant on the choice of r. The following plots were created by performing 251 trials, varying r from 0 to 0.1.



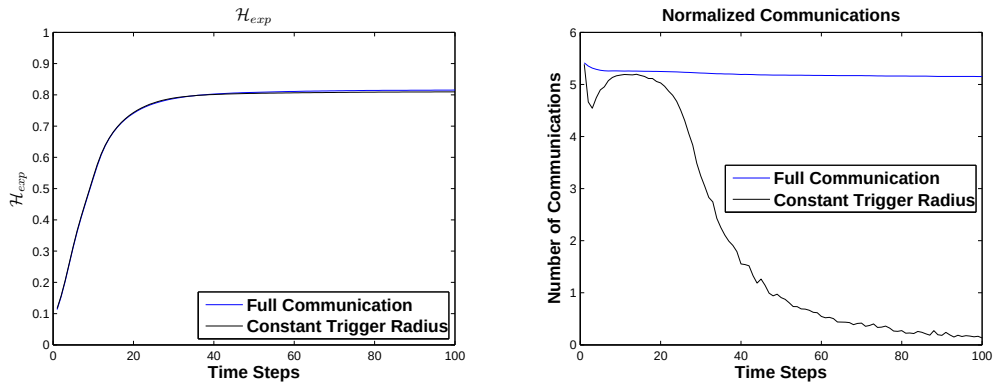
(a) (b)

(c)

Figure 5: 2D plots showing the selected constant trigger radius of 0.0025.

The above set of plots can be interpreted as a single 3-dimensional plot of trigger radius, final $\mathcal{H}_{exp}$, and total number of communications. As mentioned previously, all values were found using the Islands distribution. Examining Figure 5c, a value corresponding to an $\mathcal{H}_{exp}$ of 0.81 and total number of communications of 10 270 was selected, as marked by the red lines. Due to the lack of an explicit cost function, this point was chosen to balance the trade-off between communications and $\mathcal{H}_{exp}$, while prioritizing a high value of $\mathcal{H}_{exp}$. Increasing the number of communications any further would result in significant diminishing returns in $\mathcal{H}_{exp}$ gain. Examining figures 5a and 5b, the same point is indicated by the red lines and corresponds to an r of 0.0025. This plot also highlights the communication savings associated with using a larger trigger radius; again it can be seen that using an r of 0.0025 offers large savings over a radius of 0, which is the full communication case. It is important to note that if a different trade-off between $\mathcal{H}_{exp}$ and communications

is desired, a different point on Figure 5c can be chosen which would result in a different choice of r. For example, if an explicit threshold on $\mathcal{H}_{exp}$ was set, one could find a value of r that preserves this threshold while still offering a reduction in communications. Lastly, it is important to note that the optimal value of r is influenced by the agent's ability to estimate the drift, or equivalently the amount an agent tends to move when attempting to remain stationary. The vast majority of the communication savings occur after the settling time, which is when the agents are moving almost exclusively to counteract the drift.



(a) (b)

Figure 6: Comparison between full communication and constant trigger radius with $r = 0.0025$ using the Islands distribution

Figure 6a demonstrates that both strategies have nearly identical information gathering capabilities, with the full communication case converging to a slightly higher $\mathcal{H}_{exp}$ value. Conversely, as Figure 6b displays, the necessary communications are drastically different. While the agents are moving to their optimal positions the constant trigger strategy communicates nearly as much as the full communication case. However, after approaching the near-optimal positioning, there are significant communications savings from using the constant trigger strategy. This occurs because the agents are no longer moving over large distances, and thus often stay within r of their last communicated position. The constant triggering radius does not prevent all communication in the limit since the randomness in the drift can cause agents to exceed a distance r from their last communicated position and thus communicate.

4.2.3 Non-Constant Triggering Radius

The constant triggering radius strategy provided excellent results; however, there is no guarantee that it is an optimal strategy. As such, further effort was made in an attempt to obtain even better results. As Figure 6b displays, the constant

triggering radius strategy has close to the same number of communications as the full communication case at the beginning of the simulation. Thus, triggering strategies which communicated less during the initial time-steps were considered. Initially the radius was made as a function of time but since the time to convergence is dependant on the distribution, this idea was determined to be impractical in general.

For the second triggering strategy, the target period for reducing the number of communications was from the beginning until approximately the 40th time-step. The reason that the constant triggering strategy constantly communicated in this period was because the agents move a large amount and thus almost always exceed a distance r from their last communicated position. This led to the idea of a triggering radius which was proportional to the distance the agent had moved. Unfortunately this resulted in erratic communication and a poor amount of information collected. The following definitions are used to describe the non-constant triggering strategy.

Definition 4.2. (Neighbouring Set): *N_i^t is the set of the last communicated positions $\tilde{p}_j$ of the neighbours of agent i at time t, given by*

$$N_i^t = \{\tilde{p}_j : j \text{ is a neighbour of } i\} \tag{34}$$

Definition 4.3. (Average Distance to Neighbours): *β_i^t is the average distance between agent i and its neighbours, which is computed as*

$$\beta_i^t = \frac{1}{|N_i^t|} \sum_{n \in N_i^t} \|p_i^t - n\| \tag{35}$$

The non-constant triggering strategy uses the average distance to an agent's neighbours. During a period where an agent is moving rapidly, the average distance to its neighbours also experiences rapid change. This allows for the use of a larger triggering radius without being completely dependent on the movement of the agent. In addition, the agents are not aware of the amount of information they are collecting in comparison to the overall information available; however, agents will cluster more around the areas of high information and thus have a smaller average distance to their neighbours. By multiplying the average distance by the triggering radius (36), the agents can use a smaller radius if they are positioned in an area of high information, and use a larger radius if they are in an area of low information. This results in a higher $\mathcal{H}_{exp}$ measurement in the areas of high information, and fewer communications in areas of low information. Each agent is allowed to calculate their own triggering radius based on the available local information as an agent is not required to know the triggering radii of their neighbours. Simply put, each agent ensures they remain close enough to their last communicated position so the Voronoi partitions remain accurate. When an agent determines its Voronoi partition is no longer accurate (by leaving the ball defined by their calculated triggering radius), the agent transmits its locations to its neighbours.

Definition 4.4. (Triggering Radius Function): *The triggering radius can be computed as a function of $\tilde{p}_i$ and N_i^t as follows:*

$$r(\tilde{p}_i, N_i^t) = \begin{cases} r_{close} \cdot \beta_i^t, & \text{if } |\beta_i^t - \beta_i^{t-1}| < \delta \\ r_{far} \cdot \beta_i^t, & \text{otherwise} \end{cases} \tag{36}$$

Where r_{close}, r_{far}, and δ are design parameters.

Definition 4.5. (Non-Constant Triggering Radius Communication Condition): *Agent i communicates its position if*

$$\|p_i^t - \tilde{p}_i\| \geq r(\tilde{p}_i, N_i^t) \tag{37}$$

The design parameters r_{close}, r_{far}, and δ in Definition 4.4 were selected in a similar fashion to the radius r discussed above in Section 4.2.2. In this case since there are multiple parameters, two parameters were fixed and the third varied. By examining the output plots, a value which was considered optimal was chosen and the parameter fixed to this value. This process was applied iteratively, until all variables were varied and optimal points selected. This process resulted in the values $r_{close} = 0.021$, $r_{far} = 0.16$, and $\delta = 0.0025$. Thus the triggering radius function becomes

$$r(\tilde{p}_i, N_i^t) = \begin{cases} 0.021 \cdot \beta_i^t, & \text{if } |\beta_i^t - \beta_i^{t-1}| < 0.0025 \\ 0.16 \cdot \beta_i^t, & \text{otherwise} \end{cases} \tag{38}$$

Optimizing the design parameters resulted in $r_{close} < r_{far}$. This was as expected, since r_{far} is used when the agents are moving rapidly, and thus they can use a larger triggering radius to save some communications as they move. During movement phases the precise measurement of $\mathcal{H}_{exp}$ is not important, so the agents do not have to be positioned optimally until they are close to their final resting position.

Similar to the constant triggering radius strategy, the above approach was implemented in the moveVoronoi function by first calculating β_i^t followed by $r(\tilde{p}_i, N_i^t)$.

5 Results

5.1 Overall Trends

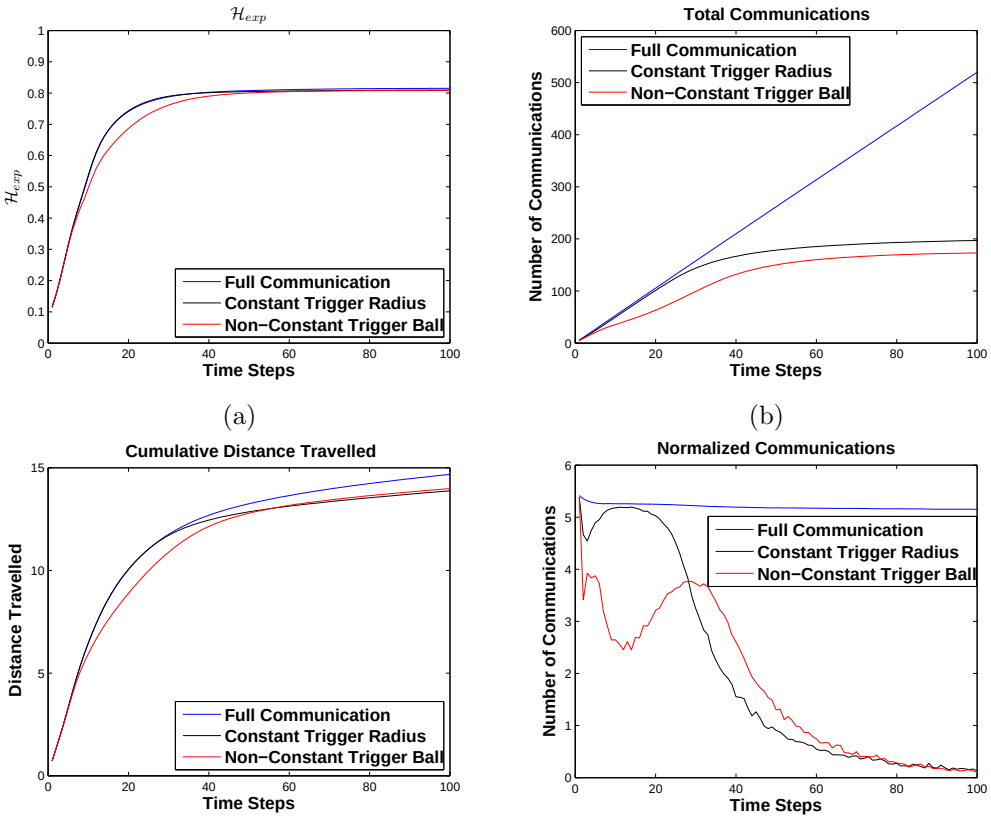


Figure 7: Average trends for the Islands distribution over 50 trials

		Performance Metric (% Difference from Base)		
		$\mathcal{H}_{exp}$	Total Communications	Settling Time
Islands	Full	0	0	0
	Constant	-0.72	-62.10	-10.75
	Non-Constant	-0.82	-66.70	+13.26
Volcano	Full	0	0	0
	Constant	-0.63	-68.30	-8.68
	Non-Constant	-0.84	-70.74	+24.37

Table 1: Comparison of all strategies for both information densities

Overall, both triggering strategies offer minimal decreases in the amount of information gathered for significant savings in the number of communications. Furthermore, both strategies decrease the total distance travelled by our agents, and thus they meet all of our design criteria.

Their performance is first studied when subjected to the Islands distribution (6). Examining the average trend in $\mathcal{H}_{exp}$ seen in Figure 7a, it can be seen that the full communication and constant triggering strategies follow a very similar pattern. In fact, as seen in Table 1, on average they converge to values within 0.72% of one another. The non-constant triggering strategy lags behind both of the others in movement, but converges to an average $\mathcal{H}_{exp}$ that is only 0.82% lower than in the full communication case.

When examining the total distance travelled in Figure 7c, once again the full communication and constant trigger strategy are very similar in overall trend. The constant strategy moves slightly less and takes an average of 10.75% less time to converge to its optimal position, as seen in Table 1. The average trend for the non-constant trigger strategy is to lag behind both of the others once again, and to converge to a value that is slightly above the distance travelled by the constant strategy. The distance travelled does not go to zero when the agents have reached an optimal position due to the effect of drift that agents must always fight. Finally, the trend in the full communication case is to travel more when the agents have reached their optimal position relative to the other two strategies. The lack of this behaviour would lead to drastic savings in movement as the number of time-steps increases.

The most important result of this project can be seen by viewing the communications in figures 7b and 7d. As is expected, the full communication case increases its communications in a linear fashion with time. This trend is not perfectly linear as the number of neighbours that each agent has can change in time since more agents will have Voronoi borders on the boundary of S. When examining the other two triggering strategies, Figure 7d shows that both strategies achieve their biggest savings after the agents have moved into their near-optimal positions. As discussed in Section 4, the constant trigger strategy communicates at almost every time-step as the agents quickly disperse themselves across the space. However, the agents quickly settle into a near-optimal position and thus communicate significantly less as time goes on. The non-constant strategy achieves its savings relative to the constant by communicating much less as the agents disperse themselves across the space. However, this lack of precision in their Voronoi partitions as they move means that they take longer to settle into their near-optimal positions after dispersing and thus communicate more for a longer period of time. The net effect is a reduction in the number of communications over both of the other strategies, as seen in Table 1.

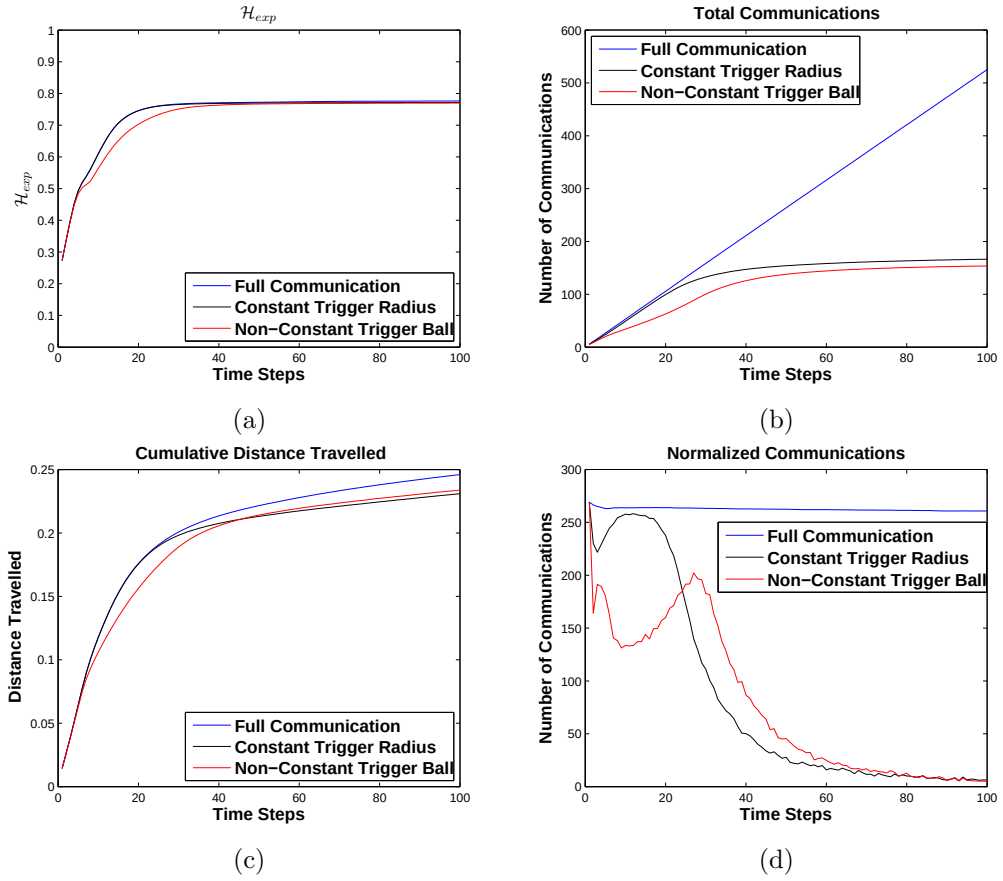
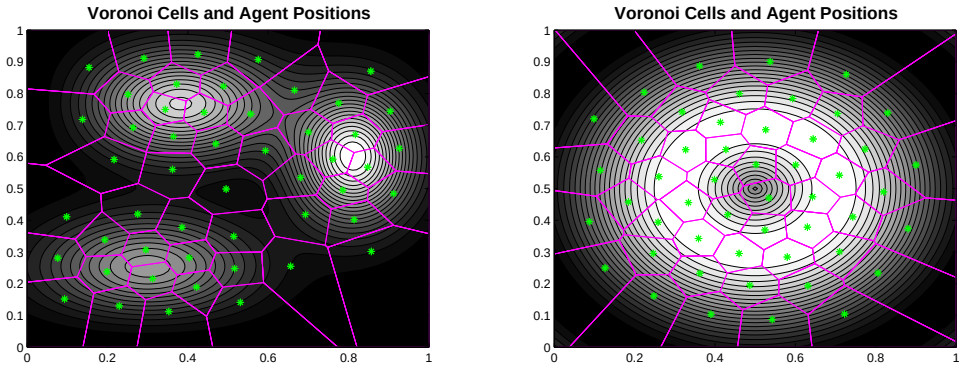


Figure 8: Average trends for the Volcano distribution over 50 trials

In Figure 8, it can be seen that many of the trends observed for the Islands distribution also appear for the Volcano distribution. Most notably, there is a drastic decrease in the number of communications for a minimal sacrifice in $\mathcal{H}_{exp}$, which is made explicit in Table 1. The largest difference between the two distributions is in the performance of the two triggering strategies relative to one another. In Figure 8d we see the same trend in communications as in Figure 7d, however the final difference between the two triggering strategies is much less.

Another difference between the two distributions is the overall trend in $\mathcal{H}_{exp}$, as seen in Figure 8a. At first all the strategies perform similarly, since agents are deployed in the middle of the distribution where the information density is highest. However, once they start to move past the ring of highest information, their rate of increase in $\mathcal{H}_{exp}$ drops as they transition through the areas of lower information. After this phase, the non-constant strategy once again lags behind as it takes longer to adapt its positioning while the agents are moving quickly and thus have a larger triggering radius.



(a) Islands Distribution (b) Volcano Distribution

Figure 9: Example of the agents' final positions and Voronoi cells

An example of the final positions of the agents after the simulation has concluded can be seen in Figure 9. As before, the agents' positions are green while the Voronoi borders are magenta. First, it can be seen that in areas of high information the Voronoi cells resemble hexagons. This is to be expected, as this is the most efficient shape to equally distribute the information in the uniform case. Furthermore, it is evident that in areas of high information the agents are closely packed; this is because their ability to gather information diminishes according to the sensing function. In areas of lower information, cells become increasingly large since there is very little information to be gathered.

The plots presented in Figure 9 represent the final resting positions when using the non-constant triggering strategy. It was noted throughout the project that the different strategies did not necessarily lead to different coverage patterns, and that the initial position of the agents had a much larger effect on their final positions. Once again this is to be expected, since Lloyd's algorithm only guarantees convergence to a locally optimal positioning. The lack of difference between the algorithms also makes sense since the difference in final values of $\mathcal{H}_{exp}$ are so low, as seen in Table 1. The visual output when running the full communication case and the two triggering strategies looked identical to one another.

5.2 Variance in Results

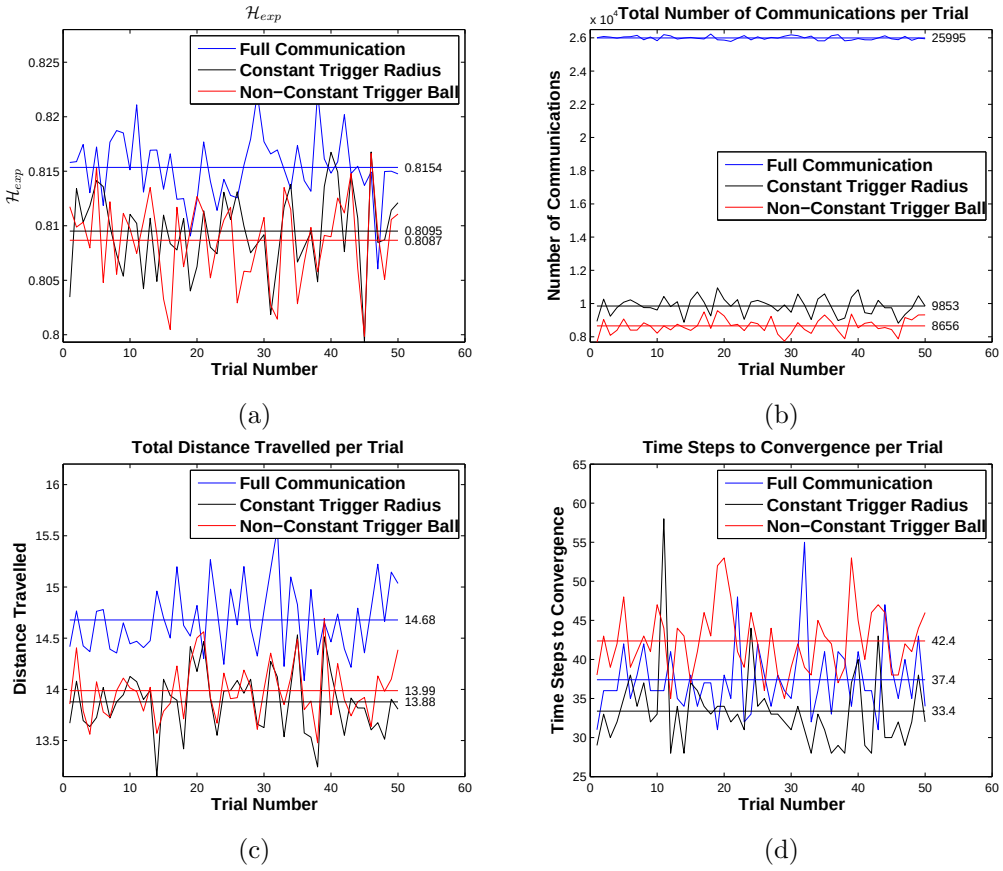


Figure 10: Metric values and means for the Islands distribution over 50 trials

As Figure 10a displays, the constant and non-constant triggering strategies produce very similar $\mathcal{H}_{exp}$ values, with the constant strategy performing slightly better than non-constant on average. As expected, the full communication case has the largest mean $\mathcal{H}_{exp}$ by a reasonable margin. It can also be seen that the full communication, constant triggering, and non-constant triggering strategies all vary by similar amounts. As a result, there are scarce scenarios where the constant and non-constant strategies can produce the greater $\mathcal{H}_{exp}$ value.

As Figure 10b makes evident, the number of communications used by the full communication strategy greatly exceeds that of the other two strategies. It is also notable that the variance in the number of communications relative to the mean is low in comparison to the other metrics. Even the constant and non-constant strategies have a small range in the number of communications where they have the potential to cross. Given these observations, it is likely the non-constant strategy

will use less communications than the constant strategy, and it is safe to assume that the full communication strategy will use the most communications.

In Figure 10c above, it is once again evident that the mean and variance values of the constant and non-constant triggering strategies are very similar. This is comparable to the trend in $\mathcal{H}_{exp}$ seen in Figure 10a, but with the constant strategy slightly out-performing the non-constant case. In general, it is likely the constant and non-constant strategies will require less movement than the full communication strategy, once again demonstrating the potential to save power through the reduction in communication.

As seen in Figure 10d, time-steps to convergence has a large variance relative to the mean, making it the most volatile of the four comparison metrics. Over the 50 simulations, almost all strategies had at least one simulation that crossed over the mean of the other strategies. This is particularly evident with the constant strategy, which has the lowest mean but the greatest single time to convergence over all strategies and simulations.

6 Discussion

6.1 Discussion of Metrics

There were two primary performance metrics defined for the scope of this project: the final value of $\mathcal{H}_{exp}$ and the number of communications. A third performance metric that was considered was the settling time, but this was not a main focus of the project.

Final value of $\mathcal{H}_{exp}$
When looking at figures 7a and 8a, the full communication strategy has the largest final $\mathcal{H}_{exp}$ value by a small margin. This is because the agents continuously move towards the exact center of mass and communicate continuously. For the triggering strategies each agent has an open ball defined by their triggering radius, henceforth referred to as the trigger ball, which encompasses the center of mass so that their optimal position is not broadcast. Instead, the position communicated by the agent could be a distance r away from the center of mass which in turn means the final Voronoi partitions are suboptimal. This results in a slightly lower $\mathcal{H}_{exp}$ value. As the triggering radii used once agents reach equilibrium approach 0, the final value of $\mathcal{H}_{exp}$ approaches the limit given by the full communication case.

Number of communications
Figures 7b, 7d, 8b, and 8d give insight to the effect on the number of communications by the triggering strategies. The total number of communications for the full communication case increases in a linear fashion as expected. The triggering strategies have a decrease in communication in time-steps 0 to 5 because the agents deployed in the middle of the cluster remain stationary. The agents that were distributed

on the outside perimeter have to escape their trigger ball first and communicate their new position before the inside agents know they should begin moving larger distances. The non-constant trigger strategy has the decrease in communication in time-steps 10 to 20 whereas the constant trigger strategy is essentially always communicating because $r < r_{far}$. In this time period, the agents are moving rapidly which means in the constant strategy they are consistently leaving their trigger ball whereas the non-constant was designed specifically to account for this.

Settling time
The more noticeable trend of figures 7a and 8a is the lower value of the non-constant $\mathcal{H}_{exp}$ curve relative to the constant and full communication $\mathcal{H}_{exp}$ curve around the 20th time-step. This difference occurs because of the large radius of the non-constant trigger ball r_{far}. Due to the radius being large, the information that the agents receives is not as updated as the other cases and and so its $\mathcal{H}_{exp}$ value does not increase as quickly in this stage. As a result, the non-constant strategy takes longer to reach its equilibrium point which results in a larger settling time.

6.2 Strategy Comparison

Both the constant and non-constant strategies provide significant communication savings in comparison to the full communication strategy, provided the user has some tolerance in the amount of information collected. Due to the lack of an explicit cost function, determining the best strategy is not entirely clear; however, given our results the constant strategy has the best performance. Using the results from the Islands distribution seen in Table 1, the constant strategy collected 12.2% more $\mathcal{H}_{exp}$ relative to the loss from the non-constant strategy. In contrast, the non-constant strategy saved 6.9% more communications relative to the savings obtained by using the constant strategy, but took 21.2% longer to converge. The constant strategy had even better performance gains compared to the non-constant strategy over the volcano distribution. In this case, constant collected 25% more $\mathcal{H}_{exp}$ relatively and converged 26.6% faster while only using 3.4% more communications. Thus, the constant strategy also tends to perform better when not optimized for the given information distribution.

Despite the constant strategy out-performing the non-constant one, it is by no means considered to be optimal. The strong performance of the non-constant strategy demonstrated the potential of such strategies to outperform the constant case, further reducing the impact of the trade-off between $\mathcal{H}_{exp}$ and the number of communications.

6.3 Application of Results

As discussed earlier when comparing the constant and non-constant triggering strategies, the preferred strategy will be determined based on how valuable communications are relative to the information being gathered for the selected application.

In this section, the results are applied to the oceanographic data collection network which was discussed previously as a motivation for the project [5]. After researching the hardware used in these aquatic agents, it was determined the communication system was an acoustic system designed specifically for underwater use. Specifically the antenna was omnidirectional, which allows the agents to communicate with maximum gain in the plane horizontal to the agent. The specifications for this transmitter were researched [3] and the total power usage dedicated to communications was determined for a single agent who was part of a 50 agent network. One assumption made from observations of the model was that the agents would be initiating less short range communications and more long range transmissions as they spent more time spread out than clustered at the beginning of the simulation. This assumption was relevant because the power requirements to make a communication were based on the distance of the link. Table 2 contains the power usage for communications based on the triggering strategy, which were found using the costs detailed in [3] and the results shown in Table 1.

Strategy	Total Power Usage for Communications
Full Communication	19,697 W
Constant Triggering	7,465 W
Non-Constant Triggering	6,559 W

Table 2: Comparison of communication power usage between strategies

Another consideration that should be noted when evaluating this data is that this is the power required for the first 100 time-steps, which was enough time to allow the network to settle near equilibrium. In reality once the network has settled there will still be some communication, most notably in the case of full communication, but it was ignored for this analysis.

Looking at the presented results, trying to monetize these values from a cost of electricity perspective is fruitless as the cost of a KWh is on the order of \$0.15 and the power savings here are not on that order of magnitude. The benefits derived from the power savings are less monetary and more based on the flexibility they provide the agents to carry smaller power supplies, and focus more power on movement, navigation, and data collection.

Weighing the cost of the lost network information against the gain of reducing communication power in the oceanographic application [5], each agent was collecting on average 200 data points per sample, so a cost of 0.72% or 0.82% of network coverage for the constant and non-constant strategies respectively would only result

in a loss of one or two data points per sample. This is likely insignificant, so in the oceanographic data application, adopting the constant or non-constant triggering strategy would be appropriate as there are sizeable savings in communication and minimal loss in information coverage.

While some consideration was given to how these strategies affected the distance travelled by the agents in our research, it is likely that these savings might have proven more quantifiable in terms of monetary benefits in comparison to the communication reductions. While the results were not fully explored, it was shown that both of the developed triggering strategies resulted in a reduction in distance travelled by the agents. Even without quantifying these in terms of actual savings, these findings further reinforce the policy of implementing one of the developed strategies in the oceanographic application.

6.4 Further Applications

While potentially beyond the scope of the actual strategies developed in our research, there are numerous other aquatic network applications where extensions of this research could be applied. The application of this project to these areas could have a wide range of environmental and societal impacts.

Consider an application where small, lightweight autonomous agents are deployed immediately after an oil spill. They are tasked with tracking the borders of the spill so clean-up units can know the coordinates as they develop over time. If all vessels with the potential to spill harmful contents were required to carry these agents for emergency deployment, having minimalistic hardware would be important to reduce costs. With a lightweight agent design, saving on communication and movement costs would be important, and obviously tracking the spill accurately would also be critical.

The environmental benefits of this application could be substantial if the network of agents were able to provide valuable information to clean-up efforts which enabled the impact of the spill to be minimized in comparison to the situation without the assistance of the autonomous network.

Depending on where a specific spill took place, it could impact human activity if the contents reached the shoreline of beaches, private residences, or commercial locations. The described application has the potential to reduce all of these risks, making further research a worthwhile endeavour.

7 Conclusion

In this project, two triggering strategies were developed which restricted a given agent's communication while it remained within an established trigger ball. The radius of the trigger ball was either constant or dependent on the average distance between the given agent and its Voronoi neighbours. Both of the implemented strategies significantly decreased the frequency of communication within the network with minimal losses in information gathered by the network. It is not clear that either strategy is superior, but by using the methods described in this project it is possible to determine which strategy would be better suited for a specific application.

Having achieved a favourable trade-off between communication reduction and information loss in each of the strategies, both were considered acceptable for implementation in the aquatic network application which acted as the inspiration for the simulation model.

7.1 Future Work

If there was more time available to continue this project, a variety of different paths could be taken. With the work already done, there is future theoretical work, engineering design work, and simulation work that could be pursued.

For theoretical work, it would be a significant achievement to derive a rigorous and tight bound on performance. Cortés and Nowzari [7] were able to prove a bound on convergence but were unable to provide a bound on the performance of their algorithm.

For engineering design work, it would be interesting to implement the algorithms on physical hardware. This would bring insight to constraints that were not considered in the model as well as provide experimental values to power consumed by the motor and communication devices.

Finally, a new algorithm or simulation direction could be to focus on minimizing the movement of the agents. Considering that the motor would consume a significant amount of energy, minimizing the distance the agents move would have a large impact on increasing the battery life. A cost function which is a function of energy consumed by the motor and communication could also be designed. Furthermore, the model could be extended to be in three dimension or equivalently $S \in \mathbb{R}^3$.

References

[1] F. Bullo, J. Cortés, and S. Martínez. *Distributed Control of Robotic Networks.* Applied Mathematics Series. Princeton University Press, 2009. Electronically available at http://coordinationbook.info.

[2] W. Evans and J. Sember. Guaranteed voronoi diagrams of uncertain sites. In *Canadian Conference on Computational Geometry, 2008*, 2008.

[3] EvoLogics. Underwater acoustic communication system s2cm series product information, January 2015.

[4] W.P.M.H. Heemels, K.H. Johansson, and P. Tabuada. An introduction to event-triggered and self-triggered control. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 3270–3285, Dec 2012.

[5] G.A. Hollinger, S. Choudhary, P. Qarabaqi, C. Murphy, U. Mitra, G. Sukhatme, M. Stojanovic, H. Singh, and F. Hover. Underwater data collection using robotic sensor networks. *Selected Areas in Communications, IEEE Journal on*, 30(5):899–911, June 2012.

[6] R. Murray, Z. Li, and S. Sastry. *A Mathematical Introduction to Robotic Manipulation.* CRC Press, 1994. Electronically available at http://www.cds.caltech.edu/ murray/courses/primer-f01/mls-lyap.pdf.

[7] C. Nowzari and J. Cortés. Self-triggered coordination of robotic networks for optimal deployment. In *American Control Conference (ACC), 2011*, pages 1039–1044, June 2011.

[8] P. Tabuada. Event-triggered real-time scheduling of stabilizing control tasks. *Automatic Control, IEEE Transactions on*, 52(9):1680–1685, Sept 2007.

Appendix

Please see the attached CD for all of the MATLAB codes that were used in this project.