Titre: Title:	Risk-Aware Swarm Exploration
Auteur: Author:	David Vielfaure
Date:	2022
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Vielfaure, D. (2022). Risk-Aware Swarm Exploration [Master's thesis, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/10380/

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Risk-aware Swarm Exploration

DAVID VIELFAURE
Département de génie informatique et génie logiciel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie informatique

Juin 2022

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Risk-aware Swarm Exploration

présenté par **David VIELFAURE**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Heng LI, présidente
Giovanni BELTRAME, membre et directeur de recherche
Mohammad Adnan HAMDAQA, membre

DEDICATION

Practice does not make perfect.
It is practice, followed by a night of sleep, that leads to perfection.
. . .

Matthew Walker,
Why We Sleep: The New Science of Sleep and Dreams

ACKNOWLEDGEMENTS

First, I would like to thank Professor Giovanni Beltrame for giving me a chance to join the Mist Lab two years ago and pursue a master's degree in robotics. Coming from an aerospace engineering background, I had no significant knowledge of this field and there was no guarantee that my involvement in the laboratory would turn out fruitful. I am grateful to have received this opportunity, the last two years have been immensely enriching.

I would also like to thank Samuel Arseneault with whom I had the chance to spend a lot of time over the last two years. His coding meticulousness and work ethic have been of tremendous help in the various research projects we have worked on.

I must also thank Polytechnique Montréal, the Natural Sciences and Engineering Research Council of Canada and Mitacs for providing financial support which allowed me to pursue my master's degree.

Finally, I would like to thank my family and friends to have made the last two years enjoyable, even in the context of the pandemic. Having activities to look for beside the master's degree definitely helps keep a more balanced lifestyle.

RÉSUMÉ

L'exploration d'environnements inconnus est au coeur de plusieurs problèmes de robotique, parmi ceux-ci, des scénarios de sauvetage et des missions d'exploration extraplanétaire. Principalement, le problème d'exploration a été étudié pour des systèmes comportant un seul robot. Cependant, l'utilisation de plusieurs robots pour la réalisation de la mission pourrait s'avérer salutaire, en ce sens qu'avec une coordination adéquate, la vitesse à laquelle le terrain est parcouru devrait augmenter avec le nombre de robots dans le système. Ainsi, l'utilisation de systèmes multirobot, en opposition à un seul, se veut un domaine de recherche intéressant pour des applications d'exploration là où une importance est accordée à la vitesse à laquelle la mission s'effectue. Cependant, plusieurs défis demeurent, notamment en ce qui a trait aux fautes affectant les robots. Même si les systèmes multirobot présentent une certaine tolérance face aux risques en raison de leur intrinsèque redondance, il a été démontré qu'en pratique, la robustesse qui les caractérise peut être moindre que celle d'un unique robot. En effet, une faute affectant un seul robot peut facilement se propager à l'ensemble du système, causant une panne généralisée. En addition aux fautes, plusieurs autres problèmes affectent les systèmes multirobot. Parmi ceux-ci, on retrouve la coordination, la communication et le stockage de données. Le présent mémoire répondra à ces défis en présentant deux algorithmes de robotiques d'essaim:

- DORA-Explorer: Distributed Online Risk-Aware Explorer
- RASS: Risk-Aware Swarm Storage

En ce qui concerne DORA-Explorer, sa contribution principale est d'introduire une conscience du risque dans un algorithme d'exploration pour essaim de robots. Disposer d'une stratégie d'exploration adaptée est particulièrement important puisque sans coordination, les robots du système couvriront les mêmes régions de l'environnement. Un tel comportement ne se traduirait que par de faibles gains d'information et n'est donc pas souhaitable d'un point de vue du système. Alors que cette coordination pourrait être orchestrée de façon optimale depuis une station centrale, il est en pratique impossible de le faire en raison de limitations au niveau de la connectivité des robots du système. En effet, une coordination centrale nécessiterait une haute bande passante et une connectivité continue. DORA-Explorer cherche à répondre à ce problème en proposant un algorithme d'exploration qui maximise la quantité d'information recueillie tout en minimisant la quantité de risque auquel les robots s'exposent. L'algorithme ne nécessite aucune coordination centrale et possède des couts de calculs très faibles ce qui le rend particulièrement adapté pour une utilisation sur essaims de

robots. Le simulateur basé sur la physique ARGoS est utilisé pour tester les performances de DORA-Explorer et finalement des expériences avec des robots physiques sont effectuées pour confirmer l'applicabilité de l'algorithme dans un scénario réel. Les résultats montrent que DORA-Explorer obtient des résultats d'exploration convaincants tout en réduisant considérablement la quantité de fautes qui affecte les robots du système lorsque comparé à d'autres solutions présentes dans la littérature.

En ce qui concerne RASS, sa contribution principale est d'apporter une conscience du risque dans un algorithme de stockage et de routage complètement décentralisé. Le stockage de données demeure un défi pour des systèmes multirobot en ce sens que la quantité d'information collectée ne fait que croitre avec le nombre de robots dans le système. Encore une fois, la connectivité hasardeuse qui caractérise ces systèmes empêche l'envoi direct de l'information collectée vers une station de base pour un stockage permanent. Les robots doivent fréquemment stocker localement l'information collectée jusqu'à avoir un canal disponible pour l'envoyer vers un stockage permanent. De plus, puisque les robots ont généralement un rayon de communication limité, l'information doit être acheminée au travers de plusieurs autres robots avant de rejoindre la destination finale. Le système multirobot devient donc un système de stockage temporaire et décider où stocker l'information devient essentiel, particulièrement en présence de risque. Acheminer l'information par le chemin le plus court peut sembler naturel, cependant en présence de risque certains noeuds du système peuvent être trop dangereux pour être utilisés. Par exemple, envoyer de l'information à un robot situé à proximité d'une source de radiation pourrait causer des corruptions et n'est donc pas souhaitable. RASS cherche à répondre à ce problème en introduisant une conscience du risque pour déterminer où devraient être acheminées les données pour éviter le risque tout en percolant vers la station de base. Encore une fois, RASS est validée à l'aide du simulateur basé sur la physique ARGoS de même qu'à l'aide d'expériences sur des robots physiques. Nous obtenons des résultats convaincants qui montrent une diminution significative de la quantité d'information corrompue par les sources de radiations tout en conservant une vitesse de routage adéquate.

ABSTRACT

The exploration of unknown environments is at the core of numerous robotic applications from search-and-rescue operations to space missions. The problem has been mostly studied in single robot setups, but the ability to perform exploration with teams of robots opens the door to even more ambitious applications. With proper coordination, the time required to explore a given environment should decrease as the number of robots increases. Therefore, multi-robot exploration is an attractive solution to many time-critical applications such as search-and-rescue operations or planetary exploration. Moreover, multi-robot teams are usually resilient to some amount of robot failures. However, robot failures are still undesirable as they can affect team performance and should therefore be avoided. Multi-robot systems come with their own sets of constraints and challenges: among those, coordination, communication and data storage are the most relevant to the exploration problem. In this thesis, two swarm robotics algorithms addressing the aforementioned challenges will be presented:

- DORA-Explorer: Distributed Online Risk-Aware Explorer
- RASS: Risk-Aware Swarm Storage

DORA-Explorer's main contribution is bringing risk awareness to a swarm exploration strategy. We define by risk awareness the acknowledgement and mitigation of dangers that could lead to failures in the swarm. This is particularly relevant as without a proper strategy, the robots will inevitably explore overlapping parts of the environment, leading to little gains in terms of efficiency compared to single-robot solutions. While the coordination could be optimally orchestrated from a central computing station, such a solution would require a perfect connectivity maintenance with each robot and a high communication bandwidth since the robots would need to send their observations and receive their commands. This motivates the need for a decentralized exploration algorithm relying only on local computation onboard the robots and communication with their neighbours. To the best of our knowledge, there exists no risk-aware collaborative exploration algorithm that relies solely on local or shared information. Our decentralized exploration algorithm leverages distributed belief maps (DBMs) to maximize coverage and decrease robot failures caused by environmental hazards. To evaluate this system, we test it on the specific problem of hazard mapping in a 2D world discretized as a grid, in which a multi-robot team simultaneously explores a dangerous environment and collaborates to avoid hazardous locations as well as obstacles. We validate our approach in a physics-based simulator, ARGoS, in which we define a grid-based environment with multiple radiation sources. We then test it on physical robots. Results from the physics-based sim-

ulator show that DORA-Explorer reduces considerably the likeliness of robot failures while keeping similar ground coverage performance compared to other solutions proposed in the literature. In addition to the simulations, physical experiments were carried and confirmed the real-world applicability of our algorithm with convincing results.

As for RASS, its main contribution resides in bringing risk awareness to a storage and routing swarm algorithm. Data storage remains a challenge for such systems as the amount of collected information only increases with the number of robots. The unreliable connectivity that these systems typically suffer from inhibits sending collected data items directly to external storage. Robots often need to store locally the data items until a path towards permanent storage becomes available. Additionally, because robots usually have limited communication range, the data items collected during the mission may need to be routed through multiple robots before reaching the external storage infrastructure. The multi-robot system becomes a temporary storage infrastructure and deciding where to store and send the data items become essential. Routing the data items through the shortest path towards the base station may seem natural, however, because the environment into which the mission is carried is usually uncontrolled, environmental hazards can compromise some of the nodes of the system. For example, routing information through a robot located near a radiation source might cause data corruptions. Avoiding such nodes of the system can effectively increase the reliability of the system; thus risk should be considered when storing and routing data items. In RASS, a fully decentralized risk-aware storage system is proposed. RASS actively routes data items towards a base station while avoiding dangerous nodes of the system and relies solely on local interaction to determine which nodes are the fittest for storing information. Again, we validate our approach in the physics-based simulator ARGoS and obtain convincing results in terms of reliability and transfer speed. Physical experiments are also presented and show that the algorithm is easily transferable to physical robots and runs the way it is intended.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ACRONYMS

APC	Antigen Presenting Cell
CRDT	Conflict-Free Replicated Data Types
DBM	Distributed Belief Map
DORA-Explorer	Distributed Online Risk Aware Explorer
FBE	Frontier-Based Exploration
KNN	K-Nearest Neighbor
LOF	Local Outlier Factor
RASS	Risk Aware Swarm Storage
ROS	Robot Operating System
UV	Ultraviolet
VANET	Vehicular Ad Hoc Network

CHAPTER 1 INTRODUCTION

The growing accessibility to robots has enabled the rise of the new field of swarm robotics. It corresponds to the application of swarm intelligence to the field of robotics and will be detailed below in section 1.1. Robot swarms are particularly appealing for exploration missions involving large and unknown environments where a team effort can yield better results. Indeed, with proper coordination between the robots, the exploration rate should increase proportionally with the number of team members [1]. We define by exploration the action of covering an unknown environment and gathering information about it. Swarm systems should in theory be less prone to failures because of their inherent redundancy. However, it has been demonstrated that in practice, they do not benefit of this property and are, in some cases, even less reliable than traditional centralized robots systems in the presence of risk [2]. Designing specific tools for risk management in swarm systems is therefore of high importance [3] and has been the primary focus of this master's degree. This thesis presents two algorithms that improve tolerance to risk in robot swarm exploration. The first, named Distributed Online Risk Aware Explorer (DORA-Explorer), is a risk-aware exploration algorithm that minimizes the risk to which robots expose themselves while maximizing the amount of terrain covered, which is published as:

D Vielfaure, S Arseneault, P-Y Lajoie, G Beltrame. "DORA: Distributed Online Risk-Aware Explorer". 2022. IEEE International Conference on Robotics and Automation (ICRA).

The second algorithm, Risk Aware Swarm Storage (RASS), introduces risk-awareness at the storage and routing level of the swarm by levering a fitness policy based both on risk and hop-count, and was published as:

S Arseneault, D Vielfaure, G Beltrame. "RASS: Risk-Aware Swarm Storage". 2022. International Conference on Autonomous Agents and Multiagent Systems (AAMAS).

These two algorithms can be used on their own or side-by-side and are intended to improve the resilience of robots swarms carrying exploration missions in dangerous environments.

1.1 Definitions and Basic Concepts

Swarm intelligence is a discipline that focuses on the mechanisms leading to global order in systems composed of many individuals coordinating through local interactions. Swarm intelligence can be seen in both natural and artificial systems where the collective behaviors

are obtained in a fully decentralized fashion. Many examples of such systems can be found in nature, for example colonies of ants, schools of fish, flocks of birds or herds of land animals [4]. Systems that fall into the discipline of swarm intelligence display the following characteristics:

- A swarm system comprises many individuals and should be able to adapt to varying quantities of individuals
- There is no central coordination, agents' behaviors emerge only from local interactions
- The system is greater than the sum of its individuals. In other words, when taken individually, the agents are relatively incapable but collectively they achieve impressive results

Such characteristics result in an adaptable system that should, in theory, display better tolerance to risk. Indeed, the absence of central coordination not only eliminates a bottleneck that could limit scalability, it also removes a central point of failure that could compromise the entire system if taken down.

As for swarm robotics, it falls into the artificial category of swarm intelligence and more specifically corresponds to the application of the discipline to the field of robotics. It is at the intersection of collective robotics and swarm intelligence. Of course, because of the engineering nature of swarm robotics, physical limitations arise. Bandwidth restrictions, communication capabilities, storage limitations and physical constraints all need to be taken into account when designing robot swarm systems. Additionally, swarm robotics needs to respect some design principles to adhere to the characteristics of swarm intelligence. Specifically, there must be no centralized control, no predefined roles and the global behavior needs to emerge from simple and local interactions.

1.2 Problem Elements

Recently, a lot of attention from the research community has been oriented towards risk-aware autonomous systems. Taking risk into account when designing robot systems is of capital importance as they are increasingly deployed on real-world scenarios with safety-critical applications, including the transportation sector, aerospace systems or collaborative manufacturing. These robotic systems evolve in uncontrolled environments and must face risks on a daily basis, motivating the need for risk awareness. Excessive risk taking, or a complete absence of considering it, may not only jeopardize the robotic system, it may also endanger components in its vicinity and possibly human lives. For robot swarms tasked with

the exploration of unknown environments, risk is generally location-based and takes the form of environmental hazards. Other types of risk exist, however this work focuses on risk that arises from interactions with the environment and corresponds to interactions faults from section 2.6. The exploration of hazardous environments using teams of robots comes with its own set of advantages and constraints. Carrying the exploration mission with numerous robots should result in faster terrain coverage than in a single robot mission [1]. However, failures can considerably reduce the performance of the team and should therefore by avoided as much as possible. Not taking risk into account when exploring will inevitably lead to a decrease in performance of the team.

1.2.1 Terrain Coverage

Applied to a scenario of terrain coverage, the problem element of not considering risk is presented in figures 1.1 and 1.2. In figure 1.1, robots of the team start exploring the environment by covering as many cells as possible. After a while, in figure 1.2, robots start experiencing failures due to environmental hazards. In this example, risk takes the form of point radiation sources, however, any other type of danger could be considered (e.g., rough terrain or fire). Because of failures, the number of team members carrying the exploration mission decreases drastically. As a result, the exploration rate decreases and large portions of the environment remain uncovered. The intuition shown in figures 1.1 and 1.2 presents DORA-Explorer's problem statement. DORA-Explorer tries to solve it by introducing a consciousness of risk in the motion control loop of the robots.

1.2.2 Information Gathering

Naturally, a similar problem statement can be expressed in terms of information gathering. Again, carrying the mission with numerous robots should result in a faster information gathering. However, if robots collecting information are susceptible to data corruptions, not taking risk into account will lead to inferior data collection performances. This is especially true for robot swarms operating in poorly connected environments, where directly sending the acquired data items to permanent external storage is not always possible. The robots often need to store the information locally and the robotic system acts as a temporary storage infrastructure. Assuming that environmental hazards might compromise some of the robots of the system, determining which ones are the best-suited for storing information becomes essential for reducing the likelihood of data losses. This intuition is presented in Figure 1.3. In figure 1.3, robots are dispatched in an unknown environment with the task of gathering information about it. Because the environment contains dangers, in our case a point radia-

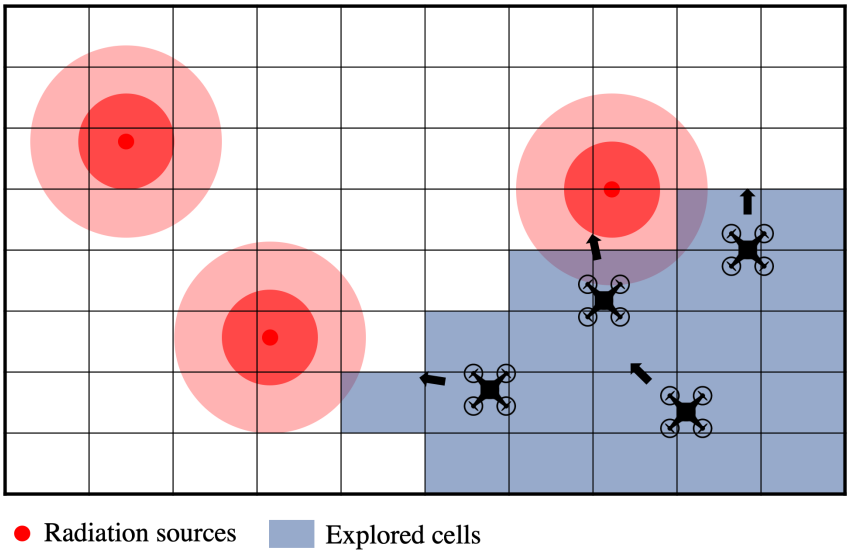


Figure 1.1 Exploration of an unknown environment with a team of robots

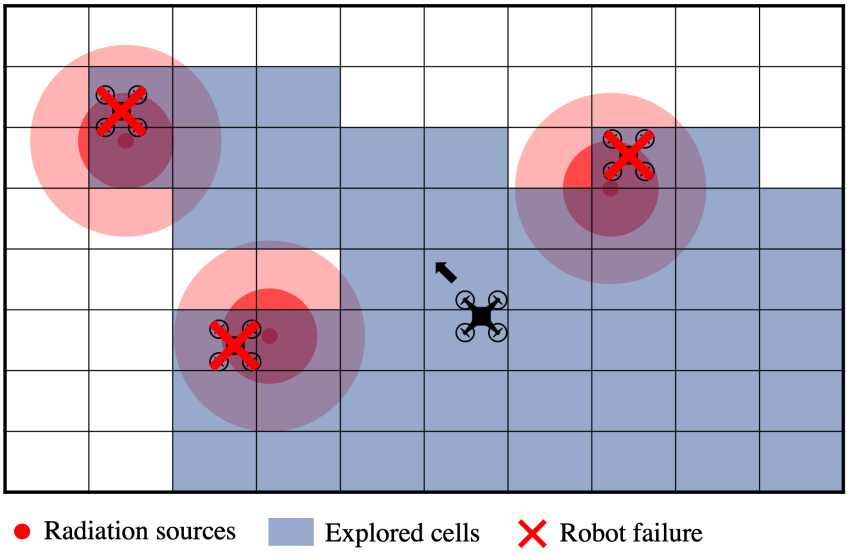


Figure 1.2 Exploration of an unknown environment with a team of robots affected by failures.

tion source, data items collected by robots may become corrupted if exposed to it. Transiting collected data items through robots located near a radiation source will therefore increase the likelihood of corruption. In figure 1.3, the lower route, although more direct towards the base station, is definitely more dangerous than the upper one. To reduce the likelihood of data losses, robots could use the upper route to reach the base station. This is the problem statement of RASS, the second algorithm of the thesis. RASS solves the problem by avoiding risky nodes of the system in its storage and routing scheme.

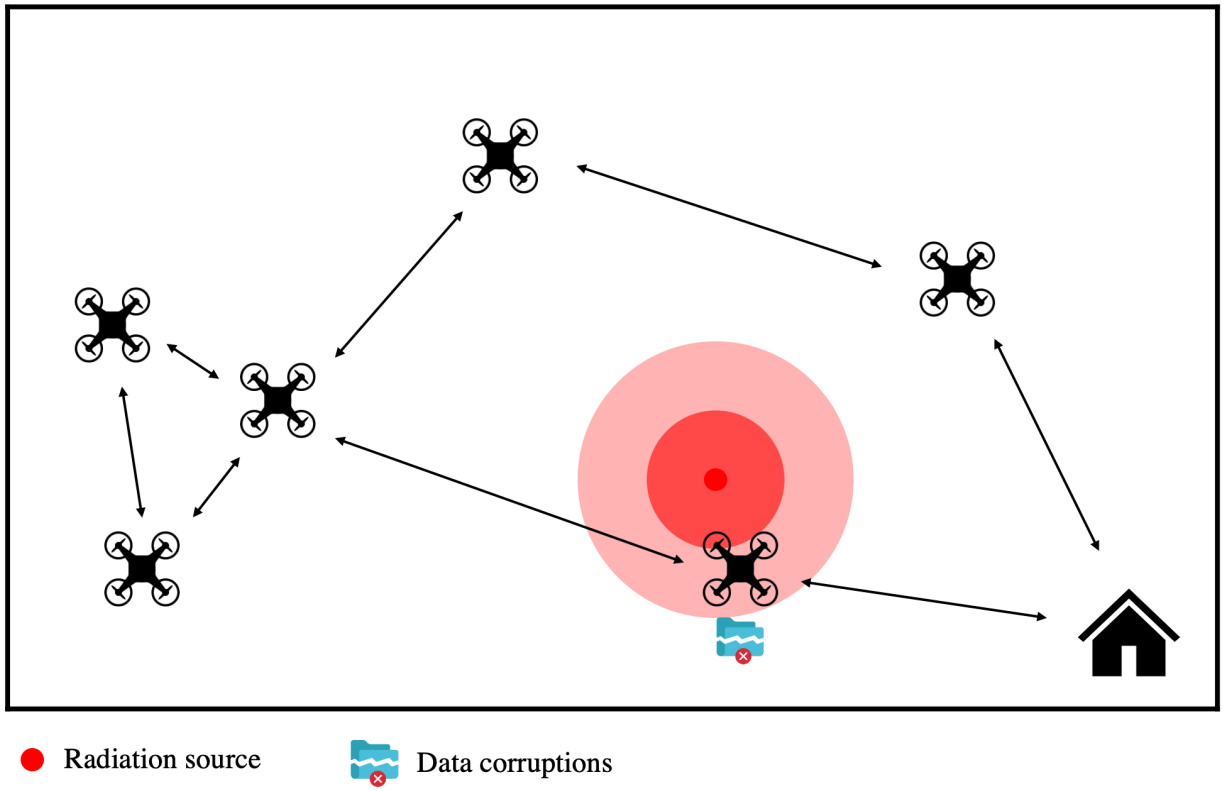


Figure 1.3 Information gathering in an hazardous environment

1.3 Research Objectives

Developing robot swarm algorithms to perform exploration missions in potentially dangerous environments has been the primary research objective of this master's degree. As mentioned in the previous section, this is of great importance as failures resulting from excessive exposure to risk will inevitably reduce the performance of the swarm.

1.3.1 Risk-aware Coverage

In this regard, we first focus on bringing risk awareness to a terrain coverage algorithm, something that was lacking in the literature. Explicitly, the developed coverage algorithm should reduce considerably the failure rate of the robots carrying the exploration mission when compared to other state-of-the-art algorithms. Of course, the proposed solution should also achieve comparable terrain coverage performances when compared to these same state-of-the-art algorithms. Because the developed algorithm is intended for robot swarms, it also needs to respect the swarm robotics design principles seen in the previous section. With these objectives in mind, the algorithm should address the problem elements specific to terrain coverage in dangerous environments from section 1.2.1. To summarize, the research objectives related to terrain coverage in dangerous environments are listed below:

1. Reduce failure rate compared to other state-of-the-art algorithms;
2. Achieve comparable terrain coverage compared to other state-of-the-art algorithms;
3. Respect swarm robotics design principles;
4. Test real-world applicability with experiments on physical robots;

1.3.2 Risk-aware Information Gathering

Second, we focus on risk awareness to the specific problem of information gathering performed in potentially dangerous environments, as seen in section 1.2.2. The related research objectives are the following: The proposed solution should reduce notably the rate of data items that are lost due to the environmental dangers. Also, the developed solution should provide adequate transfer speeds, defined as the time it takes for data items to reach the base station after creation. Again, it will need to respect the swarm robotics design principles as it is intended to be deploy on robot swarms. To summarize, the research objectives related to information gathering in dangerous environments are listed below:

1. Reduce data corruption rate compared to other state-of-the-art algorithms;
2. Achieve comparable transfer speed compared to other state-of-the-art algorithms;
3. Respect swarm robotics design principles;
4. Test real-world applicability with experiments on physical robots;

1.4 Thesis outline

The remainder of the thesis is structured as follows. In chapter 2, a literature review of relevant related works is presented. The following topics will be studied in the literature review: swarm robotics, swarm programming, information sharing, routing mechanisms, swarm exploration strategies, risk in swarm robotics and some notable fault detection methods. Following the literature review, in chapter 3, the scientific approach used to meet the research objectives will be presented. Then, in chapter 4 the algorithm DORA-Explorer, which uses risk awareness in its exploration strategy, will be presented. Subsequently, chapter 5 will cover RASS, a risk-aware storing and routing mechanism designed for robot swarms carrying exploration missions. A high-level discussion on the results obtained throughout the master's degree in relation with the research objectives will follow in chapter 6. Finally, in chapter 7, a conclusion is presented alongside the limitations of the work and interesting future research directions.

CHAPTER 2 LITERATURE REVIEW

The following chapter is an overview of the existing works found in the literature in relation with the research fields covered in the thesis. Attention will be brought to swarm programming languages and tools, information sharing in robot swarms, routing mechanisms, swarm exploration strategies, risk in swarm robotics and finally fault detection methods.

2.1 Swarm Programming

Drona [5] is a state-machine-based language providing decentralized motion planning for mobile drones. It can be run on the Robot Operating System (ROS) [6] and offers real-time collision-free planning even with not perfectly synchronized clocks between robots. This is particularly useful as perfect clock synchronization is hard to achieve, especially in large and dynamic networks [7]. However, Drona does not consider heterogeneous robot swarms.

Koord [8], a new programming language for distributed robotic applications draws considerable attention to verification and validation of the distributed algorithm. Koord provides abstraction from the physical robot and enables easy verification of the code. Its modularity and hardware-independence means that each part of the algorithm can be easily tested and validated.

A programming language, specifically designed for large-scale robot swarms, has been proposed in [9]. Buzz is an extensible programming language for heterogeneous robot swarms offering means of easily defining swarm behaviours both from a bottom-up and top-down perspective. Of course, Buzz respects the design principles of swarm robotic systems and self-organization is assured by the fully distributed run-time platform of the language. In addition to being an accessible swarm robotics programming tool, Buzz contains a large variety of the most common swarm behaviors, such as flocking, shape formation or barriers. It therefore enables fast prototyping and reduces the need for reprogramming these common swarm behaviors. The swarm-oriented programming language has also been integrated to ROS [10].

ARGoS [11] is a physics-based multi-robot simulator designed for large-scale experiments of heterogeneous swarms. ARGoS is time efficient thanks to its capability of running robots on separate threads. Experimental results demonstrate that simulation run-time increases linearly with the number of robots. The simulator is highly customizable where the simulated environment can be divided into subspaces with different physics engines. Moreover, Buzz

and ARGoS can work together particularly well. Swarm behaviors implemented using the Buzz programming language can be easily tested in the ARGoS simulator, even with large robot swarms. Because of the physics incorporated in the simulator, the behaviors displayed in the ARGoS usually translate well to real-world scenarios. The combination of Buzz and ARGoS provides an accessible and fast swarm robotics development tool.

2.2 Information Sharing

Distributed sensing and information sharing is not trivial, especially considering the challenges of consistency and partial connectivity among the robotic team [12, 13]. In social insects, pheromone trails are used to build a shared memory structure in the environment. For example, ants that communicate between each other by laying pheromones on the ground to guide peers towards food [14]. Such distributed memory mechanism is called stigmergy [15,16] and allows agents to interact with each other without the need for direct communication and centralized control.

The virtual stigmergy presented in [17] and implemented in the Buzz programming language [9] achieves consensus among a group of robots using Conflict-free Replicated Data Types (CRDTs), represented as key-value pairs shared and replicated among the swarm members. This sort of shared data structure is particularly relevant for belief maps, since it is easy to assign a unique key to each cell based on its location. In the virtual stigmergy, data is shared on writing and reading the CRDT, with the additional updates on read improving the robustness to temporary disconnections and message drops. This solution differs from distributed hash tables, which require a complete view of the system at every point in time. Essentially, information updates are propagated throughout the swarm using the stigmergy whenever it is possible. In this sense, it offers high availability while settling for eventual data consistency. However, since the information is fully replicated among the agents of the system, storing large data items can prove challenging. SOUL [18], a file sharing protocol addresses the problem by storing information in the form of (key, blob) pairs. The blob's metadata is fully replicated across the swarm, but the blobs are decomposed into datagrams stored on specific nodes of the system. Other distributed data storage approaches such as SwarmMesh [19] store data in different locations based on a fitness function instead of replicating them on all robots. This allows the storage of more data with less communication, but robots are less likely to have access to the latest values.

Belief maps are a simple yet powerful tool for robotic exploration. They render a continuous surface into a discrete set of cells which is especially useful for engineering problems. It makes possible highlighting specific regions of the environment and allows the designer of

the algorithm to tune the level of precision at which the environment is represented. A finer granularity will provide a more accurate representation of the environment and as a result should offer more precision. However, finer granularity usually results in higher computational costs. The cells of the belief map are used to store information about the corresponding surface they represent. They generally store a probability that indicates the confidence level to which the cell is believed to contain a specific feature. Belief maps are a generalization of occupancy maps: instead of storing only one bit per cell to indicate the presence of an obstacle/danger, they store obstacle/danger likelihood and offer significant improvements for exploration [20]. In the field of multi-robot exploration, early techniques leveraging belief maps date back as far as twenty years [21, 22], but they rely on a fixed grid size and are tested only with two robots. More recent works also leverage belief maps for multi-robot exploration. For example, in [23], the robots consider both the current beliefs and the expected beliefs from future observations to coordinate their exploration. Grid maps and belief maps are also widely used to train deep reinforcement learning exploration policies [24, 25], but they often rely on a trial and error process which may select actions leading to failures [26, 27].

2.3 Routing

Data storage and routing remains a challenge for robot swarms as the amount of collected data items only increases with the number of robots in the system. Because robot swarms typically suffer from bad connectivity [12], sending collected data items directly to external storage might not always be possible. Robots often need to store locally the data items until a path towards permanent storage becomes available. Because robots usually have limited communication range, the data items collected during the mission may need to be routed through multiple robots before reaching the external storage infrastructure. The multi-robot system becomes a temporary storage infrastructure and deciding where to store and send the data items become essential. One of the most popular approaches for routing data items in swarm robotics is to use a gradient-based routing scheme [28–30]. In gradient-based routing, a scalar value (also called height) is assigned to each of the nodes of the system. The value is based on a metric that evaluates how fit this node is to be used in the routing scheme. For example, in a simple hop count based routing protocol, the scalar value would increase with the distance with the base station. Then, the nodes through which data items are routed are simply chosen by using the neighbor that displays the lowest height. Probably the most used metric for defining the height of the nodes is hop count [31–33] as it allows fast and efficient routing of data items. However, other cost functions for determining the height of the nodes

are possible.

Recently, many routing strategies have been developed for vehicular ad hoc networks (VANETs). In [34], a survey on routing protocols for such networks is presented. The survey focuses on routing protocols that use several metrics as they have shown to be effective in dynamic networks. Specifically, protocols based on the nodes' geographic positions are the most adequate as the dynamicity of the network is directly included in the routing scheme [35, 36].

Other notable routing mechanisms take inspiration from nature. In [37, 38], path growth routing protocols based on slime molds are presented. In [39, 40], ant colony optimization-based routing algorithms are presented. The ant colony optimization algorithm tries to reproduce the behavior of ants in their search of the nearest food sources [41]. In the process, ants lay pheromones on the ground. Hence, the shorter the path between the nest and the food source, the more frequent it will be traversed by ants, thus increasing the pheromone intensity. Ants then follow paths with high pheromone intensities. In [40] existing approaches are investigated and a hybrid routing mechanism that combines ant colony optimization and hop count into one routing scheme is proposed. However, these approaches are tailored for static topologies and are ill suited for dynamic robotic networks.

2.4 Swarm Exploration Strategies

Many distributed exploration strategies maximizing the amount of terrain coverage have been proposed. The first approaches to stand out in this regard are Voronoi-based coverage control techniques [42, 43]. Specific to [43], connectivity awareness is included in the Voronoi-based coverage to guarantee that a minimum of connectivity is maintained between the robots of the system.

A second method covers time-varying domains, in which points in the covered region can become more or less interesting to explore, therefore prompting a change in the coverage function [44, 45].

Another method to optimize coverage is Frontier-Based Exploration (FBE) [46] of which many variations have been developed, such as those based on Particle Swarm Optimization [47] or the Wavefront Frontier Detector [48]. FBE's key principle is to assign one of three states to the cells constituting the environment:

- **Explored:** cells that have already been explored
- **Frontier:** cells between explored and unexplored space
- **Unexplored:** cells that remain uncovered

Using the state of the cells, robots coordinate to explore the regions at the frontier, thereby expanding them and eventually achieving full map coverage.

Other interesting coverage techniques take inspiration from insects and leverage virtual pheromones for encouraging or discouraging robots to explore specific regions of the environment. For example, in ant foraging, negative feedback in the form of pheromones is used to discourage ants from using unprofitable paths [49]. In [50], pheromones are employed to dissuade robots of covering already covered areas of the environment. Authors show that using repellent pheromones can effectively encourage a rapid dispersal of robots even in large swarm systems. In Phormica [51], robots start exploring the environment in a random fashion. By projecting Ultra Violet (UV) light on the ground, the robots are able to convey information to their peers and let them know that the region has already been covered. When the artificial pheromone is detected, robots can change their course of action and move towards unaltered regions of the environment.

However, none of these aforementioned coverage strategies take risk into account. In [52, 53] an exploration algorithm that maximizes information gain in the presence of unknown hazards is presented. Unfortunately, this optimal algorithm has a very high computational complexity and could benefit from approximations.

2.5 Risk in Swarm Robotics

The importance of risk management has dramatically increased over the last few years. Robotic systems are being used more widely and as a result, risk management becomes essential as to not endanger the system itself and the objects and beings in its immediate environment. The importance of enabling distributed situational awareness in robot swarms is raised in [54]. In [55], a checklist for robot swarms to be safe for the public, the environment and for itself is presented. The authors draw attention to the lack of proper systematic swarm safety assessment mechanism and propose a checklist, in the form of ten questions, that should be answered when designing such systems. They consider that swarm safety is larger than simply analyzing failure modes. Questions related to ethics, legality, accountability, security are also listed and as a result, should provide a more thorough consideration of all socio-technical risks robot swarms face.

In [56], the security challenges of robots swarms are presented. Security is defined as the state of being protected from risks originating from hostile and malicious intentions. The security of new technologies is usually not included in its design process, typically it is with the rise of the technology that security concerns appear. To prevent any unwanted consequences, the

paper presents the security challenges robot swarms will face in an attempt of including this component in the design process of the new technology. However, only security threats are considered and risk, as something not malicious, is not examined.

Several path planners based on Markov Decision Processes [57–59] take into account risk and have useful definitions of it. In [59], risk is categorized into three different groups:

- **Locale-Dependent:** Risk elements not depending on history. The risks associated with this category are usually location-based, in other words, it is the position of the robot in the environment that determines the level of risk.
- **Action-Dependent:** Risk elements depending on close history, specifically changes of states. For example, the risk associated with an aggressive turn is tied to the last states of the robot.
- **Traverse-Dependent:** Risk elements depending on the entire history of the robot. Risks associated with this category are tied to all the states traversed by the robot. For example, risk associated with low battery levels are included in the category.

In [60], a risk-aware motion planning and decision-making mechanism is presented. They automatically adjust the "conservativeness" of the motion-planner based on the risk a robot faces. The paper uses a conditional value-at-risk method used in finance to estimate the risk of an investment. They use a safety risk measure from [61] onto which they apply the conditional value-at-risk method to achieve safe motion planning and control. Unfortunately, risk only includes collisions and does not translate to other types of hazard robots could face. Value-at-risk strategies for robot swarms in hazardous environments were also studied in [62]. Again, the method of value-at-risk allows quantifying the foreseen loses over a period of time where the environment contains potentially damaging radiation sources. In detail, agents of the swarm calculate the value-at-risk at every time step and share it with their neighbors when their value-at-risk limit is exceeded. The information helps team members avoid dangerous locations of the environment and overall decreases exposure to risk. Some shortcomings of the method include the determination of the value-at-risk limit and the lack of instantaneous responsiveness of the method as it relies on past observations. Overall, the value-at-risk methods show that financial risk management techniques are an interesting avenue for risk awareness in swarm robotics.

Another interesting idea for risk awareness is proposed in [63, 64], where a "risk budget" is allocated to their agents, allowing them to optimize a balance between risk and reward to

guide robots. However, these systems assume knowledge of the global state of the environment, which is unavailable when exploring unknown environments. Furthermore, most are only applied to single-robot systems.

In SPIDER [65], multiple agents are tasked with chain formation in dangerous environments. They adapt to varying levels of risk to be resilient to significant failures and member losses in order to maximize information gathering. They introduce a level of "boldness" which represent the risk appetite of the agents of the swarm. This risk appetite is modulated by the connectivity of the agent, specifically its frequency of interactions of neighbors. When an agent is well connected, its risk appetite grows and, as a result, should be encouraged to explore new parts of the environment. On the other hand, an agent that is isolated and far from any other members of the swarm will increasingly display a shy behavior and go back to safer areas of the environment. SPIDER effectively allows a swarm to trade off the benefits of information gain versus robot failures. However, the problem is only studied for a chain formation scenario and risk is only a measure of how well connected is an agent.

2.6 Fault Detection

A taxonomy of the faults affecting robotic systems is presented in [66]. They compromise the sensing and acting abilities of the robots. Understanding faults and exploring the methods to detect them is of interest as they can be the result of an excessive exposure to risk. Although the algorithms developed in this thesis reduce considerably the likelihood of failures, they do not guarantee fault-free operation. Therefore, rapidly detecting faults is still of importance and could be added as an additional layer to the robotic system for increased tolerance to risk. Software faults affect the behaviour of the robots and are caused by faulty algorithms and/or faulty implementations. Interaction faults affect the dynamics of the robotic system and are caused by exogenous events. Hopefully, fault detection methods have been developed to mitigate the presence of faults in robotic systems. They are divided into three big families: data-driven; model-based; knowledge-based [66].

- **Data-driven** approaches use sensor data and compare it to known faults, to past normal/abnormal behaviors or to the behavior of neighbour agents. Using statistical tools, data-driven approaches compute the deviation of the data and classify it as normal/abnormal depending on the extent of this deviation.
- **Model-based** approaches use an *a priori* explicit model of the system to identify faults. The model is a set of analytical equations or logical formulas. When an irregularity is identified between what is observed and the theoretical model, a fault is presumed. The

main drawback of these approaches is the work needed in constructing the theoretical model.

- **Knowledge-based** approaches are similar to the way a human would perform fault detection. A fault is quickly associated with its cause. Faults are typically represented in a tree structure where the fault can be linked backwards to where it originated. It is particularly useful for fault isolation.

Outlier detection methodologies have been widely used for identifying anomalies that could be the result of a fault. A definition of "outlier" was proposed by Grubbs in [67]:

An outlying observation, or outlier, is one that appears to deviate markedly from other members of the sample in which it occurs.

Outlier detection is used for classifying data as normal or abnormal. Applying it to fault detection is to suppose that the abnormality observed is the result of a fault. Outlier detection is part of the data-driven family of the fault detection approaches. From [67], outlier detection can be divided into 3 types:

- **Unsupervised clustering:** The outlier detection does not need any prior knowledge of the data. Using the distribution of the data, the most isolated points are classified as outliers.
- **Supervised classification:** The outlier detection needs a prior knowledge of normality and abnormality. Using pre-labelled normal/abnormal data, new incoming data can be classified in either of these classes based on its distance.
- **Semi-supervised recognition:** The outlier detection needs prior knowledge of normality. Using pre-labelled normal data, new incoming data can be classified as normal or abnormal. It resembles type 2 outlier detection but without the need for labelled abnormal data.

Additionally, two strategies can be used for fault detection: endogenous fault detection and exogenous fault detection [68–70]. Endogenous fault detection refers to the actions taken by an individual to perform fault detection on itself and on its own. While this approach is well suited for single robots, in the context of multi-robot systems, opting for this approach does not take advantage of the multiple entities close to one another forming the swarm. On the other hand, exogenous fault detection refers to the actions taken by neighbour robots on a

central entity. This approach relies on the observations of multiple robots and is, as a result, better suited for swarm robotics. Exogenous data-driven fault detection methods are the best suited for robot swarms. Exogenous strategies leverage the power of swarms through collaboration of all team members. Data-driven methods allows the detection of previously unseen faults and, as a result, enable the deployment of the swarm in unknown and dynamic environments.

A fault detection method inspired by the human immune system has been proposed in [71,72]. The method uses the mathematical formulation of the cross-regulation model to distinguish between normal and abnormal behaviors in a swarm of robots. Abnormality is the indication of a fault, the abnormality being the result of it. The fault detection method is divided into three main phases: (i) observing the behaviors of the agents; (ii) based on the observed behaviors, perform anomaly detection using the cross-regulation model; (iii) voting to decide if the agent's behavior is normal or abnormal. The first phase of the method (i) uses feature vectors to characterize the behavior of an agent. The feature vector contains six features $(F_1, F_2, F_3, F_4, F_5, F_6)$, each indicating the presence $(Fi = 1)$ or absence $(Fi = 0)$ of a particular behavior. Then, the method uses these informative binary feature vectors for running the cross-regulation model (ii). In this fault detection method, if at the end of the cross-regulation cycle, the Antigen Presenting Cell (APC) (modelled as a feature vector) is considered a foreign pathogen, an abnormality is inferred. Finally, in the last phase of the methods (iii) the agents consolidate their individual decisions and vote on the normal/faulty behavior of the neighboring agents. Agents that receive more than five "foreign pathogen" votes are considered faulty. The method was tested on a swarm of seven physical robots and results showed that the method was able to reliably identify the faults injected in the system [73]. The efficiency of the method actually surpassed the ones of other traditional outlier detection methods, namely K-Nearest Neighbor (KNN) and Local Outlier Factor (LOF). However, for both KNN and LOF binary feature vectors were used, however, these methods could benefit from using non-binary feature vectors [73]. The advantage of this method is that it does not require any prior knowledge of the fault to identify it. Indeed, being an outlier detection method, the presence of faults is only inferred on the presence of an abnormal behavior. This is particularly important in swarm of robots where it is very difficult to know in advance the potential fault that the system will encounter. It is also easier to implement. However, because the method identifies anomalies and not faults, it is very hard to perform diagnosis and recovery procedures. It is difficult to assess the correct plan of action when the nature of the problem remains unknown. In [74], the problem is solved by introducing a fault diagnosis layer after the fault detection one.

The LOF [75] is a measure of how much a data point can be considered an outlier. The

LOF represents the density of a point compared to the density of its neighbors. A LOF of around 1 means that the point is not an outlier. A LOF much higher than 1 indicates that the density of the data point is smaller the density of the neighboring points. The data point is isolated and is, as a result, most likely to be an outlier. In summary, the higher the LOF, the most likely the data point is an outlier.

A neural network fault detection approach was proposed in [76]. The approach assumes that the occurrence of a fault in the robotic system will cause the flow of sensory data to change. By monitoring it and feeding it to a neural network, the method is capable of detecting faults in the system with manageable latency using s-bots robots [77]. For every control cycle, sensory data is given as an input to the neural network which, in turn, outputs the state of the system (0 or 1), with 0 corresponding to not faulty and 1 to faulty. Again, this falls into the anomaly detection category and more specifically under the data-driven group. One of the drawback of the method is the need for training runs to train the neural network.

In recent years, deep learning anomaly detection methods have been developed and have shown to generally outperform the traditional anomaly detection methods [78]. They are especially effective when working with complex and large data flows where traditional methods tend to struggle.

CHAPTER 3 RESEARCH APPROACH AND THESIS ORGANIZATION

This chapter will present the research approach used to meet the objectives detailed in 1.3. The link between the two works DORA-Explorer and RASS will follow and finally, the document structure will be presented.

3.1 Research approach

Bringing risk-awareness and increasing robustness of swarm algorithms has been the primary focus of the master's work. Because the laboratory in which the master's degree was carried, the MIST Lab, focuses on space technologies, the approach chosen to meet the identified objectives was to develop meaningful algorithms for the exploration of space. In this regard, an exploration algorithm, DORA-Explorer, was built and enables efficient coverage of an unknown environment while avoiding its hazardous locations. Then, with a satisfactory exploration algorithm, focus switched towards building an efficient storing and routing algorithm: RASS. This algorithm is meant to be used by a robot swarm carrying the coverage task and collecting information in the process. The two algorithms are linked by their contribution to space exploration in the presence of risk. They are meant to be used side by side when exploring and gathering information about a new environment and should provide increased robustness to the swarm carrying the mission.

3.2 Document structure

The document's structure follows the one prescribed for a thesis with articles. Because I am first author for the article on DORA-Explorer, this one will be directly included in the body of the thesis under its original article format whereas the second work, RASS, will be summarized. At the end of the thesis, appendices in relation with DORA-Explorer are presented. In appendices A, B, C, D, E will be respectively presented the execution loop of DORA-Explorer, the intuition of the algorithm, the ARGoS environment from the simulations, some additional results and finally, a picture of the physical experiments carried in the lab. The document is structured as follows:

- Chapter 1 gives an introduction on the research subject and presents the basic concepts upon which the work done in the course of the master's degree has been built.
- Chapter 2 provides relevant contributions to the master's thesis found in the literature.

- Chapter 3 presents the common thread between the two research projects carried in the course of the master's degree
- Chapter 4 presents a fully decentralized and risk-aware exploration algorithm called DORA-Explorer. This work was published in the IEEE International Conference on Robotics and Automation (ICRA) in May 2022.
- Chapter 5 presents a fully decentralized and risk-aware routing algorithm called RASS. This work was published at the International Conference on Autonomous Agents and Multiagent Systems (AAMAS) as an extended abstract in May 2022 and was presented in the Autonomous Robots and Multirobot Systems (ARMS) workshop.
- Chapter 6 presents the results obtained by the two algorithms and to what extent they satisfied the objectives of the master's degree.
- Chapter 7 provides a summary of the works as well as some limitations of the algorithms presented and some interesting future research directions.

CHAPTER 4 ARTICLE 1: DORA: DISTRIBUTED ONLINE RISK-AWARE EXPLORER

Preface: This chapter presents a risk-aware coverage algorithm that maximizes the amount of terrain covered by the robots while minimizing the risk to which robots expose themselves in the process. The algorithm leverages distributed belief maps to share across the swarm the risk belief and the information gain belief associated with areas of the environment. Using a control policy based on risk and information gain, a movement vector that encompasses both objectives is computed locally and provides good short term trajectory planning to the robots. The approach is validated through exhaustive simulations and its real-world applicability is verified through physical experiments.

Full Citation: D Vielfaure, S Arseneault, P-Y Lajoie, G Beltrame. "DORA: Distributed Online Risk-Aware Explorer". 23rd of May 2022. IEEE International Conference on Robotics and Automation (ICRA).

4.1 Abstract

Exploration of unknown environments is an important challenge in the field of robotics. While a single robot can achieve this task alone, evidence suggests it could be accomplished more efficiently by groups of robots, with advantages in terms of terrain coverage as well as robustness to failures. Exploration can be guided through belief maps, which provide probabilistic information about which part of the terrain is interesting to explore (either based on risk management or reward). This process can be centrally coordinated by building a collective belief map on a common server. However, relying on a central processing station creates a communication bottleneck and single point of failure for the system. In this paper, we present Distributed Online Risk-Aware (DORA) Explorer, an exploration system that leverages decentralized information sharing to update a common risk belief map. DORA-Explorer allows a group of robots to explore an unknown environment discretized as a 2D grid with obstacles, with high coverage while minimizing exposure to risk, effectively reducing robot failures.

4.2 Introduction

The exploration of unknown environments is at the core of numerous robotic applications from search-and-rescue operations [79] to space missions [80]. The problem has been mostly studied in single robot setups, but the ability to perform exploration with teams of robots opens the door to even more ambitious applications, because with proper coordination, the time required to explore a given environment should decrease proportionally to the number of robots [1]. Therefore, multi-robot exploration is an attractive solution to many time-critical applications such as search-and-rescue operations or planetary exploration. Moreover, multi-robot teams are usually resilient to some amount of robot failures [81–83]. However, robot failures are still undesirable as they can affect team performance and should therefore be avoided, which is the main motivation for this work, in which we present a risk-aware exploration algorithm for multi-robot systems: Distributed Online Risk-Aware (DORA) Explorer.

Multi-robot systems come with their own sets of constraints and challenges: among those, coordination and communication are the most relevant to the exploration problem. Without coordination, the robots will inevitably explore overlapping parts of the environment, leading to little gains in terms of efficiency compared to single-robot solutions. While the coordination could be optimally orchestrated from a central computing station, such a solution would require a perfect connectivity maintenance with each robot and a high communication bandwidth since the robots would need to send their observations and receive their commands. This motivates the need for a decentralized exploration algorithm relying only on local computation onboard the robots and communication with their neighbours. To the best of our knowledge, there exists no risk-aware collaborative exploration algorithm that relies solely on local or shared information. Therefore, in this paper, we make the following contribution to the field of multi-robot exploration: *A decentralized exploration algorithm leveraging distributed belief maps (DBMs) to maximize coverage and decrease robot failure probability using risk-awareness.* To evaluate this system, we test it on the specific problem of *hazard mapping* in a 2D world discretized as a grid, in which a multi-robot team simultaneously explores a dangerous environment and collaborates to avoid hazardous locations as well as obstacles. We validate our approach in a physics-based simulator, ARGoS [11], in which we define a grid-based environment with multiple radiation sources. We then test it on physical robots and obtain convincing results.

4.3 Related Work and Background

Distributed sensing and information sharing is not trivial, especially considering the challenges of consistency and partial connectivity among the robotic teams [12, 13]. The virtual stigmergy presented in [17] and implemented in the Buzz programming language [9] achieves consensus among a group of robots using Conflict-free Replicated Data Types (CRDTs), represented as key-value pairs shared and replicated among the swarm members. This sort of shared data structure is particularly relevant for belief maps, since it is easy to assign a unique key to each cell based on its location. In the virtual stigmergy, data is shared on writing and reading the CRDT, with the additional updates on read improving the robustness to temporary disconnections and message drops. This solution differs from distributed hash tables, which require a complete view of the system at every point in time. Essentially, information updates are propagated throughout the swarm using the stigmergy whenever it is possible. In this sense, it offers high availability while settling for eventual data consistency. Other distributed data storage approaches such as SwarmMesh [19] store data in different locations based on a fitness function instead of replicating them on all robots. This allows the storage of more data with less communication, but robots are less likely to have access to the latest values.

Belief maps are a simple yet powerful tool for robotic exploration because they can represent an environment with a 2D cell grid. They are a generalization of occupancy maps: instead of storing only one bit per cell to indicate the presence of an obstacle/danger, they store obstacle/danger likelihoods and offer significant improvements for exploration [20]. In the field of multi-robot exploration, early techniques leveraging belief maps date back as far as twenty years [21, 22], but they rely on a fixed grid size and are tested only with two robots. More recent works also leverage belief maps for multi-robot exploration. For example, in [23], the robots consider both the current beliefs and the expected beliefs from future observations to coordinate their exploration. Grid maps and belief maps are also widely used to train deep reinforcement learning exploration policies [24, 25], but they often rely on a trial and error process which may select actions leading to failures [26, 27].

To provide more insight into our claim that there are no risk-aware collaborative exploration strategies in the literature, it should be noted that risk-awareness has indeed been used in some swarm systems to improve robustness. For example, several path planners based on Markov Decision Processes [57–59] take into account risk and have useful definitions of it. Another interesting idea for risk awareness is proposed in [63, 64], where a "risk budget" is allocated to their agents, allowing them to optimize a balance between risk and reward to guide robots. However, these systems assume a knowledge of the global state of the

environment, which is unavailable when exploring unknown environments. Furthermore, most are only applied to single-robot systems. In SPIDER [65], multiple agents are tasked with chain formation in dangerous environments. They adapt to varying levels of risk to be resilient to significant failures and member losses. However, their task is significantly different from DORA Explorer's objectives.

Many distributed exploration strategies maximizing the amount of covered terrain have been proposed. The first approaches to stand out in this regard are Voronoi-based coverage control techniques [43, 44]. A second method covers time-varying domains, in which points in the covered region can become more or less interesting to explore, therefore prompting a change in the coverage function [44, 45]. Another method to optimize coverage is Frontier-Based Exploration (FBE) [46] of which many variations have been developed, such as those based on Particle Swarm Optimization [47] or the Wavefront Frontier Detector [48]. However, none of these strategies take risk into account. Therefore, the exploration strategy implemented in this paper takes inspiration of the multi-robot control algorithm presented in [52, 53] which maximizes the information gain during exploration in the presence of unknown hazards. However, this optimal algorithm has a very high computational complexity and could benefit from approximations.

4.4 System Model

DORA-Explorer builds on the previously mentioned approaches and addresses some of their shortcomings, namely by leveraging risk awareness to provide better efficiency when exploring hazardous environments. Reducing the likelihood of robot failures is of high importance as they lead to poor exploration performance. Indeed, if robots experiencing complete failures are not replaced, individual failures lead to lower numbers of robots carrying the exploration task resulting in a decrease of the exploration rate. We model the 2D environment as cells forming a grid represented as $E \subset \mathbb{Z}^2$. The team of robots is denoted as the collection of agents $a_i \in A$.

4.4.1 Risk Modelling

Without loss of generality, we model risk considering point radiation sources, denoted by the set S. The intensity of each radiation source is given by $I_j \sim \mathcal{U}(0, 1)$. Each source's position is denoted by $\boldsymbol{s}_j \in E$. Given a robot a_i's discrete position $\boldsymbol{x}_i \in E$, the perceived radiation level by that robot coming from radiation source $\boldsymbol{s}_j$ is given by:

$$r_{\boldsymbol{s}_j}(\boldsymbol{x}_i) = \frac{I_j}{1 + \lambda\rho^2} \tag{4.1}$$

which decays as the distance ρ between $\boldsymbol{s}_j$ and $\boldsymbol{x}_i$ increases, and λ is a decay constant. Measurement noise is accounted for in the form of a Gaussian background radiation $b \sim \mathcal{N}(0, 0.05)$. The total radiation perceived by a robot is:

$$r(\boldsymbol{x}_i) = b + \sum_{\boldsymbol{s}_j \in S} r_{\boldsymbol{s}_j}(\boldsymbol{x}_i) \tag{4.2}$$

Robots are only able to sense the radiation level associated with their current position using an onboard sensor. They do not hold any knowledge of where the radiations sources are located in the environment. For the following definitions, it should be noted that $r_{s_j} : E \to [0, 1]$. Let the event of robot a_i failing be $f_i = 1$, the probability of such a failure due to an individual source of radiation follows a Bernoulli distribution: $\mathbb{P}(f_i = 1|\boldsymbol{s}_j) \sim \mathcal{B}(r_{\boldsymbol{s}_j}(\boldsymbol{x}_i))$. We assume that the sources of radiation affect the robots independently, consequently the probability of a robot failing due to the combined effect of all radiation sources is:

$$\mathbb{P}(f_i = 1|S) = 1 - \prod_{\boldsymbol{s_j} \in S} (1 - \mathbb{P}(f_i = 1|\boldsymbol{s}_j)) \tag{4.3}$$

4.4.2 Information Modelling

The objective of exploring an unknown dynamic environment is to gain information about it. Moreover, this information should be as up to date as possible. Therefore, it is unlikely that visiting a recently explored cell will yield any significant gain as the information should not have changed drastically. Conversely, exploring areas visited long ago should yield a greater information gain, and unvisited areas should provide the highest information gain. The last time of exploration t_ϵ by robot a_i of a cell at position $\boldsymbol{x}_i$ can be represented by the scalar field $\epsilon(\boldsymbol{x}_i) = t_\epsilon$. Let $u_i = 1$ be the event of robot a_i finding useful information in a cell and $\Delta t = t - t_\epsilon$ the time elapsed since the cell was last visited, with t being the current time. Then, the probability of *not* finding useful information $\mathbb{P}(u_i = 0|f_i = 0, \Delta t)$ can be modelled as an exponential distribution with the following probability density function:

$$f(\Delta t; \omega) = \omega e^{-\omega \Delta t} \tag{4.4}$$

where ω is the rate parameter of the distribution. In words, the longer the cell has not been visited, the higher the chance something has changed and consequently the lower the chance

of not finding useful information. Intuitively, no information can be acquired by failed robots.

4.4.3 Distributed Belief Map

To implement a DBM, we use the virtual stigmergy [17] from the Buzz [9] programming language. Because $r(\boldsymbol{x}_i)$ and $\epsilon(\boldsymbol{x}_i)$ are both scalar fields, they lend themselves particularly well to being stored in a CRDT at a low cost. At each time step, the robots store their values of $r(\boldsymbol{x}_i)$ and $\epsilon(\boldsymbol{x}_i)$ in their respective stigmergies. The inputs to both fields are used as keys in the distributed belief map (more precisely, a concatenation of $\boldsymbol{x}_{i;x}$ and $\boldsymbol{x}_{i;y}$). This means that the cost of storing the information for a given time step is very low, especially as the keys consist of a few characters and the values are floating point numbers. Storing the information into the DBM via the virtual stigmergy allows robots to share their observations as it is accessible by every robot in the system. Thus, a robot visiting a cell for the first time could still have information from which to compute a good control policy if this cell was previously visited by another robot. In the event of a collision in the stigmergy (when robots write to the same key in the same t), the robot making the latest observation updates the table. When a robot writes to a key already present in the stigmergy (from a previous time step), the new data is merged with an average and the result is propagated. A running average was used to update the belief map because of the noisy readings.

4.4.4 Control Law

We assume that robots can be controlled through a position-based control law. The best control policy should attempt to minimize probability of failure and to maximize information gain. While the directions achieving these individual objectives might be at odds in the short term, they are in fact complementary in the long term because no information can be gained if a robot failed, which means that avoiding danger implicitly leads to more opportunities of gaining information [53].

For a robot at a given position $\boldsymbol{x}_i$, the directions where the risk is minimized and the information gain is maximized are respectively $\boldsymbol{\nabla} r(\boldsymbol{x}_i)$ and $\boldsymbol{\nabla} \epsilon(\boldsymbol{x}_i)$, also denoted as $\boldsymbol{\nabla}_{r;i}$ and $\boldsymbol{\nabla}_{\epsilon;i}$. Calculating these globally at every time step is too computationally expensive [52, 53]. Instead, we compute them locally in a Moore neighborhood ν centered on $\boldsymbol{x}_i$ as shown in Fig. 4.1 where each neighboring cell $\boldsymbol{n}_{i,j} \in \nu$ is a vector in $\mathbb{Z}^2$ representing an offset from $\boldsymbol{x}_i$. We then have:

$$\boldsymbol{\nabla}_{r;i} = \sum_{\boldsymbol{n}_j \in \nu} \frac{\partial r}{\partial \boldsymbol{n}_{i,j}} \hat{\boldsymbol{n}}_{i,j} \text{ , with } \frac{\partial r}{\partial \boldsymbol{n}_{i,j}} = r(\boldsymbol{x}_i) - r(\boldsymbol{n}_{i,j}) \tag{4.5}$$

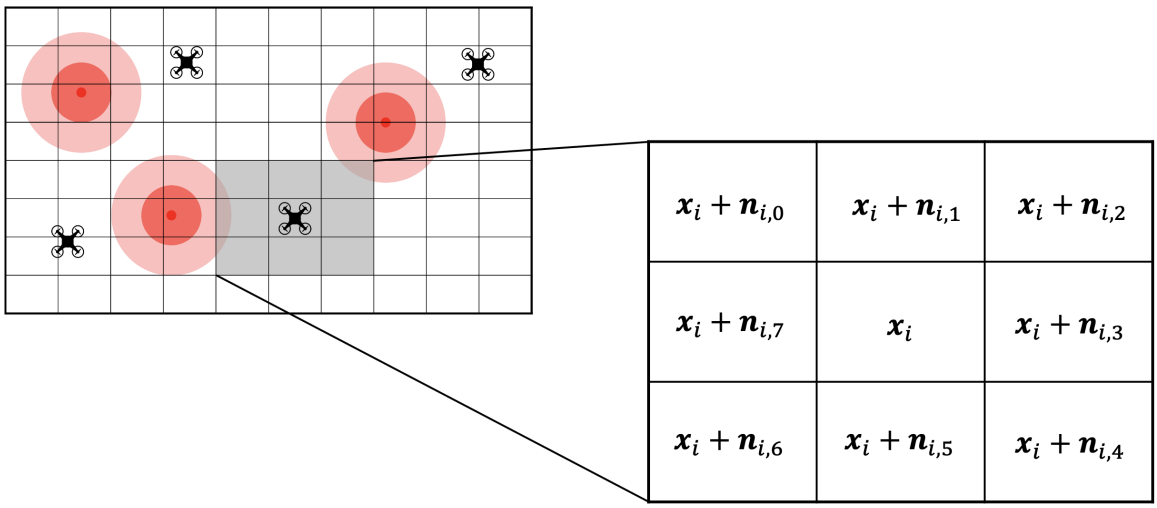


Figure 4.1 $\boldsymbol{x}_i$'s neighborhood. $\boldsymbol{n}_{i,0} = (-1, 1)$ is neighbor 0's offset from $\boldsymbol{x}_i$.

$$\boldsymbol{\nabla}_{\epsilon;i} = \sum_{\boldsymbol{n}_j \in \nu} \frac{\partial \epsilon}{\partial \boldsymbol{n}_{i,j}} \hat{\boldsymbol{n}}_{i,j} \text{ , with } \frac{\partial \epsilon}{\partial \boldsymbol{n}_{i,j}} = \epsilon(\boldsymbol{x}_i) - \epsilon(\boldsymbol{n}_{i,j}) \tag{4.6}$$

where $\hat{\boldsymbol{n}}$ is the unit form of $\boldsymbol{n}$. Neighboring cells that have never been visited before are simply ignored when determining the gradients. If none of the neighboring cells have been explored yet, or if the computed direction is null, the robot simply moves forward until it's able to compute a meaningful direction. The movement vector $\boldsymbol{m}_i \in \mathbb{R}^2$ for the next time step gives a good approximation for short term trajectory planning and is given by:

$$\boldsymbol{m}_i = \alpha \boldsymbol{\nabla}_{r;i} + \beta \boldsymbol{\nabla}_{\epsilon;i} + \gamma \boldsymbol{o}_i \tag{4.7}$$

where α, β, γ are respectively the risk avoidance, exploration and obstacle avoidance control gains. The parameters can be adjusted arbitrarily; setting them to zero removes the effect of the corresponding control law. The obstacle avoidance vector was included to insure robustness and is taken from [84]. With $\hat{\boldsymbol{m}}_{\boldsymbol{i}}$ being the normalized vector movement and k a speed constant, the control law for an agent a_i at time step t is expressed as:

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + k\hat{\boldsymbol{m}}_i^t \tag{4.8}$$

A constant search speed enables a fair comparison with the baselines detailed in section 4.5.1 as the exploration capabilities of the robots remain the same across the algorithms.

4.4.5 Scalability

To achieve scalability to a high number of agents and to large environments, DORA-Explorer must have low communication and computational costs. Lowering the communication costs associated with sharing the DBM can be done by using the virtual stigmergy, which is designed to limit information exchange to read or write operations only on the requested data. Because DORA-Explorer relies solely on local information, the data transfer cost $D(A, \nu, E)$ for an agent at a given time step is independent from the total number of agents in A and from the size of the environment E. For a neighboring cell $\boldsymbol{n}_{i,j} \in \nu$, 2 stigmergy read operations are needed per time step: one each to read $r(\boldsymbol{n}_{i,j})$ and $\epsilon(\boldsymbol{n}_{i,j})$. In the same time step, the agent updates $r(\boldsymbol{x}_i)$ and $\epsilon(\boldsymbol{x}_i)$ after it has moved to a new location, which requires 2 stigmergy write operations. Each stigmergy access requires only a few tens of bytes of data transfer for the key and value. This mostly constant data quantity is represented as d, we have $D(A, \nu, E) = 2d(|\nu| + 1)$. Similarly, the computational cost $C(A, \nu, E)$ for the same agent at the same time step is kept very low because of the reliance on local information only. Each time step requires to compute 2 gradients, and referring to (4.5), (4.6), (4.7) and (4.8) we have that $C(A, \nu, E) = 12|\nu| + 7$. The costs related to $\boldsymbol{o}_i$ have been excluded from this analysis as obstacle avoidance is not a critical part of DORA-Explorer. The step-wise communication and computational costs for an agent are thus both bounded by:

$$D(A, \nu, E) \text{ and } C(A, \nu, E) \in \Theta(|\nu|) \tag{4.9}$$

Such low costs mean that DORA-Explorer should scale well to a large number of robots and should enable real time computation on even very limited computational platforms.

4.5 Simulations

4.5.1 Experimental setup

We tested our system through simulations in ARGoS [11], which is an open-source physics-based simulation environment designed for robotic swarms. The agents we used in the simulation are KheperaIV robots [85] programmed in Buzz [9] to facilitate swarm management and interaction.

We deployed a set of N = {10, 15, 20} robots in a simulated environment of 20x20m with set of 2 radiation sources. The environment is discretized into 400 cells, where each cell of the grid is 1x1m large. The robots' initial positions are chosen randomly. We arbitrarily set λ from (4.1) to be 5 as it was providing an adequate decay speed in relation with the size of our

environment: The robots would generally fail in a 3m radius around the radiation sources. The speed constant k from (4.8) is set to 20 to match the maximal speed of the KheperaIV robots. Because no information gain can be achieved by a failed robot, failure must be avoided. This leads to choosing $\alpha >= \beta$ in (4.8). For our experiments, we set $\alpha = 2, \beta = 1$ and $\gamma = 1$. The robots are all given a random initial orientation. Radiation sensing is emulated by an ARGoS controller reading the randomly generated radiation sources. Failures are randomly triggered by using (4.3): if $f_i = 1$, the robot stops exploring. We added 5 randomly distributed 0.8m x 0.8m obstacles to verify the robot's ability to perform exploration even in cluttered environments.

We performed 50 simulation runs over 300 steps of the DORA-Explorer algorithm. Each time step lasts 0.8s. To assess DORA-Explorer's performance, we compare it to the results obtained by a random walk algorithm and by a FBE algorithm. The latter's key principle is to assign one of three states (explored, frontier, unexplored) to the cells constituting the environment and to coordinate the robots to explore the regions near the frontier. To implement it, we adapted the algorithm from [46] by having the robots share an exploration map through a virtual stigmergy. The comparison with frontier exploration is particularly relevant because it allows us to gain insights on our algorithm performance in terms of terrain coverage compared to an algorithm which was specifically designed to maximize this objective. We also compare DORA-Explorer with a random walk algorithm as a baseline it absolutely needs to outperform. These two baselines are commonly used for the exploration of unknown environments in the field of swarm robotics. They do not take risk into account, but to the best of our knowledge, no other existing swarm exploration strategy does. To address this issue, we could have modified the baselines to take risk into account, but chose against it. Adding risk thresholds for movement could be considered. However, if the robots find themselves in radiation hotspots, they might remain stuck in these locations because all surrounding cells will a have similar/equal risk which is too high to allow movement, resulting in fatal stagnation. The first metric used to assess the validity of our approach is the number of robots which remain active (not failed) over time. This is perhaps the most important metric because it shows how well DORA-Explorer performs in terms of risk avoidance, i.e. its main objective. The second metric used to evaluate the algorithms is the total number of cells explored by the swarm. This allows us to evaluate how well our algorithm performs in its objective of maximizing information gain and to verify that avoiding risk does not impact too much the exploration performance. The third metric we studied is the communication costs of the algorithms, measured in KB of data transmitted per robot at each time step. We included this in our analysis to examine if the algorithms can scale to large number of robots. We also ran simulations to study the impact of the parameters α and β from (4.7).

4.5.2 Results

The following results are an average of the 50 simulation runs of each algorithm.

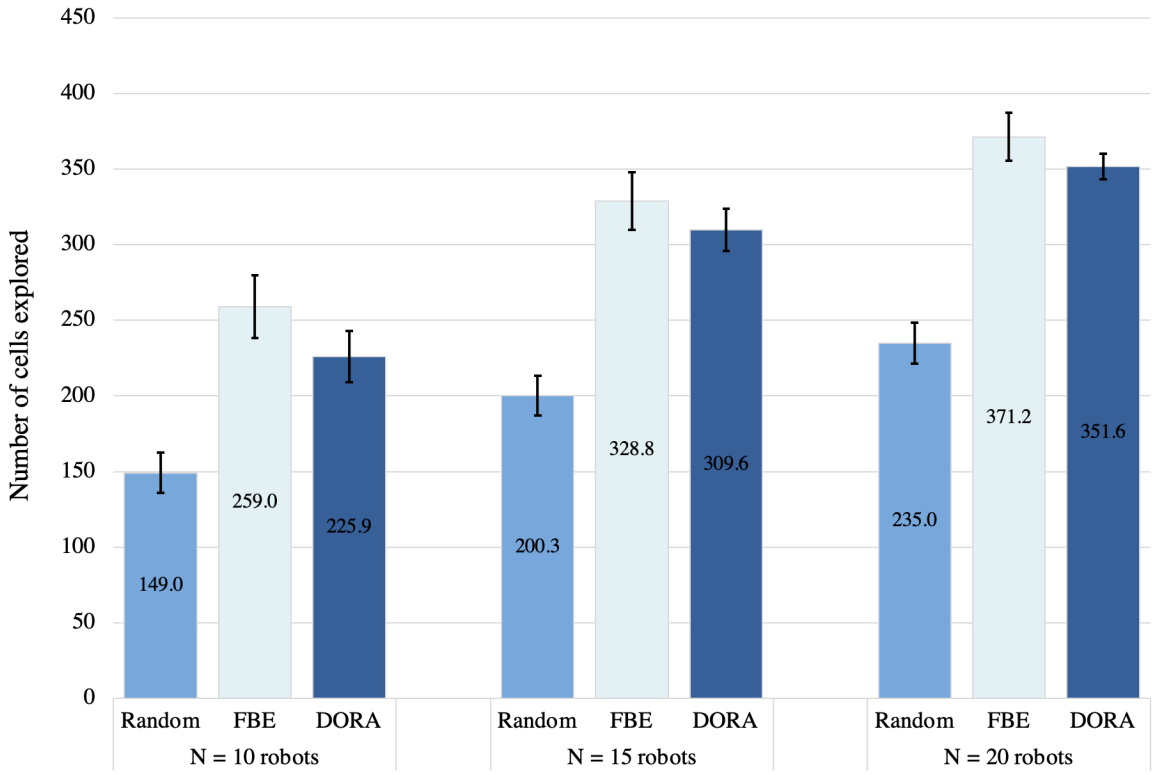


Figure 4.2 Performance comparison of DORA-Explorer, FBE and random walk for number of explored cells at the end of the simulation.

Results from Fig. 4.2 show that FBE achieves slightly higher exploration coverage than DORA-Explorer, but this gap in performance decreases as the number of robots increases. This is an expected result, because DORA-Explorer's main goal is not to achieve maximal coverage at all costs, unlike FBE. Both FBE and DORA-Explorer clearly outperform the random walk algorithm. The other trend is that adding more robots to the swarm results in a higher number of cells being explored for all three algorithms after 300 steps. This shows that DORA-Explorer scales well to large number of robots, and even gains in performance when swarm size increases, which is in line with the benefits associated with swarm algorithms. In terms of avoiding failures, DORA-Explorer unsurprisingly outperforms both FBE and the random walk, as it is its main purpose. This is shown in Fig. 4.3, where DORA-Explorer exhibits a higher level of active robots at the end of the simulation runs, with this difference only increasing with larger swarm sizes. For all values of N, there are few survivors for FBE,

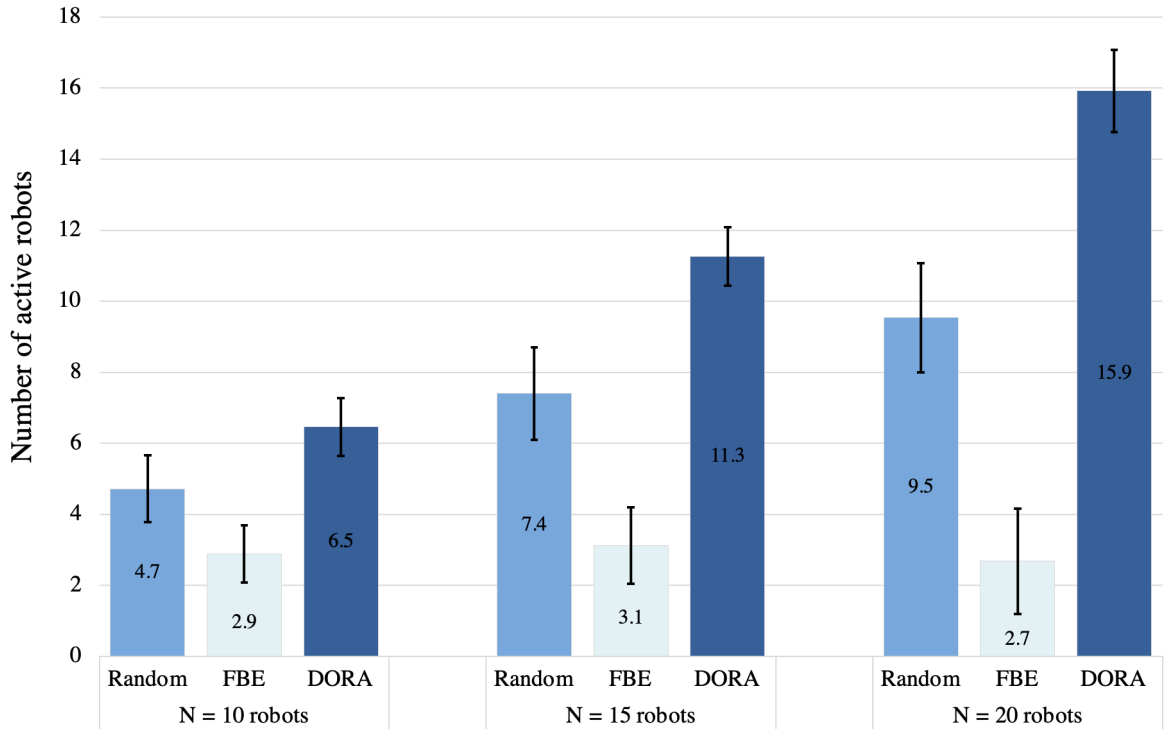


Figure 4.3 Performance comparison of DORA-Explorer, FBE and random walk for number of active robots at the end of the simulation.

and random walks perform only slightly better, while DORA-Explorer keeps most robots alive, achieving its objective.

Fig. 4.4 shows the DBMs obtained at the end of an arbitrarily selected simulation where N = 20 for each algorithm. In other words, it represents which cells were explored by each algorithm and the sensed radiation intensity associated with them for one specific run.

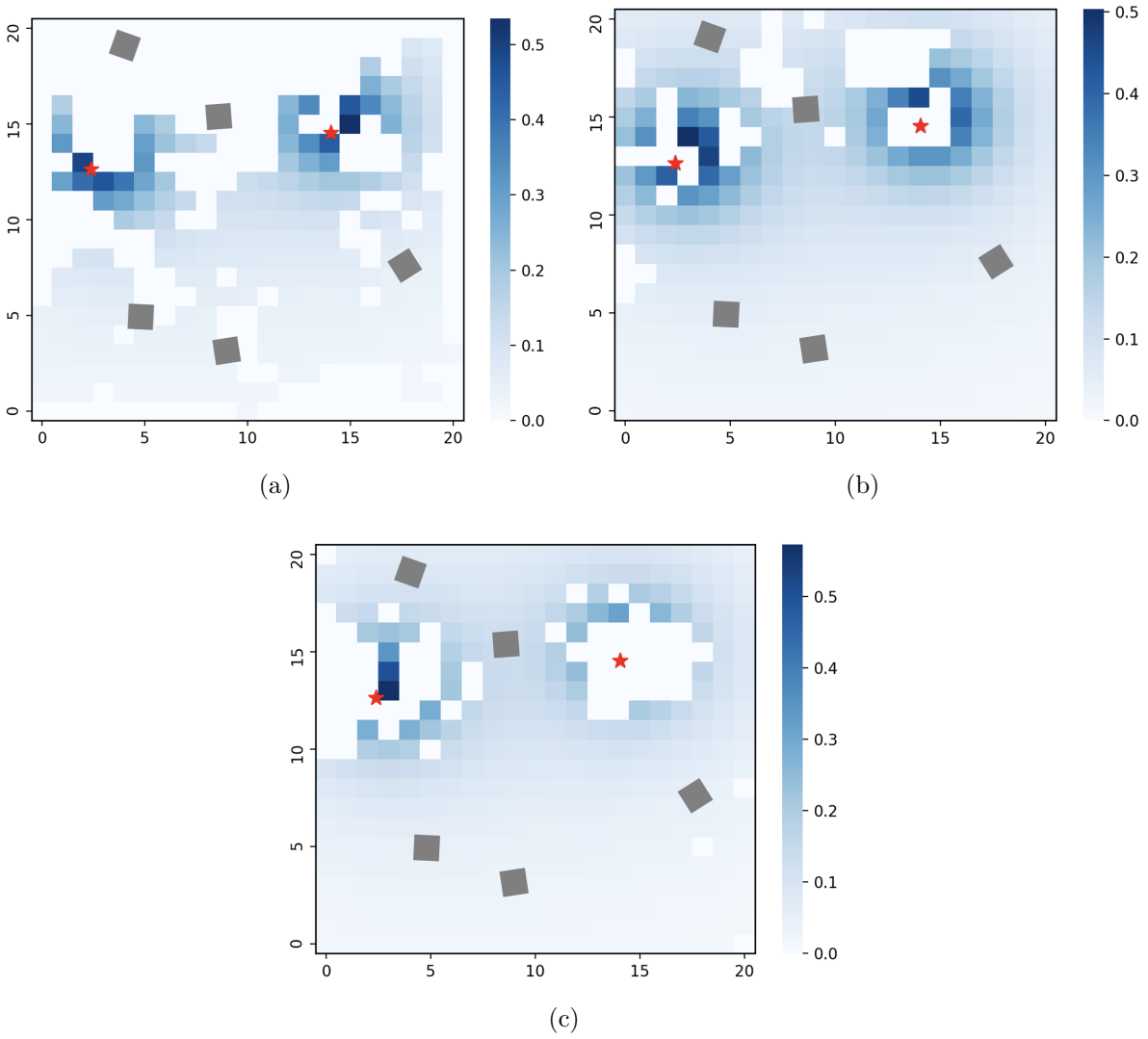


Figure 4.4 (a) Random walk (b) FBE (c) DORA-Explorer. Radiation belief maps of the 20m x 20m environment for each exploration algorithm of one specific simulation. Blank cells are unvisited areas, red stars are the point radiation sources and grey squares are the randomly generated obstacles.

The random walk covered much fewer cells than DORA-Explorer and FBE, which both covered roughly the same areas of the map, with the same sections remaining unexplored.

However, these areas remained unvisited for different reasons. For FBE, the parts of the environment close to the radiation sources remained uncovered because its agents failed when approaching them. In contrast, DORA-Explorer did not explore these cells because it *avoided them*. Again, DORA-Explorer achieves very similar coverage than FBE but does so with less robot failures. In this particular simulation, DORA-Explorer finished with 18 active robots, random walk finished with 7, and FBE with none.

The results from Fig. 4.5 show the amount of data transferred by individual agents at each time step by both algorithms. We excluded the random walk algorithm from this figure as it does not require any coordination or communication between its agents. DORA-Explorer transmits more data than FBE, which was expected because the former shares information through two DBMs, while the latter uses only one. In section 4.4.5, we predicted that the amount of data transmitted at each time step would only depend on the size of the neighborhood used, and this is confirmed by Fig. 4.5, where it remains roughly constant for different number of agents. The small increase in data transmission with increasing number of robots can be attributed to packet collision.

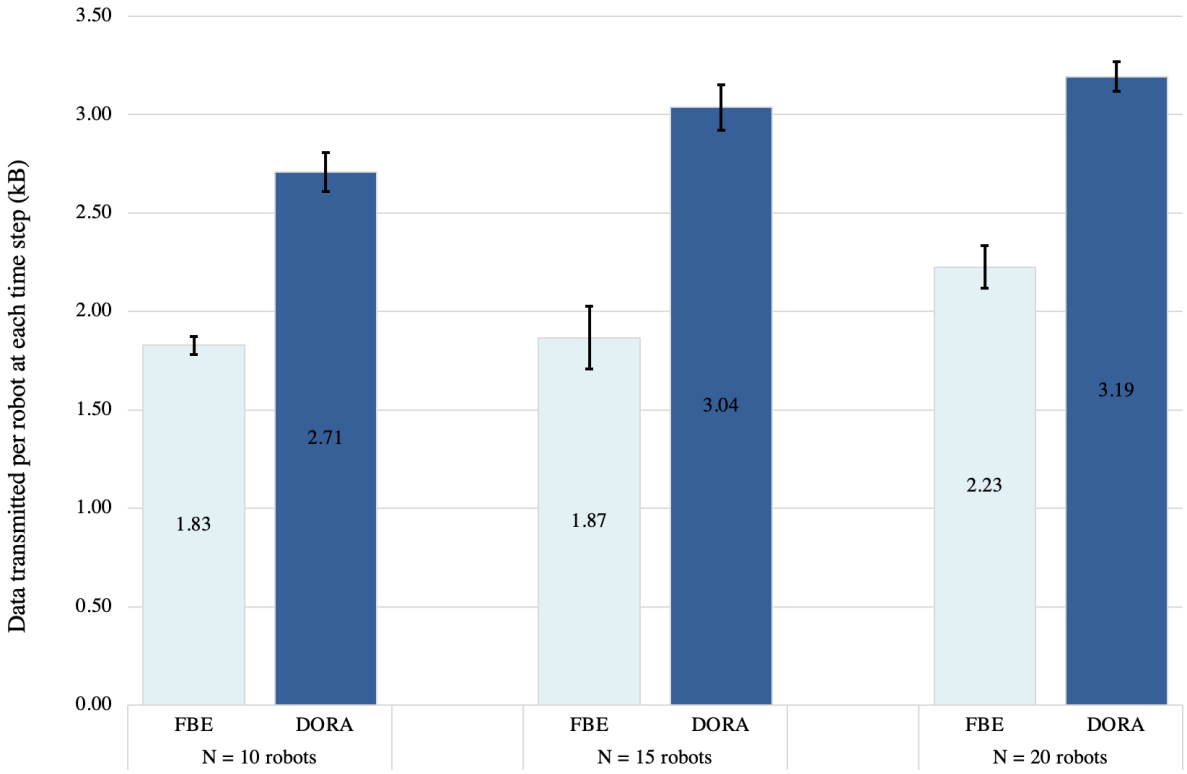


Figure 4.5 Communication costs for DORA-Explorer and FBE

Results from Fig. 4.6 show the performance of DORA-Explorer with varying ratios of risk gain α to exploration gain β. The experiments were carried with 10 robots. Results show that the number of remaining active robots at the end of the simulation only increases with a higher ratio. This is the expected behaviour as increasing the ratio corresponds to giving more importance to the risk avoidance gain from (4.7). As for the number of cells explored, the relationship is not monotonic. In our experiments, a ratio $\alpha/\beta = 2$ provided the best result in terms of number of cells explored. For lower ratios, the robotic team is increasingly impacted by failures which in turns worsen the exploration performance. For higher ratios, the robot are too careful and don't explore as much the environment.

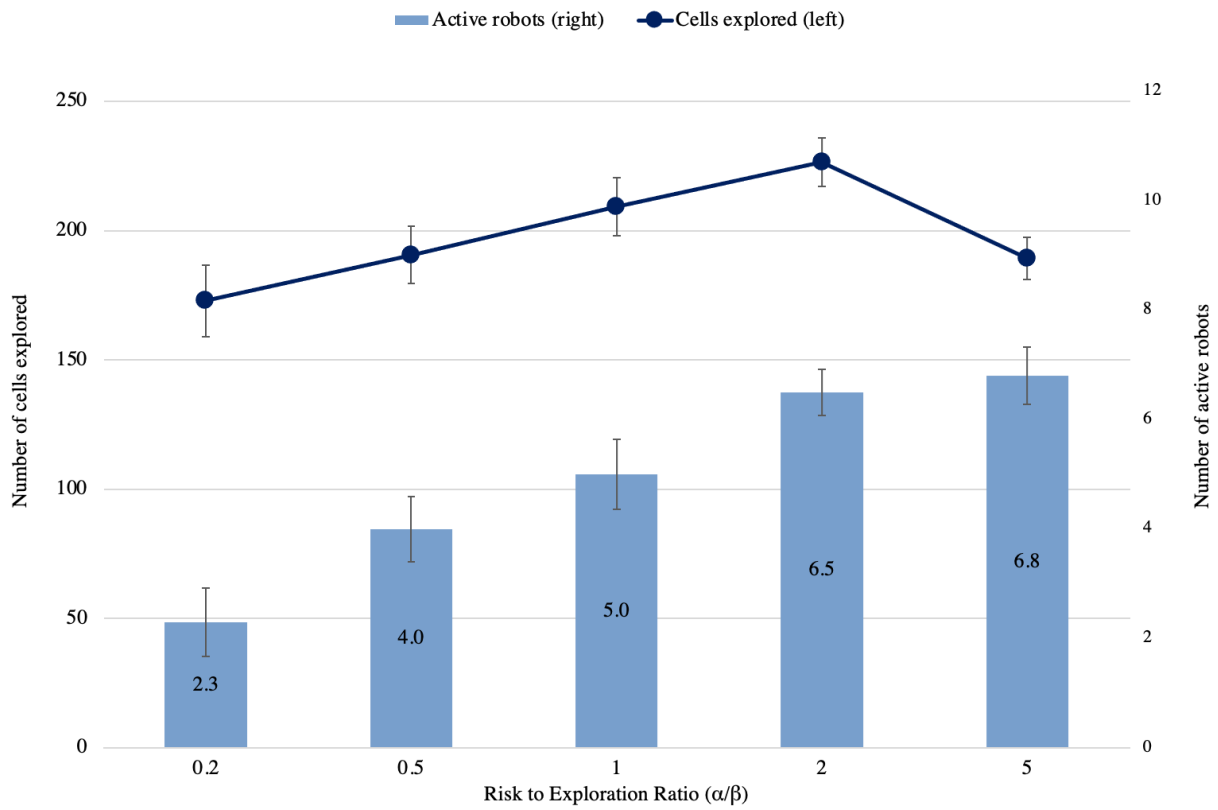


Figure 4.6 Performance of DORA-Explorer with varying ratios of α/β

4.6 Physical experiments

4.6.1 Experimental setup

In addition to the extensive simulations conducted in ARGoS, we tested our system on a team of three physical KheperaIV robots in a 2x2m environment containing 1 point radiation source. The environment is discretized as a 10x10 grid, meaning that each of the 100 cells of the grid is 20x20cm large. Because the arena in which we conducted the experiments was already limited in terms of space we decided not to add obstacles. Positioning of the robots is done using an OptiTrack motion capture system. Radiation sensing is emulated by an on board controller that reads the distance between the robot and the radiation source to determine the current radiation level. Failures are then triggered using equation (4.3). If a robot fails, it stops moving and stops contributing to the exploration effort. The point radiation source is located in a corner of the arena and the robots are initially placed in the three remaining corners. We performed 5 runs over 200 steps of the DORA-Explorer algorithm. Each time step lasts 1s. Again, to assess DORA-Explorer's performance, we compare it to the results obtained by FBE and random walk algorithms.

4.6.2 Results

The following results are an average of the 5 runs of each algorithm on physical robots.

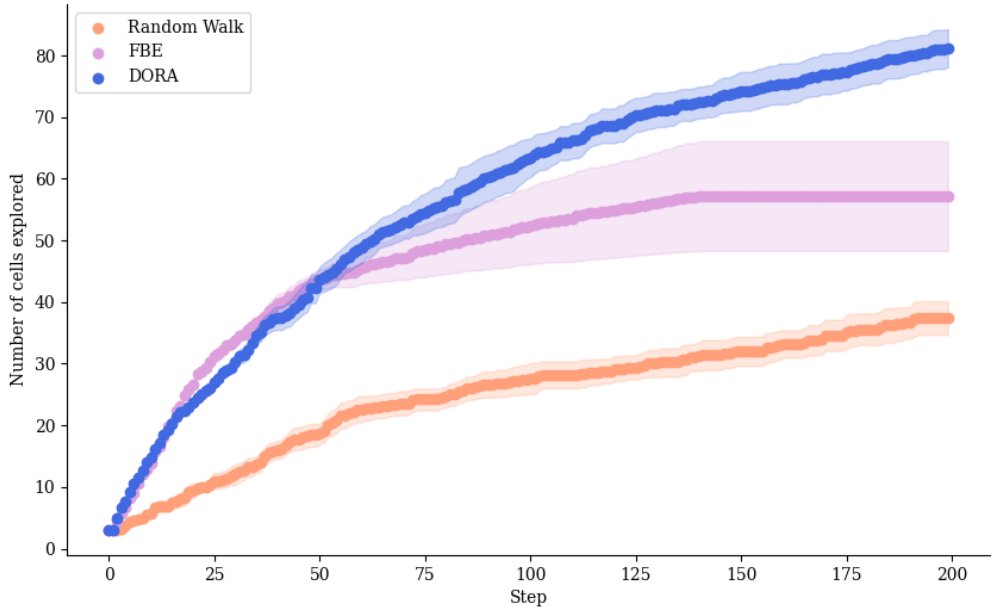


Figure 4.7 Performance comparison of DORA-Explorer, FBE and random walk for number of explored cells over time on physical robots.

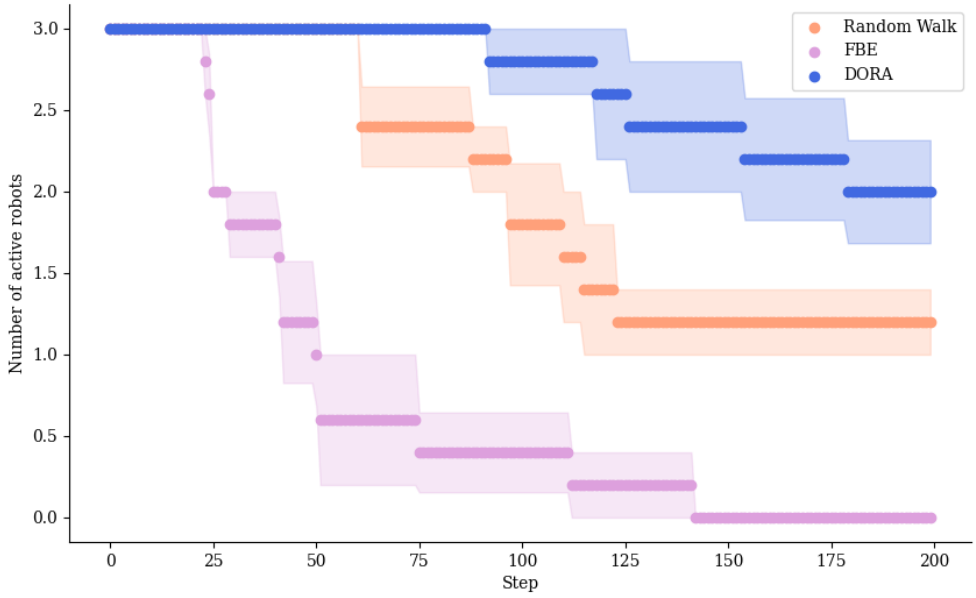


Figure 4.8 Performance comparison of DORA-Explorer, FBE and random walk for number of active robots over time on physical robots.

At the beginning of the exploration process, DORA-Explorer and FBE perform similarly in terms of number of cells explored as shown in Fig. 4.7. The random walk algorithm's exploration rate is considerably lower which can be attributed to the fact that some of the cells of the environment are visited multiple times: robots sometimes come back to positions that they just had visited since their motion is determined randomly. As time progresses, DORA-Explorer starts showing better exploration results than FBE and at the end of the runs DORA-Explorer achieves a considerably better coverage. Both FBE and DORA-Explorer clearly outperform the random walk algorithm. In terms of robot failures, DORA-Explorer outperforms both FBE and the random walk algorithm. This is shown in Fig. 4.8, where DORA-Explorer exhibits a higher level of active robots over time. When using FBE, all three robots always fail before the end of the experiment. Random walk shows a level of active robots that is in between DORA-Explorer and FBE. The results show that while DORA-Explorer and FBE initially have similar performances, as time progresses, DORA-Explorer gets better when compared to FBE. The trend between DORA-Explorer and FBE in the physical experiments is inverted in comparison with the simulations because of the small environment into which the physical experiments were carried. Indeed, the likeliness of getting close of the radiation source was high, and as a result, without risk-awareness, the robots would fail very quickly. Because FBE experiences a lot of failures, the failed robots stop exploring causing the exploration rate to decrease dramatically. In fact, FBE always loses all its robots before the end of the experiments. In contrast, DORA-Explorer keeps

most of its robots active throughout the experiment and as a result the exploration rate remains high.

4.7 Conclusion

We presented DORA-Explorer, a novel lightweight risk-aware exploration algorithm that minimizes the risk to which robots expose themselves in order to maximize the amount of ground they will be able to cover. We expected that our exploration algorithm, which leverages DBMs, would greatly outperform non-coordinated solutions, and this has been the case. Indeed, it succeeded in reducing considerably the likeliness of robot failures while keeping similar ground coverage performance compared to other solutions proposed in the literature. DORA-Explorer also showed good scalability thanks to its low communication costs and its decentralized nature. It also showed applicability to real world scenarios through experiments with physical robots.

Taking inspiration from obstacle avoidance algorithms for the purpose of risk-avoidance could be an interesting future direction. Other future works include allowing DORA-Explorer to become more or less risk-avoiding depending on the changing needs of the situation. For example, in a search-and-rescue scenario, an increasing urgency to rescue victims could motivate the willingness to take more risks as time progresses. Also, more experiments could be conducted by testing DORA-Explorer on a larger team of physical robots exploring larger outdoor environments. Further applications of DORA-Explorer could include using the generated risk belief map to determine robots' fitness to store data in distributed storage systems like SwarmMesh [19], with robots assigned to tasks in dangerous regions being discouraged from storing sensitive information. Finally, in this work we considered that risk could be sensed by an on-board sensor. However, in some scenarios, the risk cannot be directly perceived by any sensors. In these cases, the belief map could instead be constructed using the previous failures of the agents by assigning risk to areas only where failures have been detected in the past.

CHAPTER 5 RISK-AWARE ROUTING IN ROBOT SWARMS

In this chapter, a Risk-Aware Swarm Storage (RASS) algorithm is presented. RASS has been heavily inspired by DORA-Explorer and similarly to the latter, brings risk awareness to a robot swarm algorithm. Specifically, the algorithm makes the following contribution to the field of swarm robotics: A fully decentralized storing and routing algorithm where the decisions made are based solely on local interactions. The absence of central coordination makes it well suited for robot swarm applications where scalability is at the core of the problem. The figures presented in the chapter are taken from [86] with permission from the authors.

5.1 Introduction

The exploration of unknown environments has proven to be more effective using teams of robots instead of a single robotic unit [1]. Search and rescue scenarios [87] or nuclear inspection and cleanup [53], where speed is of the essence, are therefore well suited for multi-robot applications. However, data storage remains a challenge for such systems as the amount of collected information only increases with the number of robots. The unreliable connectivity that these systems typically suffer from [12] inhibits sending collected data items directly to external storage. Robots often need to store locally the data items until a path towards permanent storage becomes available. Additionally, because robots usually have a limited communication range, the data items collected during the mission may need to be routed through multiple robots before reaching the external storage infrastructure. The multi-robot system becomes a temporary storage infrastructure and deciding where to store and send the data items become essential. Routing the data items through the shortest path towards the base station may seem natural, however, because the environment into which the mission is carried is usually uncontrolled, environmental hazards can compromise some of the nodes of the system. For example, routing information through a robot located near a radiation source might cause data corruptions. Avoiding such nodes of the system can effectively increase the reliability of the system; thus risk should be considered when storing and routing data items. We propose a fully decentralized Risk-Aware Swarm Storage (RASS) system that actively routes data items towards a base station while avoiding dangerous nodes of the system. The algorithm relies solely on local interaction to determine which nodes are the fittest for storing information and works in both static and dynamic topologies.

5.2 System model

We consider a multi-robot system exploring an unknown environment in a fully decentralized fashion. The agents of the system, denoted as $a_i \in A$, are assumed to have limited storage capacities and communication capabilities. While exploring the environment, robots keep acquiring new data and try to convey the information to a base station for permanent storage. Because of their limited communication range R, robots may not be able to directly send data items to the base station. They might instead need to route the information through multiple robots before reaching the base station. Finally, point radiation sources S with intensities $I_j \sim \mathcal{U}(0, 1)$ are randomly distributed in the environment.

5.2.1 Risk Modelling

We assume the radiation sources to be the cause of data corruptions on nearby robots [88,89]. Figure 5.1 shows an example of risk estimates held by robots based on their respective locations. The closer they are to the radiation sources, the higher the risk level will be. We assume the robots to be able to sense the radiation level associated with their current position using an on-board sensor.

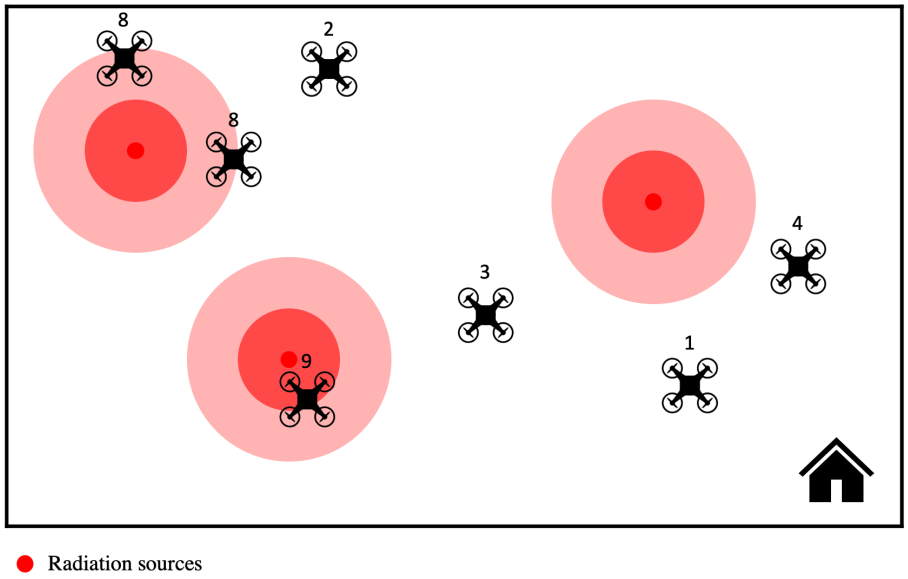


Figure 5.1 An example of risk levels (from 0 to 10) measured by robots

The total radiation level perceived by a robot a_i at position $\boldsymbol{x}_i \in E$ is given by:

$$r(\boldsymbol{x}_i) = b + \sum_{\boldsymbol{s_j} \in S} \frac{I_{\boldsymbol{s_j}}}{1 + \lambda \rho(\boldsymbol{x}_i)^2} \tag{5.1}$$

The radiation level decays exponentially with the Euclidean distance $\rho(\boldsymbol{x}_i)$ between the position of the point radiation source $\boldsymbol{s}_j$ and the robot $\boldsymbol{x}_i$. λ is used as a decay rate parameter and Gaussian measurement noise is added to the readings of the on-board sensor.

We assume that the radiation sources affect the robots independently and therefore, the probability of data items to become corrupted is given by:

$$\mathbb{P}(c_i = 1|S) \sim \mathcal{B}(r(\boldsymbol{x}_i)) = 1 - \prod_{\boldsymbol{s_j} \in S} 1 - \mathbb{P}(c_i = 1|\boldsymbol{s}_j) \tag{5.2}$$

5.2.2 Potential-Based Percolation

RASS is built upon the assumption that nodes of the system exposed to higher levels of radiation should be used less for storing and routing data items. Indeed, because radiation is assumed to cause data corruptions, avoiding dangerous locations of the system should offer better reliability results and effectively reduce the number of lost data items. Additionally, to alleviate the memory usage of the robots of the system, percolating data items towards a base station for permanent storage should be encouraged. From this, and taking inspiration from [19], we define a fitness policy that indicates how "fit" an agent is for storing data items. The fitness of an agent is given by:

$$\phi_i = \begin{cases} \frac{1}{\alpha h_i + \beta r(\boldsymbol{x}_i)} & \text{if } m_i > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.3}$$

where α and β are respectively the routing and risk control gains, m_i is the available memory of agent a_i, r_i is the risk given by eq. 5.1 and h_i is the minimum hop count to the base station. An example of a minimum hop count gradient for a robot swarm is presented in figure 5.2.

When a node becomes unfit to store data items, it will simply evict items to its fittest neighbour:

$$T\phi_i < \max_{j \in \mathcal{N}} \phi_j \tag{5.4}$$

$\mathcal{N}$ is the set of neighbours of agent a_i and T is a fitness threshold that ensures that data

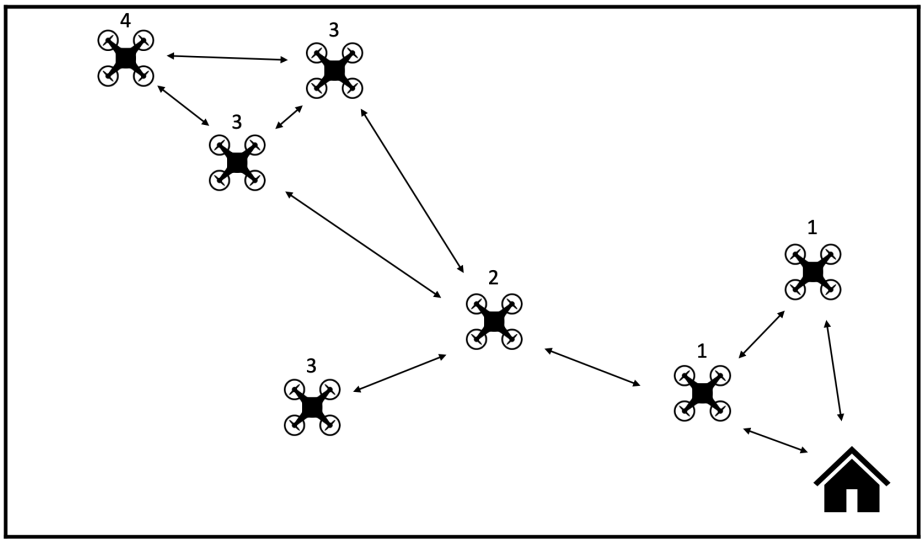


Figure 5.2 An example of hop count distance to base station

items are routed only when a neighbour is significantly better for storing them.

Because of the hop count gain included in the fitness policy, data items will tend to passively percolate towards the base station. Indeed, for similar levels of risk, nodes located closer to the base station should have higher fitness. RASS' execution loop is summarized in algorithm 1. Because of its fully decentralized nature, all the robots of the swarm execute the same code.

Algorithm 1: RASS Execution Loop

```
while Running do
    get_hop_count()
    get_risk_measurement()
    update_fitness()

    if not is_fit() and |𝒩| > 0 then
        evict_data()
    end

end
```

5.3 Simulations

To assess the validity of our system, we tested RASS through simulations in the physics-based simulator ARGoS [11]. The agents used in the simulation are KheperaIV robots [85]

and RASS' implementation is done using the Buzz programming language from [9].

To thoroughly assess RASS' performance, four different robot topologies were studied:

- Grid like pattern (static topology) as shown in figure 5.3
- Scale-free pattern (static topology) as shown in figure 5.4
- Lennard-Jones potential interactions (dynamic topology) as shown in figure 5.5
- Random search motions (dynamic topology) as shown in figure 5.6

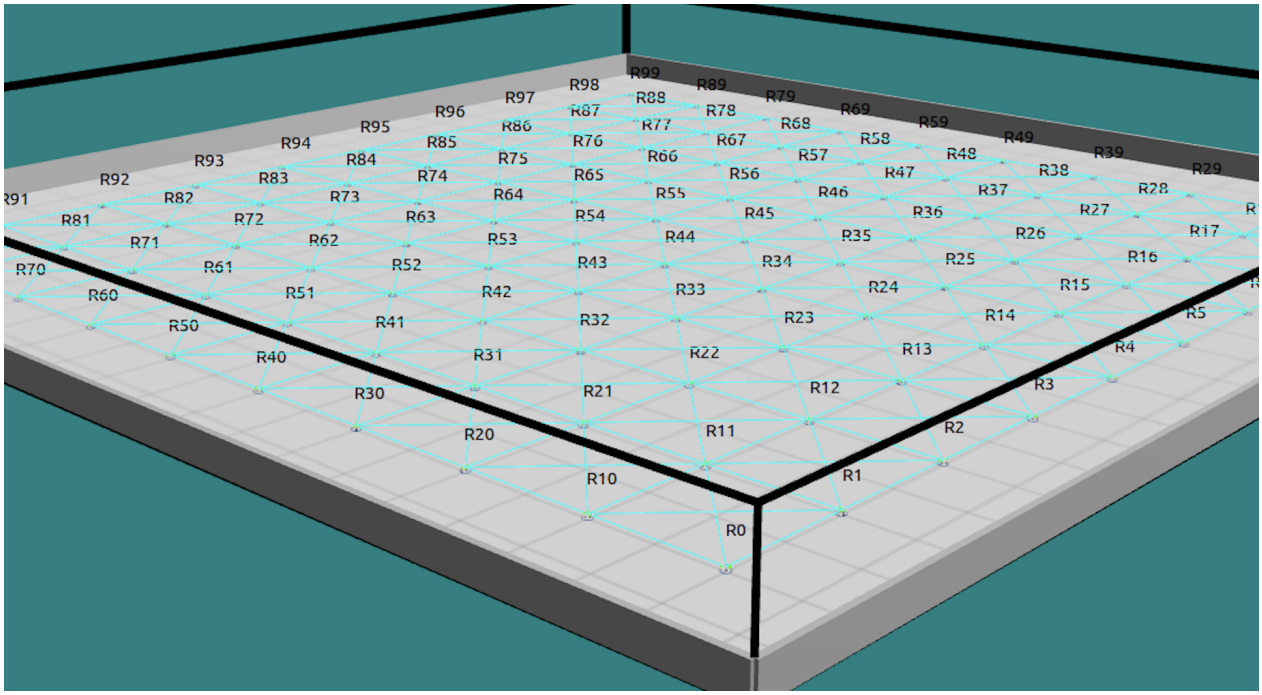


Figure 5.3 100 KheperaIV robots in 20m x 20m ARGoS simulated environment distributed in a grid-like pattern.

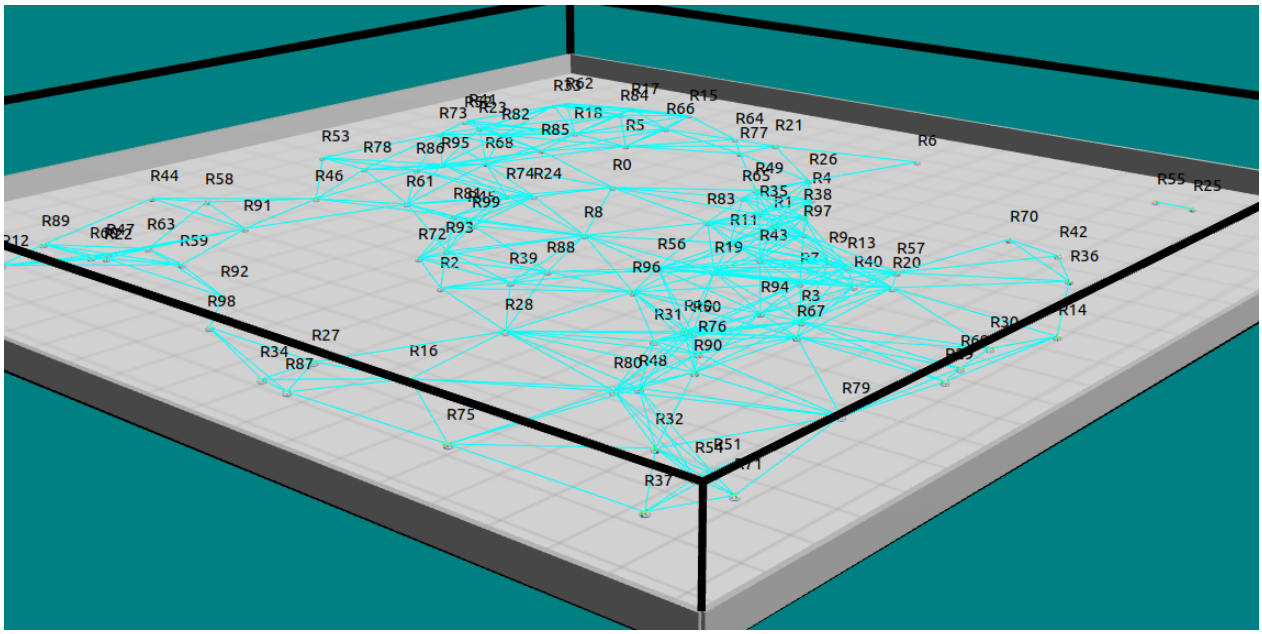


Figure 5.4 100 KheperaIV robots in 20m x 20m ARGoS simulated environment distributed in a scale-free pattern.

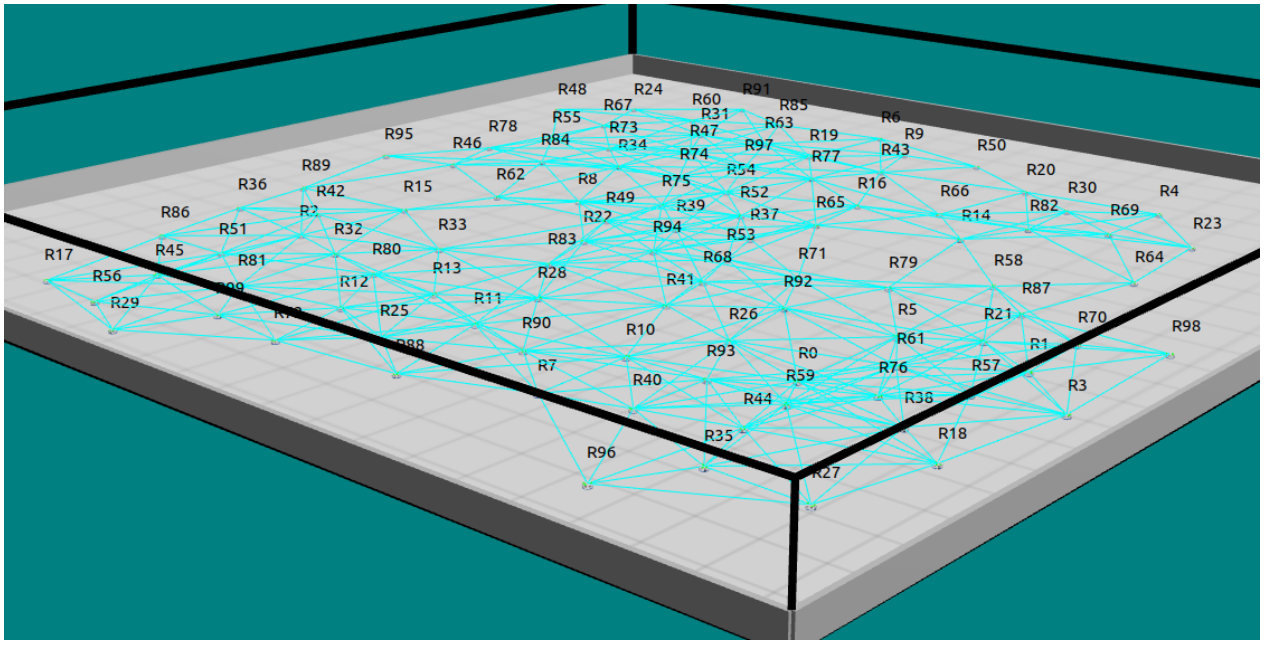


Figure 5.5 100 KheperaIV robots in 20m x 20m ARGoS simulated environment in a dynamic topology obtained through Lennard-Jones potential interactions.

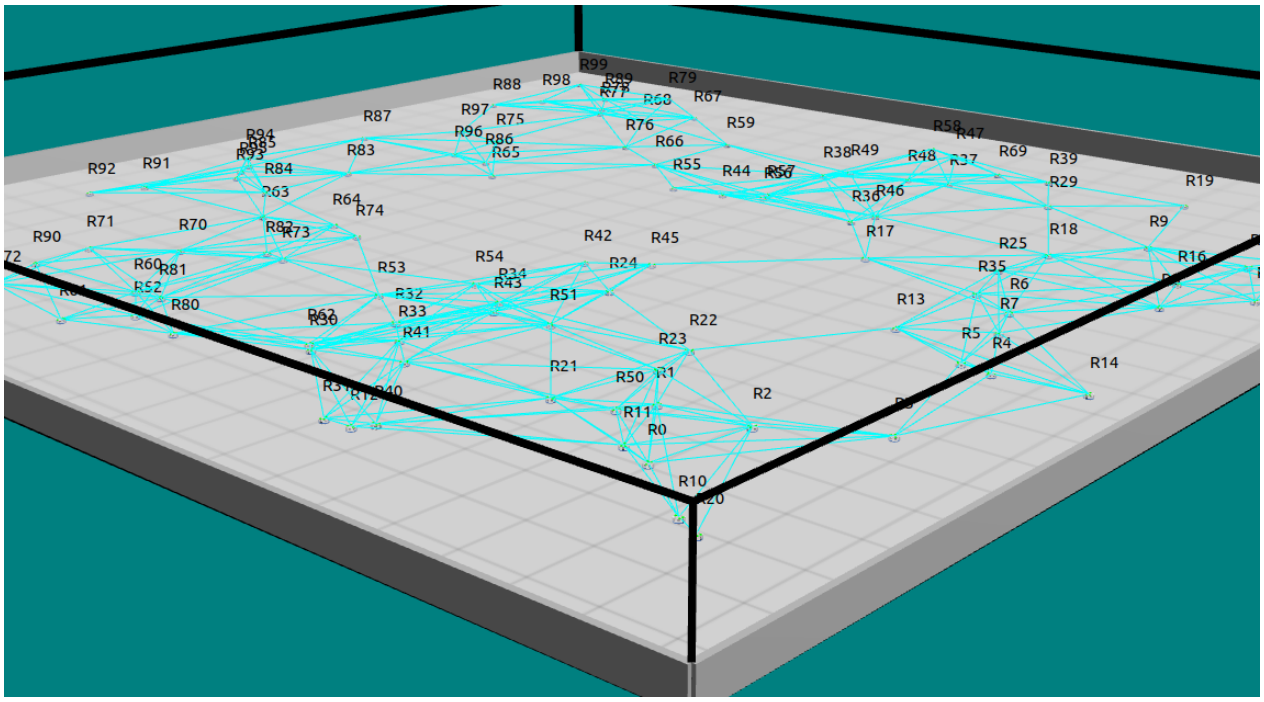


Figure 5.6 100 KheperaIV robots in 20m x 20m ARGoS simulated environment in a dynamic topology obtained through random walk motions.

We deploy N = 100 robots with a communication radius $R = 3$ meters in a 20m x 20m environment containing a set of 3 randomly distributed radiation sources. The base station's storage capacity is assumed to be infinite and is located in one of the corners of the environment. To bring our simulations closer to real-world scenarios, we added bandwidth restrictions to the amount of data robots can possibly exchange. In our simulations, the maximal number of data items that could be exchanged between two neighbouring robots was 10 items every time step. Data items were generated at a fixed interval by all robots in the system. The storage capacity of robots was set to 50 data items. For our simulations, we used $\alpha = 10$ and $beta = 1$ for Eq. 5.3. This choice was justified by the fact that the risk $r(x_i)$ was bounded by [0,1] while the hop-count value h_i was not. In our simulations, h_i was usually upper-bounded by 10. Therefore, choosing of value of $\alpha = 10$ gave the two parameters a similar importance when determining the fitness in Eq. 5.3. Two benchmark algorithms were used to assess RASS' performance:

- A routing algorithm based on hop count
- The virtual stigmergy from [17]

The hop-count based algorithm enables comparison with a commonly used routing mechanism meant to yield fast transfer speeds. The virtual stigmergy was chosen to showcase the benefit of percolating data towards a base station.

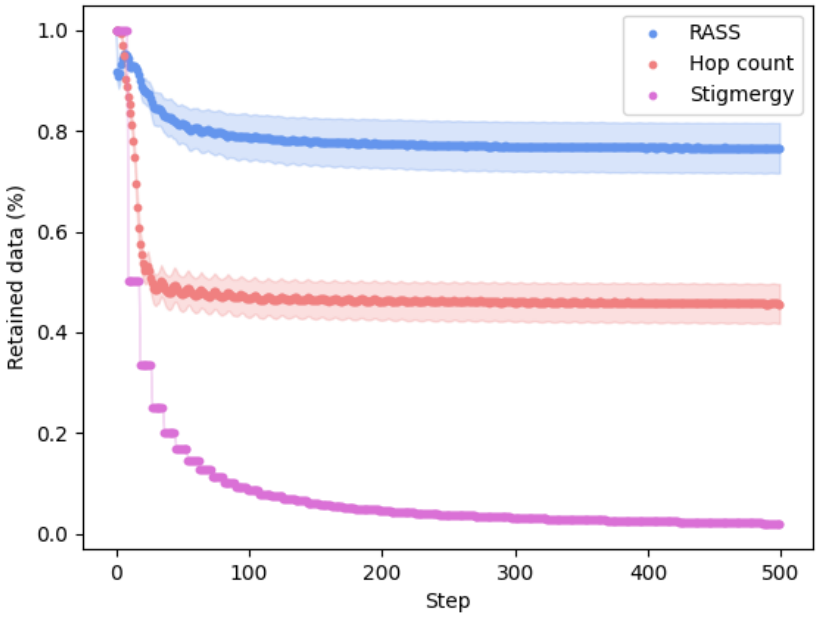
The results obtained in the simulations for the different topologies are presented in Fig. 5.7, 5.8, 5.9 and 5.10 as well as in table 5.1. They show that RASS persistently outperforms the hop-count based algorithm and the virtual stigmergy in terms of reliability across all four topologies. Indeed, the percentage of retained data for RASS is usually between 80% and 100%, meaning that no more than 20% of generated data items during the experiments are lost due to corruption. On the other hand, the hop-count based algorithm offers reliability results ranging from 50% to 80% and is therefore more affected by the radiation sources of the environment. Finally, the virtual stigmergy offers low reliability results as the storage capacity of the agents is quickly saturated and new data items cannot be stored any more. In terms of transfer speeds, results show that the fastest routing mechanism is the hop-count based algorithm. In average, RASS is 54.9% slower than the hop-count based algorithm, in other words, it takes 54.9% more steps for data items to reach the base station using RASS. This is not surprising as the hop-count based algorithm always takes the fastest route to the base station, without considering the risk associated with it. In opposition, RASS takes the route that optimizes both speed and safety. RASS offers better reliability results at the cost of slower transfer speeds.

5.4 Physical experiments

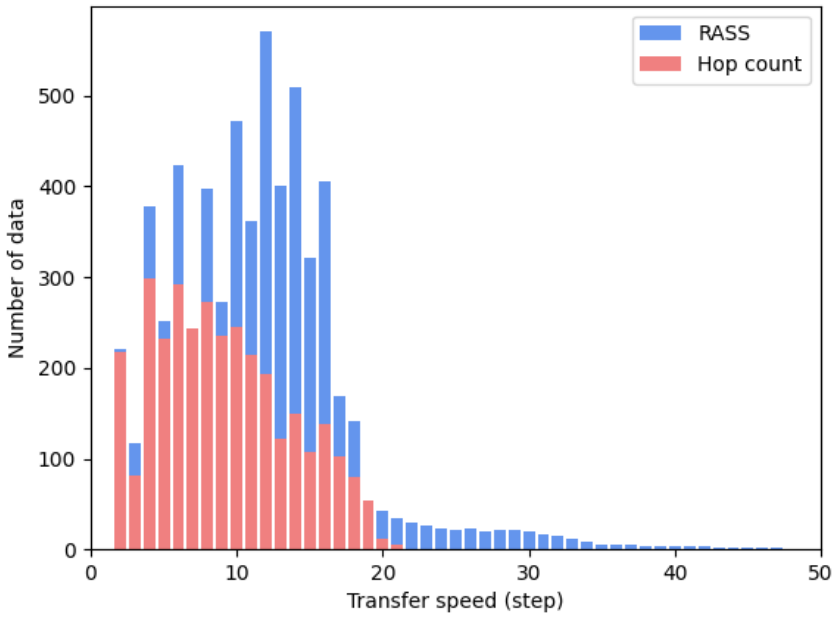
To assess its real-world applicability, we also tested RASS with a swarm of 5 physical robots using a static topology illustrated in fig. 5.11 and fig. 5.12.

The goal was showing that the algorithm was easily transferable to physical robots and that it behaves the ways it is intended to. In this particular topology, one route is clearer safer than the other. We expect RASS to choose the safest one even if it is not necessarily the shortest towards the base station. We performed 3 runs of the RASS algorithm and compared its performance with the hop-count based algorithm. Results are presented in fig. 5.13.

Again, results show that RASS offers better reliability results than the hop-count based algorithm. RASS' percentage of retained data is about 75%, what was expected in the particular topology. Indeed, one of the four robots is located on the radiation source and therefore looses almost all its data items due to corruption. The other three robots are able to route their data items using the safe route. On the other hand, the hop-count based algorithm looses a bit more than 50% of the data items as the two possible routes towards the base

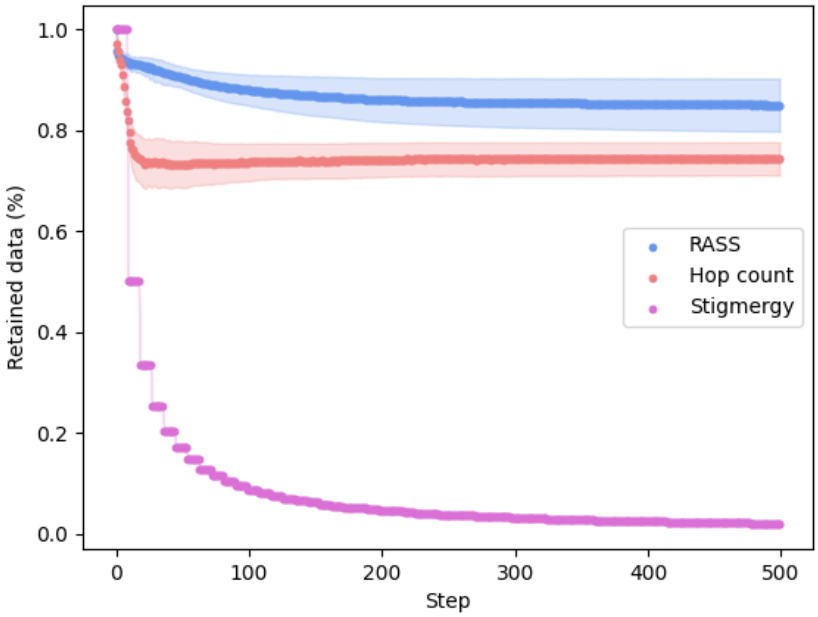


(a)

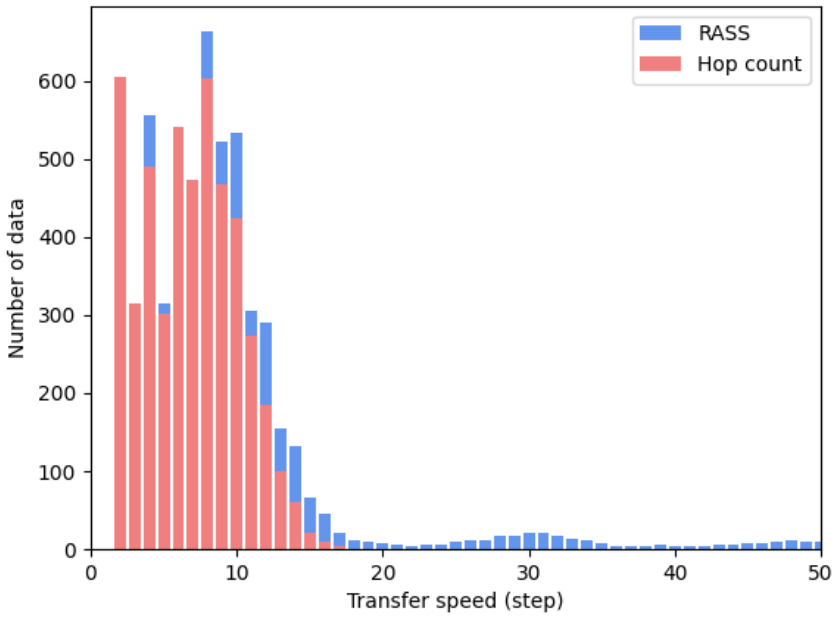


(b)

Figure 5.7 (a) Reliability over time (b) Distribution of transfer speeds. Performance comparison of RASS, hop count and stigmergy in a static grid-like topology.

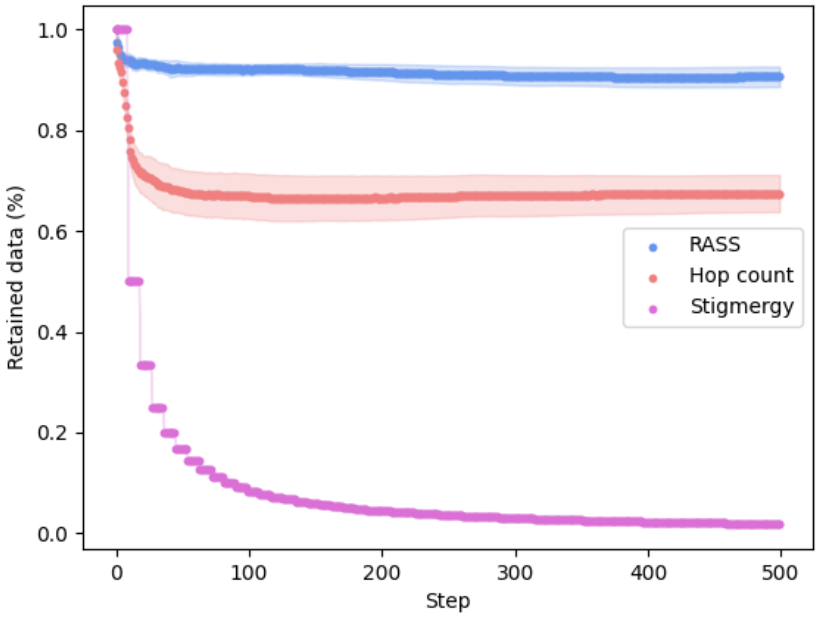


(a)

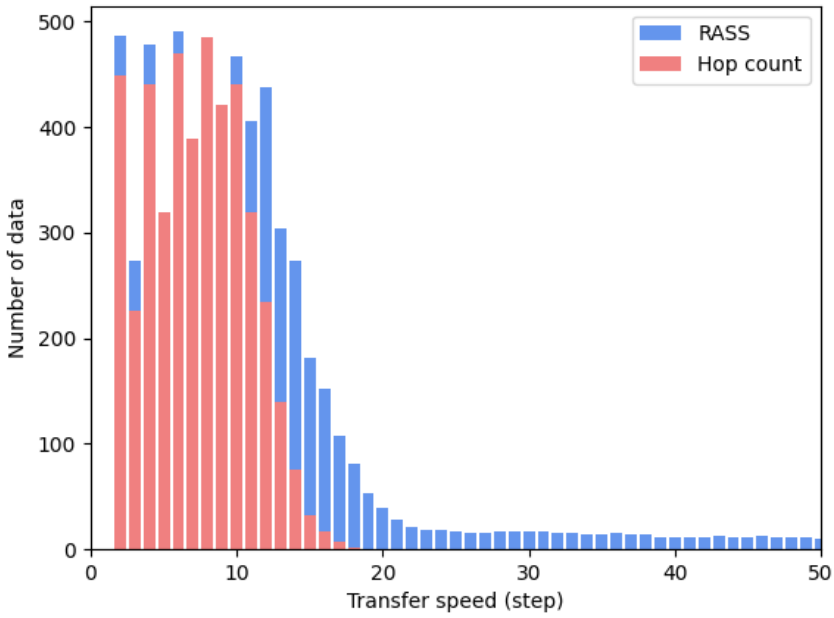


(b)

Figure 5.8 (a) Reliability over time (b) Distribution of transfer speeds. Performance comparison of RASS, hop count and stigmergy in a static Scale Free topology.

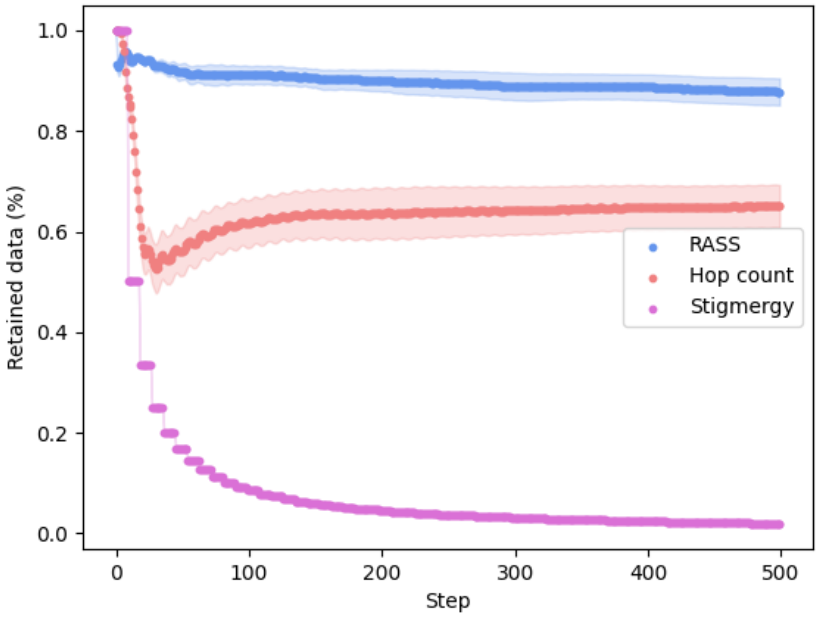


(a)

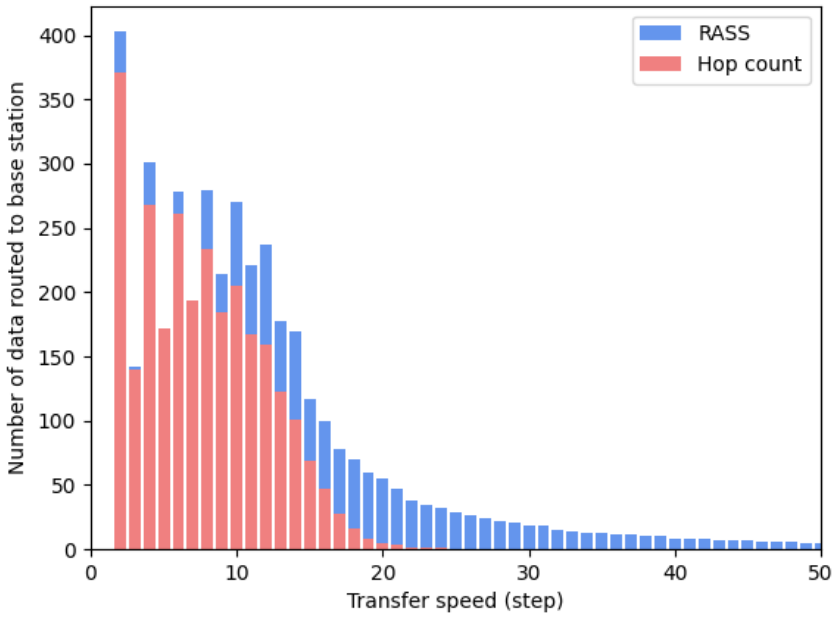


(b)

Figure 5.9 (a) Reliability over time (b) Distribution of transfer speeds. Performance comparison of RASS, hop count and stigmergy in a dynamic Lennard-Jones topology.



(a)



(b)

Figure 5.10 (a) Reliability over time (b) Distribution of transfer speeds. Performance comparison of RASS, hop count and stigmergy in a dynamic random search topology.

Table 5.1 Average transfer speed and average individual memory usage with different topologies

Topology	*Algorithm*	*Transfer speed (hops)*
Grid-like	RASS	11.45
	Hop-Count	9.11
	Stigmergy	N.A.
Scale-Free	RASS	11.44
	Hop-Count	6.85
	Stigmergy	N.A.
Lennard-Jones	RASS	12.51
	Hop-Count	7.32
	Stigmergy	N.A.
Random Search	RASS	12.68
	Hop-Count	7.76
	Stigmergy	N.A.

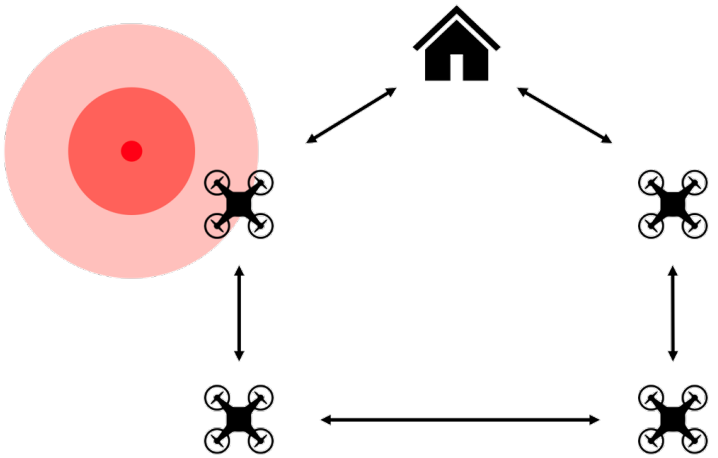


Figure 5.11 Topology of the 3x3m environment with 4 drones, a base station and a radiation source used in the physical experiments.

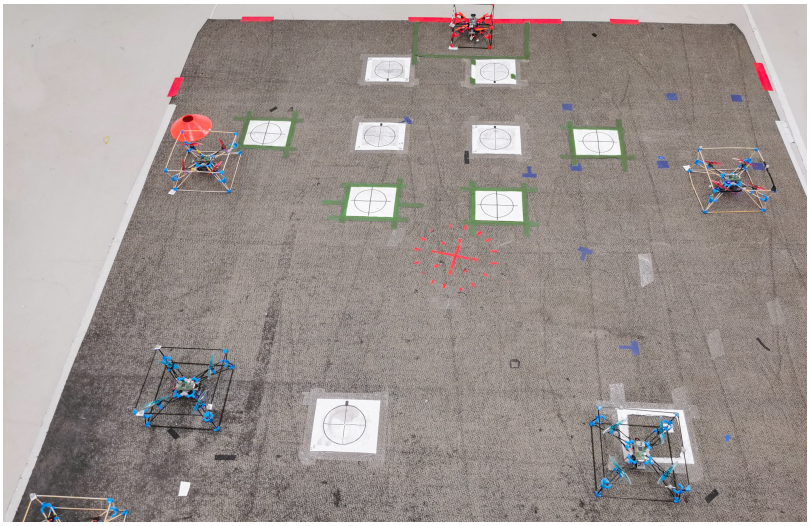


Figure 5.12 Picture of the 3x3m environment with 4 drones, a base station and a radiation source used in the physical experiments.

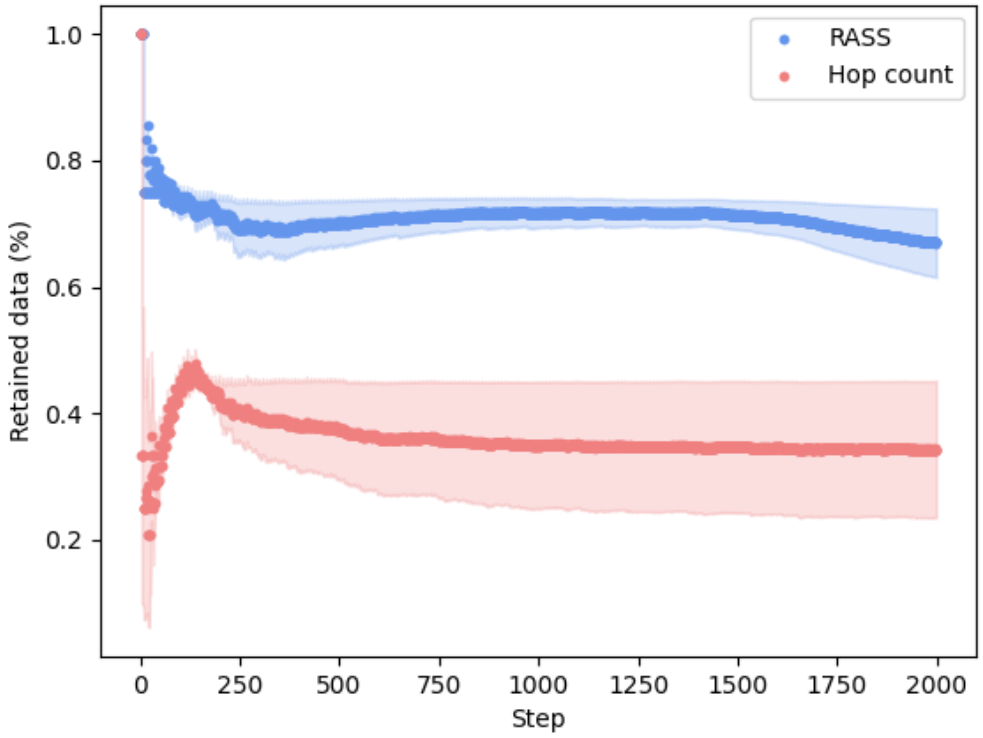


Figure 5.13 Evolution of reliability over time on the physical experiments

station as taken analogously, regardless of the risk. It is true that very few conclusions can be drawn by the physical experiments as the swarm size is very limited and the topology chosen arbitrarily. However, the main motivation was showing that the algorithm, as it is, can be easily implemented on physical robots and can run the way it is intended to. Of course, further physical experiments, with a greater number of robots and more realistic topologies, will need to be carried to assess RASS' performance in real-world scenarios.

5.5 Conclusion

We presented RASS, a fully decentralized risk-aware storing and routing algorithm. By leveraging a fitness policy based on both hop count and risk, RASS is able to choose the appropriate agents through which to route data items towards the base station. Results from our experiments show that RASS consistently outperforms the hop-count based algorithm in terms of reliability across different topologies, both static and dynamic. RASS also performed well in physical experiments where the main motivation was showing that the algorithm, as it is, can be easily implemented on physical robots and can run the way it is intended to.

Additional experiments on larger swarms of robots and in more diverse scenarios will be necessary to assess RASS' performance in real world missions. Also, some additional work includes understanding the impact of the communication network assumptions on the performance of RASS. For example, a network where there exists only a few routes towards the base station and where one is clearly safer than the other is, of course, well suited for our algorithm. On the other hand, the value of RASS should decrease in well-connected networks where the radiation sources are only located at the periphery of the network (far from the base station). Indeed, in this particular scenario, because data naturally moves away from the risk, RASS' risk awareness shouldn't improve noticeably the network's reliability.

CHAPTER 6 GENERAL DISCUSSION

In this chapter, a general discussion regarding the results obtained during the master's degree in regard to the objectives defined in section 1.3 will be presented. We will highlight the main contributions of the work and its importance in the swarm robotics community. First, a discussion on DORA-Explorer will be held, followed by one on RASS.

6.1 Risk-Aware Exploration

As a reminder, the main objective of the research was bringing risk awareness to swarm algorithms intended for exploration missions in hazardous environments.

Specific to DORA-Explorer, the research objectives were the following:

1. Reduce failure rate compared to other state-of-the-art algorithms;
2. Achieve comparable terrain coverage compared to other state-of-the-art algorithms;
3. Respect swarm robotics design principles;
4. Test real-world applicability with experiments on physical robots;

The algorithm DORA-Explorer, presented in chapter 4, indeed managed to reduce significantly the rate of failures of a swarm exploring a hazardous environment. The results obtained and presented in section 4.5.1 clearly show that DORA-Explorer exhibit a higher tolerance to faults in the presence of risk when compared with a random walk and FBE algorithms. Additionally, the exploration rate of DORA-Explorer is considerably higher than the one of the random walk algorithm and is on par with the FBE, a state-of-the-art exploration algorithm. DORA-Explorer respects the design principles of swarm robotics: The algorithm runs in a fully decentralized fashion with no central coordination and without predefined roles. All the robots are initialized equally and they share information through the virtual stigmergy [17], a mechanism built into the Buzz programming language [9] and specifically designed for robot swarms. The behaviors of the agents are achieved through local interactions, namely the virtual stigmergy, and doesn't require a high computational platform to be executed. Finally, DORA-Explorer was tested on a swarm of three physical robots which showed its real-world applicability. In summary, DORA-Explorer is meant to be used by swarms of robots covering open and unknown environments containing hazards that can be

sensed. In this specific scenario, the algorithm should provide good coverage results while reducing considerably the likelihood of robot failures.

6.2 Risk-Aware storage and routing

Specific to RASS, the research objectives were the following:

1. Reduce data corruption rate compared to other state-of-the-art algorithms;
2. Achieve comparable transfer speed compared to other state-of-the-art algorithms;
3. Respect swarm robotics design principles from section;
4. Test real-world applicability with experiments on physical robots;

Similarly to DORA-Explorer, RASS met its research objectives by offering a decentralized risk-aware storing and routing algorithm that effectively reduces the likelihood of data corruption in the system. Indeed, when compared with a standard hop-count based algorithm results show that RASS always outperforms its counterpart in the presence of risk. However, the higher reliability comes at the cost of slower transfer speeds, RASS takes on average 54% more time to send data items for permanent storage. Fortunately, it is possible to tune the importance of transfer speed by increasing the weight of the routing gain α in equation 5.3, giving flexibility to the operator of the swarm system to decide the desired behavior from RASS. Again, all swarm robotics design principles are respected and experiments were carried on physical robots to validate the real-world applicability of the algorithm. Overall, both DORA-Explorer and RASS were proven to be valuable assets for robot swarms exploring hazardous environments. Of course, more experiments with larger real-world swarm systems will need to be carried to corroborate the results obtained in simulation with greater swarms. Again, RASS is meant to be used by swarms of robots exploring an unknown and dangerous environment and collecting information about it. RASS can be used alongside DORA-Explorer and should provide safer routing and storage for the data items being collected and forwarded to the base station.

CHAPTER 7 CONCLUSION

The work presented in the thesis aimed at improving the robustness of swarm robotic systems in the presence of risk. Results show that by introducing a conscience of risk in the algorithms developed, a better tolerance to danger was achieved by the swarm in comparison with our benchmarks. The thesis presented two risk-aware swarm algorithms used for the exploration of unknown environments:

- DORA-Explorer: Distributed Online Risk-Aware Explorer
- RASS: Risk-Aware Swarm Storage

Both algorithms achieved convincing results and are a contribution to risk tolerance in the field of swarm robotics.

7.1 Summary of Works

We first presented DORA-Explorer, a risk-aware exploration algorithm that minimizes the risk to which robots expose themselves in order to maximize the amount of ground they will be able to cover. Leveraging distributed belief maps, DORA-Explorer greatly outperformed our benchmark algorithms in terms of exposure to risk. Results showed that our solution considerably reduces the likeliness of robot failures while keeping similar ground coverage performance compared to other solutions proposed in the literature. DORA-Explorer also showed good scalability thanks to its low communication costs and its fully decentralized nature. It also showed applicability to real-world scenarios through experiments with physical robots.

Then, RASS, a fully decentralized risk-aware storing and routing algorithm, was presented. It leverages a fitness policy based on hop-count and risk to choose the fittest agents through which to route data items towards the base station. Our experiments showed that RASS consistently outperforms a hop-count based algorithm in terms of reliability across different topologies, both static and dynamic. RASS was also tested with physical robots and again showed convincing results. The physical experiments proved that the algorithm can easily be used on real world missions and will run the way it is intended to.

7.2 Limitations

In DORA-Explorer, simulations with varying ratios of risk gain α to exploration gain β were conducted and showed that they have an impact on the performance of the algorithm. In our simulations, a ratio of $\alpha/\beta = 2$ provided the best results. Luckily, because we had a simulated environment, we could easily test with different values and choose the best one. However, testing with different values may not always be feasible and as a result, determining the right set of parameters for the algorithm to perform optimally is not completely solved.

Additionally, the physical experiments performed to assess the real-world applicability of DORA-Explorer and RASS were carried on very small robot swarms. For DORA-Explorer, only three KheperaIV robots were used and for RASS five CogniFlies. Consequently, the scalability of the algorithms remains untested on real hardware.

7.3 Future Research

Specifically for DORA-Explorer, taking inspiration from obstacle avoidance algorithms could be an interesting future research direction. We could, in DORA-Explorer's distributed belief maps, model the risk associated with the cells as a probability of containing an obstacle. Using state-of-the-art obstacle avoidance algorithms, we could potentially avoid dangerous regions of the environment the same way you would avoid obstacles in the environment. However, some challenges remain in that regard. Risk is considered to be gradual, in opposition, obstacles are punctual. Obstacle avoidance algorithms try to guarantee to avoidance of obstacles. On the other hand, DORA-Explorer does not guarantee the avoidance of risk but instead tries to minimize exposure to it. Therefore, using obstacle avoidance algorithms would most certainly require some design changes.

Specifically for RASS, some additional work includes understanding the impact of the communication network assumptions on the performance of RASS. For example, a network where there exists only a few routes towards the base station and where one is clearly safer than the other is, of course, well suited for our algorithm. On the other hand, the value of RASS should decrease in well-connected networks where the radiation sources are only located at the periphery of the network. Indeed, in this particular scenario, because data naturally moves away from the risk, RASS' risk-awareness shouldn't improve noticeably the network's reliability. Additional experiments with different network topologies would be an interesting future research direction and could provide insights on which topologies RASS is particularly valuable.

For both algorithms, additional experiments on real outdoor missions and with actual risk

would be useful in determining their value. In both our simulations and physical experiments, risk in the environment was simulated and took the form of radiation. It would be worth testing the algorithms with real danger and confirm that the results are in line with what was seen in our simulations. However, we are aware that such experiments are costly and difficult to realize.

Finally, another potential future research direction would be to tackle the creation of the distributed belief map in a scenario where risk cannot be sensed by a sensor. Indeed, in both DORA-Explorer and RASS risk was modelled as point radiation sources that, we suppose, could by sensed by robots with an on-board sensor. However, in some scenario risk cannot be directly sensed. In that case, building the distributed belief map becomes problematic. A potential way to solve this would be the use fault detection methods to identify the hazardous locations of the environment. By knowing where failures have happened in the past, greater risk could be associated with regions where failures seem to happen more frequently.

REFERENCES

[1] W. Burgard *et al.*, "Coordinated multi-robot exploration," *IEEE Transactions on robotics*, vol. 21, no. 3, pp. 376–386, 2005. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1435481

[2] J. D. Bjerknes and A. F. T. Winfield, "On Fault Tolerance and Scalability of Swarm Robotic Systems," in *Distributed Autonomous Robotic Systems: The 10th International Symposium*, A. Martinoli *et al.*, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 431–444. [Online]. Available: https://doi.org/10.1007/978-3-642-32723-0_31

[3] A. Prorok *et al.*, "Beyond robustness: A taxonomy of approaches towards resilient multi-robot systems," *arXiv preprint arXiv:2109.12343*, 2021.

[4] M. Dorigo and M. Birattari, "Swarm intelligence," *Scholarpedia*, vol. 2, no. 9, p. 1462, 2007, revision #138640.

[5] A. Desai *et al.*, "Drona: a framework for safe distributed mobile robotics," in *Proceedings of the 8th International Conference on Cyber-Physical Systems*, 2017, pp. 239–248.

[6] M. Quigley *et al.*, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[7] Y. Cao *et al.*, "Distributed tdma for mobile uwb network localization," *IEEE Internet of Things Journal*, vol. 8, no. 17, pp. 13 449–13 464, 2021.

[8] R. Ghosh *et al.*, "Koord: a language for programming and verifying distributed robotics application," *Proceedings of the ACM on Programming Languages*, vol. 4, no. OOPSLA, pp. 1–30, 2020.

[9] C. Pinciroli and G. Beltrame, "Buzz: A programming language for robot swarms," *IEEE Software*, vol. 33, no. 4, pp. 97–100, 2016.

[10] D. St-Onge *et al.*, "Ros and buzz: consensus-based behaviors for heterogeneous teams," *arXiv preprint arXiv:1710.08843*, 2017.

[11] C. Pinciroli *et al.*, "ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems," *Swarm Intelligence*, vol. 6, no. 4, pp. 271–295, 2012.

[12] F. Amigoni, J. Banfi, and N. Basilico, "Multirobot exploration of communication-restricted environments: A survey," *IEEE Intelligent Systems*, vol. 32, no. 6, pp. 48–57, 2017. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8267592

[13] M. Otte, "An emergent group mind across a swarm of robots: Collective cognition and distributed sensing via a shared wireless neural network," *The International Journal of Robotics Research*, vol. 37, no. 9, pp. 1017–1061, 2018. [Online]. Available: https://doi.org/10.1177/0278364918779704

[14] E. Bonabeau *et al.*, *Swarm intelligence: from natural to artificial systems*. Oxford university press, 1999, no. 1.

[15] F. Heylighen, "Stigmergy as a universal coordination mechanism i: Definition and components," *Cognitive Systems Research*, vol. 38, pp. 4–13, 2016.

[16] ——, "Stigmergy as a universal coordination mechanism ii: Varieties and evolution," *Cognitive Systems Research*, vol. 38, pp. 50–59, 2016.

[17] C. Pinciroli, A. Lee-Brown, and G. Beltrame, "A tuple space for data sharing in robot swarms," in *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*, 2016, pp. 287–294. [Online]. Available: https://carlo.pinciroli.net/pdf/Pinciroli:BICT2015.pdf

[18] V. S. Varadharajan *et al.*, "Soul: data sharing for robot swarms," *Autonomous Robots*, vol. 44, no. 3, pp. 377–394, 2020.

[19] N. Majcherczyk and C. Pinciroli, "Swarmmesh: A distributed data structure for cooperative multi-robot applications," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 4059–4065. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9197403

[20] C. Stachniss and W. Burgard, "Mapping and exploration with mobile robots using coverage maps," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, vol. 1. Las Vegas, Nevada, USA: IEEE, 2003, pp. 467–472.

[21] F. Kobayashi, S. Sakai, and F. Kojima, "Sharing of exploring information using belief measure for multi robot exploration," in *2002 IEEE World Congress on Computational Intelligence. 2002 IEEE International Conference on Fuzzy Systems. FUZZ-IEEE'02. Proceedings (Cat. No.02CH37291)*, vol. 2, May 2002, pp. 1544–1549 vol.2.

[22] ——, "Determination of exploration target based on belief measure in multi-robot exploration," in *Proceedings 2003 IEEE International Symposium on Computational Intelligence in Robotics and Automation. Computational Intelligence in Robotics and Automation for the New Millennium (Cat. No.03EX694)*, vol. 3, Jul. 2003, pp. 1545–1550 vol.3.

[23] V. Indelman, "Cooperative multi-robot belief space planning for autonomous navigation in unknown environments," *Autonomous Robots*, vol. 42, no. 2, pp. 353–373, Feb. 2018.

[24] L. Han *et al.*, "Grid-wise control for multi-agent reinforcement learning in video game AI," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 2576–2585. [Online]. Available: http://proceedings.mlr.press/v97/han19a.html

[25] A. I. Panov, K. S. Yakovlev, and R. Suvorov, "Grid Path Planning with Deep Reinforcement Learning: Preliminary Results," *Procedia Computer Science*, vol. 123, pp. 347–353, 2018.

[26] J. Garcia and F. Fernandez, "Safe Exploration of State and Action Spaces in Reinforcement Learning," *Journal of Artificial Intelligence Research*, vol. 45, pp. 515–564, Dec. 2012. [Online]. Available: https://doi.org/10.1613/jair.3761

[27] P.-A. Andersen, M. Goodwin, and O.-C. Granmo, "Towards safe reinforcement-learning in industrial grid-warehousing," *Information Sciences*, vol. 537, pp. 467–484, Oct. 2020. [Online]. Available: https://doi.org/10.1016/j.ins.2020.06.010

[28] J. Faruque, K. Psounis, and A. Helmy, "Analysis of gradient-based routing protocols in sensor networks," in *International Conference on Distributed Computing in Sensor Systems*. Springer, 2005, pp. 258–275.

[29] R. Draves, J. Padhye, and B. Zill, "Comparison of routing metrics for static multi-hop wireless networks," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 4, pp. 133–144, 2004. [Online]. Available: https://doi.org/10.1145/1030194.1015483

[30] T. Watteyne *et al.*, "Implementation of gradient routing in wireless sensor networks," in *GLOBECOM 2009-2009 IEEE Global Telecommunications Conference*. IEEE, 2009, pp. 1–6. [Online]. Available: https://doi.org/10.1109/GLOCOM.2009.5425543

[31] J. Kuruvila, A. Nayak, and I. Stojmenovic, "Hop count optimal position-based packet routing algorithms for ad hoc wireless networks with a realistic physical layer," *IEEE*

Journal on selected areas in communications, vol. 23, no. 6, pp. 1267–1275, 2005. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1435519

[32] X. Zhang *et al.*, "An efficient hop count routing protocol for wireless ad hoc networks," *International Journal of automation and computing*, vol. 11, no. 1, pp. 93–99, 2014. [Online]. Available: https://link.springer.com/content/pdf/10.1007/s11633-014-0770-0.pdf

[33] F. Al-Salti *et al.*, "An efficient and reliable grid-based routing protocol for uwsns by exploiting minimum hop count," *Computer Networks*, vol. 162, p. 106869, 2019. [Online]. Available: https://doi.org/10.1016/j.comnet.2019.106869

[34] C. Tripp-Barba *et al.*, "Survey on routing protocols for vehicular ad hoc networks based on multimetrics," *Electronics*, vol. 8, no. 10, p. 1177, 2019.

[35] S. Boussoufa-Lahlah, F. Semchedine, and L. Bouallouche-Medjkoune, "Geographic routing protocols for vehicular ad hoc networks (vanets): A survey," *Vehicular Communications*, vol. 11, pp. 20–31, 2018.

[36] H. Füßler *et al.*, "A comparison of routing strategies for vehicular ad hoc networks," *Technical reports*, vol. 2, 2002.

[37] K. Li *et al.*, "Slime mold inspired routing protocols for wireless sensor networks," *Swarm Intelligence*, vol. 5, no. 3, pp. 183–223, 2011. [Online]. Available: https://doi.org/10.1007/s11721-011-0063-y

[38] N. Jiang *et al.*, "Toward biology-inspired solutions for routing problems of wireless sensor networks with mobile sink," *Soft Computing*, vol. 22, no. 23, pp. 7847–7855, 2018. [Online]. Available: https://doi.org/10.1007/s00500-018-3506-1

[39] A. Jiang and L. Zheng, "An effective hybrid routing algorithm in wsn: Ant colony optimization in combination with hop count minimization," *sensors*, vol. 18, no. 4, p. 1020, 2018. [Online]. Available: https://doi.org/10.3390/s18041020

[40] W.-H. Liao, Y. Kao, and C.-M. Fan, "Data aggregation in wireless sensor networks using ant colony algorithm," *Journal of Network and Computer Applications*, vol. 31, no. 4, pp. 387–401, 2008. [Online]. Available: https://doi.org/10.1016/j.jnca.2008.02.006

[41] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 26, no. 1, pp. 29–41, 1996.

[42] O. Arslan and D. E. Koditschek, "Voronoi-based coverage control of heterogeneous disk-shaped robots," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 4259–4266.

[43] W. Luo and K. Sycara, "Voronoi-based coverage control with connectivity maintenance for robotic sensor networks," in *2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*. IEEE, 2019, pp. 148–154. [Online]. Available: https://ieeexplore.ieee.org/document/8901078

[44] M. Santos *et al.*, "Decentralized minimum-energy coverage control for time-varying density functions," in *2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*. IEEE, 2019, pp. 155–161. [Online]. Available: https://ieeexplore.ieee.org/document/8901076

[45] X. Xu and Y. Diaz-Mercado, "Multi-robot control using coverage over time-varying domains," in *2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*. IEEE, 2019, pp. 179–181. [Online]. Available: https://ieeexplore.ieee.org/document/8901067

[46] B. Yamauchi, "Frontier-based exploration using multiple robots," in *Proceedings of the second international conference on Autonomous agents*, 1998, pp. 47–53. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/280765.280773

[47] Y. Wang, A. Liang, and H. Guan, "Frontier-based multi-robot map exploration using particle swarm optimization," in *2011 IEEE symposium on Swarm intelligence*. IEEE, 2011, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/5952584

[48] A. Topiwala, P. Inani, and A. Kathpal, "Frontier based exploration for autonomous robot," *arXiv preprint arXiv:1806.03581*, 2018. [Online]. Available: https://arxiv.org/abs/1806.03581

[49] E. J. Robinson *et al.*, "'no entry'signal in ant foraging," *Nature*, vol. 438, no. 7067, pp. 442–442, 2005.

[50] E. R. Hunt, S. Jones, and S. Hauert, "Testing the limits of pheromone stigmergy in high-density robot swarms," *Royal Society open science*, vol. 6, no. 11, p. 190225, 2019. [Online]. Available: https://doi.org/10.1098/rsos.190225

[51] M. Salman *et al.*, "Phormica: Photochromic pheromone release and detection system for stigmergic coordination in robot swarms," *Frontiers in Robotics and AI*, vol. 7, p. 195, 2020.

[52] P. Dames *et al.*, "A decentralized control policy for adaptive information gathering in hazardous environments," in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*. IEEE, 2012, pp. 2807–2813. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6426239

[53] M. Schwager *et al.*, "A Multi-robot Control Policy for Information Gathering in the Presence of Unknown Hazards," in *Robotics Research : The 15th International Symposium ISRR*, ser. Springer Tracts in Advanced Robotics, H. I. Christensen and O. Khatib, Eds. Cham: Springer International Publishing, 2017, pp. 455–472.

[54] S. Jones *et al.*, "Distributed situational awareness in robot swarms," *Advanced Intelligent Systems*, vol. 2, no. 11, p. 2000110, 2020.

[55] E. R. Hunt and S. Hauert, "A checklist for safe robot swarms," *Nature Machine Intelligence*, vol. 2, no. 8, pp. 420–422, 2020.

[56] F. Higgins, A. Tomlinson, and K. M. Martin, "Threats to the swarm: Security considerations for swarm robotics," *International Journal on Advances in Security*, vol. 2, no. 2&3, 2009.

[57] A. Undurti and J. P. How, "An online algorithm for constrained pomdps," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 3966–3973. [Online]. Available: https://doi.org/10.1109/ROBOT.2010.5509743

[58] S. Thiébaux, B. Williams *et al.*, "Rao*: An algorithm for chance-constrained pomdp's," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/10423

[59] X. Xiao, J. Dufek, and R. R. Murphy, "Robot risk-awareness by formal risk reasoning and planning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2856–2863, 2020. [Online]. Available: https://doi.org/10.1109/LRA.2020.2974434

[60] A. Hakobyan, G. C. Kim, and I. Yang, "Risk-aware motion planning and control using cvar-constrained optimization," *IEEE Robotics and Automation letters*, vol. 4, no. 4, pp. 3924–3931, 2019.

[61] S. Samuelson and I. Yang, "Safety-aware optimal control of stochastic systems using conditional value-at-risk," in *2018 Annual American Control Conference (ACC)*. IEEE, 2018, pp. 6285–6290.

[62] E. R. Hunt, C. B. Cullen, and S. Hauert, "Value at risk strategies for robot swarms in hazardous environments," in *Unmanned Systems Technology XXIII*, vol. 11758. International Society for Optics and Photonics, 2021, p. 117580M.

[63] M. Ono and B. C. Williams, "An efficient motion planning algorithm for stochastic dynamic systems with constraints on probability of failure." in *AAAI*, 2008, pp. 1376–1382. [Online]. Available: http://hdl.handle.net/1721.1/40803

[64] M. P. Vitus and C. J. Tomlin, "On feedback design and risk allocation in chance constrained control," in *2011 50th IEEE Conference on Decision and Control and European Control Conference*. IEEE, 2011, pp. 734–739. [Online]. Available: https://doi.org/10.1109/CDC.2011.6160721

[65] E. R. Hunt *et al.*, "SPIDER: a bioinspired swarm algorithm for adaptive risk-taking," in *Artificial Life Conference Proceedings*. MIT Press, 2020, pp. 44–51.

[66] E. Khalastchi and M. Kalech, "On fault detection and diagnosis in robotic systems," *ACM Comput. Surv.*, vol. 51, no. 1, jan 2018. [Online]. Available: https://doi.org/10.1145/3146389

[67] V. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artificial intelligence review*, vol. 22, no. 2, pp. 85–126, 2004.

[68] A. L. Christensen, "Fault detection in autonomous robots," *Phd, Université Libre de Bruxelles*, 2008.

[69] H. K. Lau, "Error detection in swarm robotics: A focus on adaptivity to dynamic environments," Ph.D. dissertation, University of York, 2012.

[70] O. Graham Miller and V. Gandhi, "A survey of modern exogenous fault detection and diagnosis methods for swarm robotics," *Journal of King Saud University - Engineering Sciences*, vol. 33, no. 1, pp. 43–53, 2021.

[71] D. Tarapore *et al.*, "To err is robotic, to tolerate immunological: fault detection in multirobot systems," *Bioinspiration & biomimetics*, vol. 10, no. 1, p. 016014, 2015.

[72] D. Tarapore, A. L. Christensen, and J. Timmis, "Generic, scalable and decentralized fault detection for robot swarms," *PloS one*, vol. 12, no. 8, p. e0182058, 2017.

[73] D. Tarapore, J. Timmis, and A. L. Christensen, "Fault detection in a swarm of physical robots based on behavioral outlier detection," *IEEE Transactions on Robotics*, vol. 35, no. 6, pp. 1516–1522, 2019.

[74] J. O'Keeffe *et al.*, "Adaptive online fault diagnosis in autonomous robot swarms," *Frontiers in Robotics and AI*, vol. 5, p. 131, 2018.

[75] M. M. Breunig *et al.*, "Lof: identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 2000, pp. 93–104.

[76] A. L. Christensen *et al.*, "Fault detection in autonomous robots based on fault injection and learning," *Autonomous Robots*, vol. 24, no. 1, pp. 49–67, 2008.

[77] F. Mondada *et al.*, "The cooperation of swarm-bots: Physical interactions in collective robotics," *IEEE Robotics & Automation Magazine*, vol. 12, no. 2, pp. 21–28, 2005.

[78] G. Pang *et al.*, "Deep learning for anomaly detection: A review," *ACM Computing Surveys (CSUR)*, vol. 54, no. 2, pp. 1–38, 2021.

[79] A. Matos *et al.*, "Multiple robot operations for maritime search and rescue in eurathlon 2015 competition," in *OCEANS 2016-Shanghai*. IEEE, 2016, pp. 1–7. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7485707

[80] T. Fong and I. Nourbakhsh, "Interaction challenges in human-robot space exploration," *Interactions*, vol. 12, no. 2, pp. 42–45, 2005. [Online]. Available: https://dl.acm.org/doi/fullHtml/10.1145/1052438.1052462

[81] R. K. Ramachandran, J. A. Preiss, and G. S. Sukhatme, "Resilience by reconfiguration: Exploiting heterogeneity in robot teams," *arXiv preprint arXiv:1903.04856*, 2019. [Online]. Available: https://arxiv.org/abs/1903.04856

[82] R. Wehbe and R. K. Williams, "Probabilistic resilience of dynamic multi-robot systems," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1777–1784, 2021. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9357930

[83] A. F. Winfield and J. Nembrini, "Safety in numbers: fault-tolerance in robot swarms," *International Journal of Modelling, Identification and Control*, vol. 1, no. 1, pp. 30–37, 2006. [Online]. Available: https://www.inderscienceonline.com/doi/abs/10.1504/IJMIC.2006.008645

[84] M. Shahriari *et al.*, "Lightweight collision avoidance for resource-constrained robots," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–9. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8593841

[85] K-Team, "Khepera IV," 2021. [Online]. Available: https://www.k-team.com/khepera-iv

[86] S. Arseneault, D. Vielfaure, and G. Beltrame, "Rass: Risk-aware swarm storage," 2022.

[87] G. Kantor *et al.*, *Distributed Search and Rescue with Robot and Sensor Teams*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 529–538.

[88] R. Sharp and M. Decreton, "Radiation tolerance of components and materials in nuclear robot applications," *Reliability Engineering & System Safety*, vol. 53, no. 3, pp. 291–299, 1996. [Online]. Available: https://doi.org/10.1016/S0951-8320(96)00054-3

[89] G. C. Messenger and M. S. Ash, *The effects of radiation on electronic systems*. Van Nostrand Reinhold Co., 1986.

APPENDIX A DORA-EXPLORER: EXECUTION LOOP

Algorithm 2: DORA Execution Loop

$\boldsymbol{x} \leftarrow random\_coordinates$
while *True* **do**
  $\nabla_r, \nabla_e \longleftarrow (0,0), (0,0)$

  for $n \in \nu$ **do**
    $\nabla_r \leftarrow \nabla_r + (r\_stig[\boldsymbol{x}] - r\_stig[\boldsymbol{n}]) \cdot normalize(\boldsymbol{n})$
    $\nabla_e \leftarrow \nabla_e + (e\_stig[\boldsymbol{x}] - e\_stig[\boldsymbol{n}]) \cdot normalize(\boldsymbol{n})$
  end

  $\boldsymbol{m} \leftarrow \alpha \cdot \nabla_r + \beta \cdot \nabla_e + \gamma \cdot compute\_avoidance(sensors)$
  $\boldsymbol{x} \leftarrow \boldsymbol{x} + k \cdot normalize(\boldsymbol{m})$
  $r\_stig[\boldsymbol{x}], e\_stig[\boldsymbol{x}] \leftarrow get\_radiation(), time()$
end

APPENDIX B DORA-EXPLORER: RISK-AWARE EXPLORATION

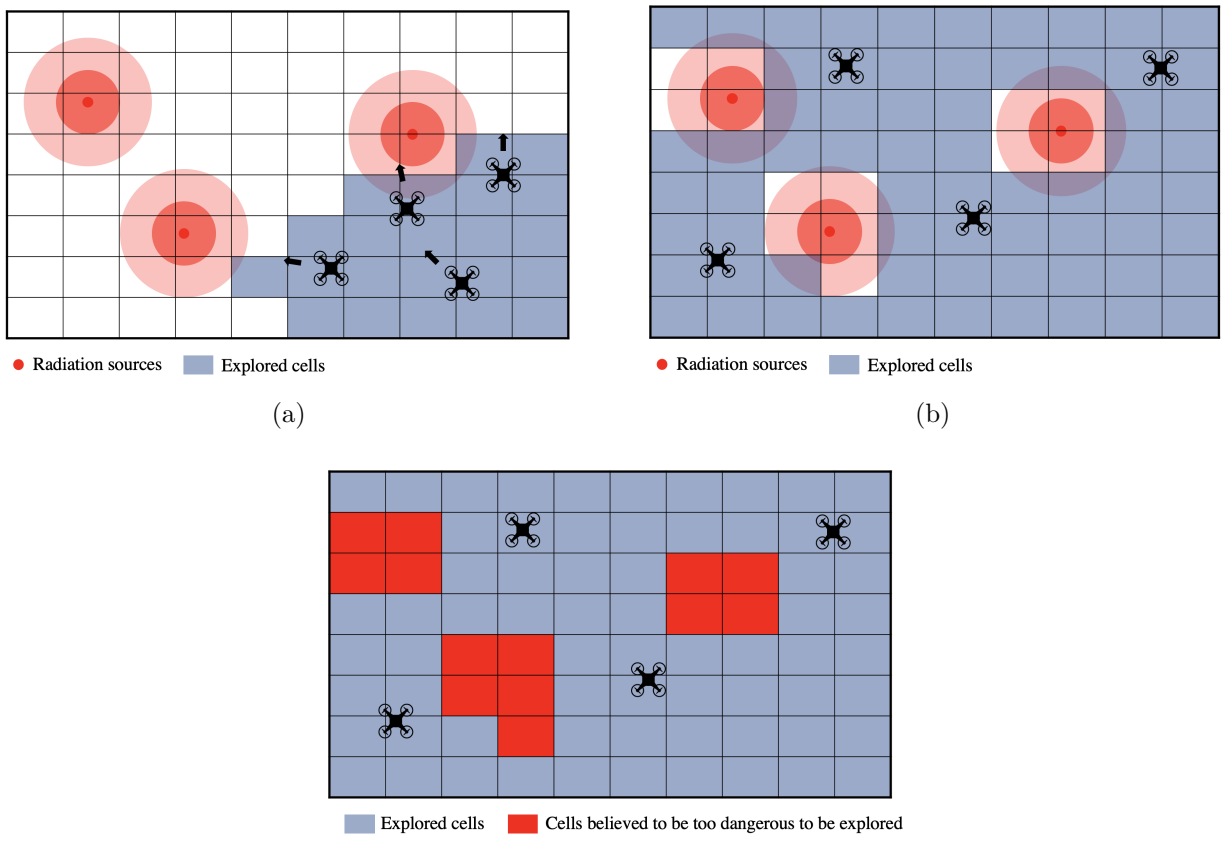


Figure B.1 Risk aware exploration intuition. Fig. B.1(a): Robots start exploring a hazardous environment. When a new cell is explored, the sensed radiation is used to update the DBM. Fig. B.1(b): The cells have been mostly covered by the robots. Fig. B.1(c): Only cells believed to be too dangerous remain unexplored.

APPENDIX C DORA-EXPLORER: ARGOS SIMULATED ENVIRONMENT

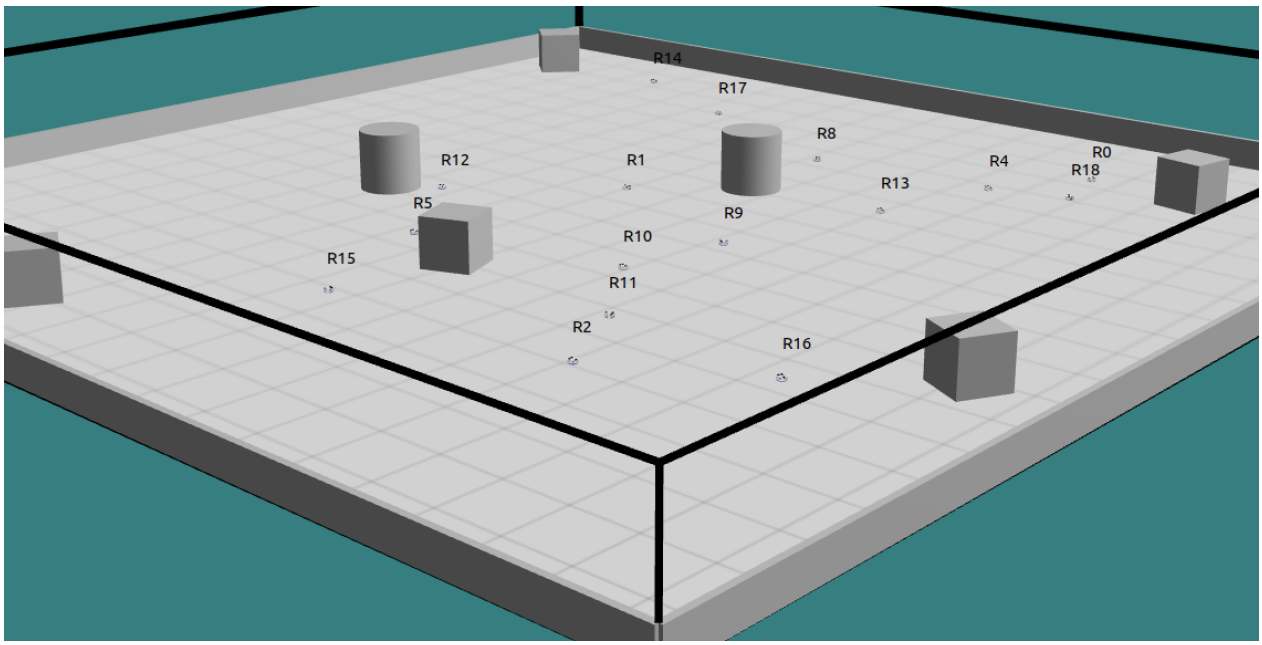


Figure C.1 400m^2 environment in the ARGoS simulator with 20 KheperaIV robots. Cylinders are radiation sources and boxes are random obstacles.

APPENDIX D DORA-EXPLORER: RESULTS OVER TIME

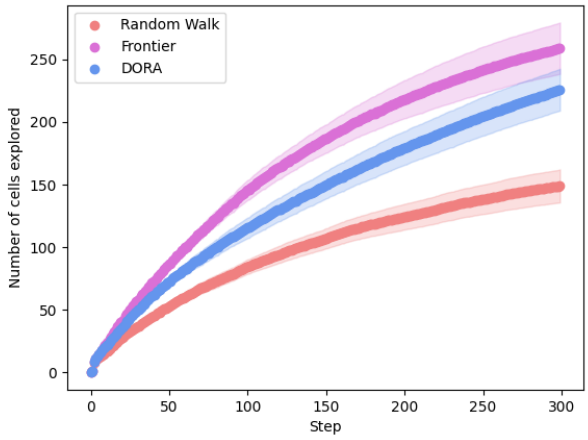


Figure D.1 Performance comparison of DORA, FBE and random walk for number of explored cells over time, with N=10 robots.

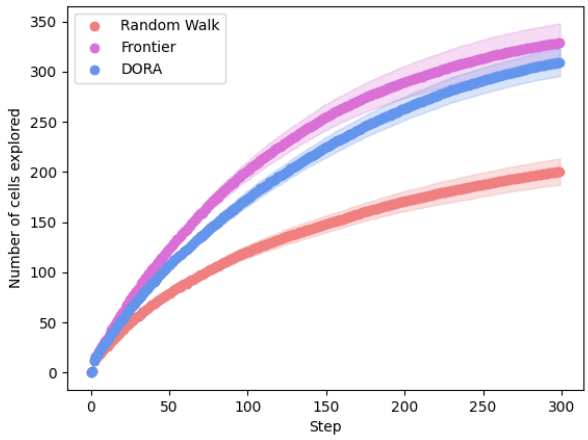


Figure D.2 Performance comparison of DORA, FBE and random walk for number of explored cells over time, with N=15 robots.

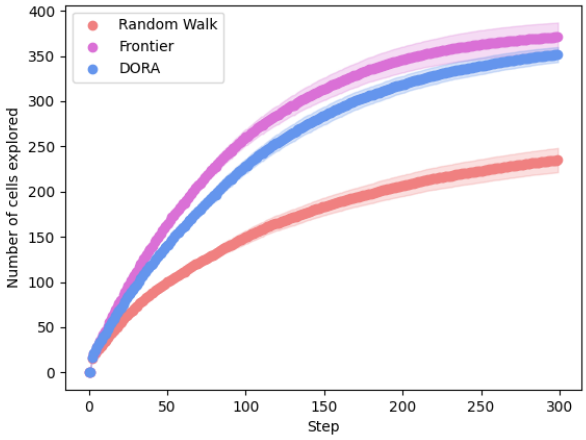


Figure D.3 Performance comparison of DORA, FBE and random walk for number of explored cells over time, with N=20 robots.

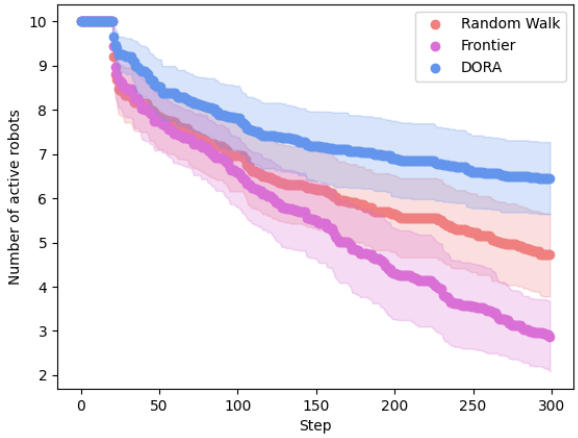


Figure D.4 Performance comparison of DORA, FBE and random walk for number of active robots over time, with N=10 robots.

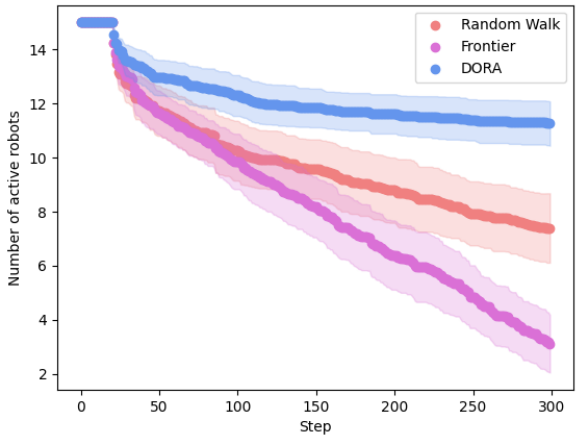


Figure D.5 Performance comparison of DORA, FBE and random walk for number of active robots over time, with N=15 robots.

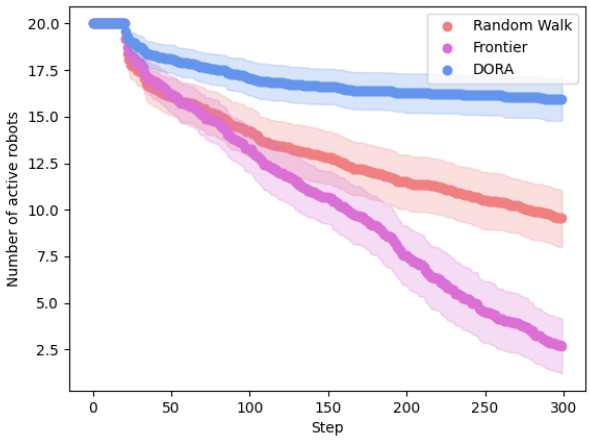


Figure D.6 Performance comparison of DORA, FBE and random walk for number of active robots over time, with N=20 robots.

APPENDIX E DORA-EXPLORER: PHYSICAL EXPERIMENTS

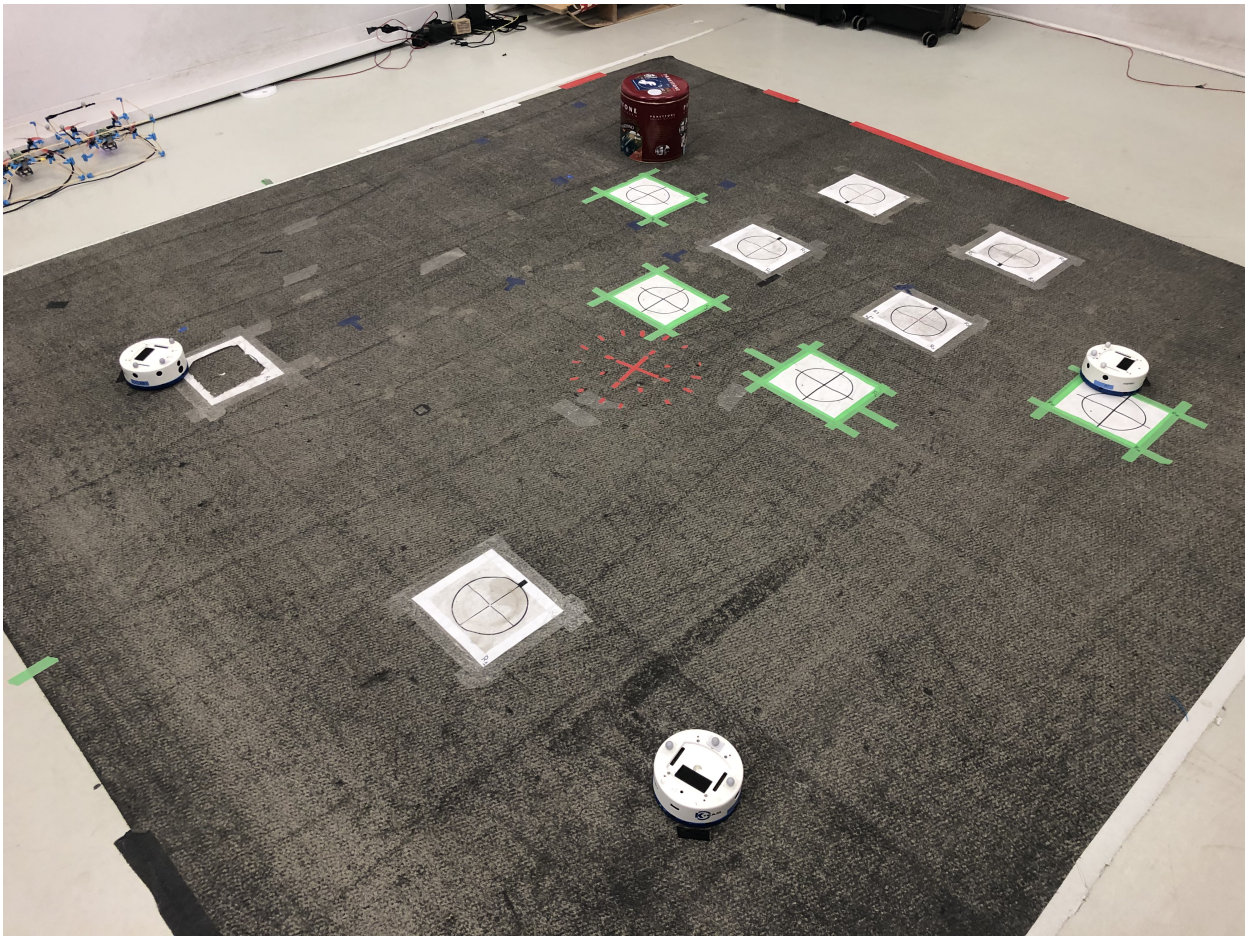


Figure E.1 Experiments on three physical KheperaIV robots. The red canister represents the point radiation source in the environment.