Coverage Path Planning for Autonomous Robots

Zeba Khanam

School of Computer Science and Electronic Engineering
University of Essex

The thesis is submitted for the degree of
Doctor of Philosophy

September 2022

To
Mamma, Abbu and Maimoona,
for standing through thick and thin
&
my grandfather,
Late (Dr.) Mumtaz Ali Khan,
for being an inspiration to undertake this beautiful journey.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

Zeba Khanam
September 2022

Acknowledgements

'A journey of a thousand miles begins with a single step.' My PhD is no exception. It started with a small conversation that I had with my father in Grade 8 which ignited this dream. There are many dreams you see but only few become reality. Many actors and situations play a major role. Therefore, this dream cannot be realised without giving due credits and acknowledgment. Prime lesson I have learnt in this study is *'Success is a mark of privilege'*. The biggest privilege and blessing of my life is my connection with Almighty. No word in dictionary can do justice to express my gratitude towards Him. *'All praise is due to Allah, Lord of the worlds - Quran 1:2'*. I can only hope on the day of judgment when I will stand in front of him I can justify that I was able to make the best out of this opportunity.

This thesis will always be incomplete without expressing gratitude to my supervisors Prof. Klaus McDonald-Maier and Dr. Shoaib Ehsan for providing me with excellent guidance. I can clearly remember I had seen an advert on CVPR website and dropped an email to them expressing my interest to join the lab. From that day to this date, they have been nothing but supportive and accommodating. Their mentorship role was not only restricted to my thesis specifically but extended to overall growth of my professional career. I am also grateful to my collaborators Dr. Dimitri Ognibene and Prof. Rustom Stolkin for providing me with useful feedback on my research. I have learnt a lot from our discussions.

I am grateful to my parents who have been a support system through out my life. They instilled love for education in my heart in early days of my life and they believed in me when even I did not believe in myself. I can only pray to Allah I can continue to make them proud and live according to the values they have taught me. My younger sister whose love, wisdom and affection is unparalleled. I had hit rock bottom many times during this thesis. It was my family's support which always gave me courage to start again.

This journey would not have been as engaging and enjoyable without my lab mates at Essex. A big shout out to my colleagues, specifically to Mubariz Zaffar, Ahmad Khaliq, Bilal Aslam, Server Kasap, Eduardo Watcher, and Sangeet Saha for making my thesis memorable. Last but not the least, I am grateful to the University of Essex for providing me a conducive environment to carry out my studies. This dream wouldn't have materialized without the generous support that was provided through University of Essex Doctoral Scholarship.

Abstract

Coverage Path Planning (CPP) is a problem of path computation with minimal length that guarantees to scan the entire area of interest. CPP finds its application in diverse fields like cartography, inspection, precision agriculture, milling, and demining. However, this thesis is a prominent step to solve CPP for real-world problems where environment poses multiple challenges. At first, four significant and pressing challenges for CPP in extreme environment are identified. Each challenge is formulated as a problem and its solution has been presented as a dedicated chapter in this thesis. The first problem, Goal-Oriented Sensor based CPP, focuses on cumbersome tasks like Nuclear Decommissioning, where the robot covers an abandoned site in tandem with the goal to reach a static target in minimal time. To meet the grave speeding-up challenge, a novel offline-online strategy is proposed that efficiently models the site using floor plans and grid maps as a priori information. The proposed strategy outperforms the two baseline approaches with reduction in coverage time by $45\% - 82\%$. The second problem explores CPP of distributed regions, applicable in post-disaster scenarios like *Fukushima Daiichi*. Experiments are conducted at radiation laboratory to identify the constraints robot would be subjected to. The thesis is successfully able to diagnose transient damage in the robot's sensor after 3 Gy of gamma radiation exposure. Therefore, a region order travel constraint known as *Precedence Provision* is imposed for successful coverage. The region order constraint allows the coverage length to be minimised by 65% in comparison to state-of-the-art techniques. The third problem identifies the major bottleneck of limited on-board energy that inhibits complete coverage of distributed regions. The existing approaches allow robots to undertake multiple tours for complete coverage which is impractical in many scenarios. To this end, a novel algorithm is proposed that solves a variant of CPP where the robot aims to achieve near-optimal area coverage due to path length limitation caused by the energy constraint. The proposed algorithm covers $23\% - 35\%$ more area in comparison to the state-of-the-art approaches. Finally, the last problem, an extension of the second and third problems, deals with the problem of CPP over a set of disjoint regions using a fleet of heterogeneous aerial robots. A heuristic is proposed to deliver solutions within acceptable time limits. The experiments demonstrate that proposed heuristic solution reduces the energy cost by $15 - 40\%$ in comparison to the state-of-the art solutions.

Contents

List of Figures

List of Tables

Abbreviations

2D	*Two – Dimensional*
3D	*Three – Dimensional*
B&B	*Branch – and – Bound*
CARMA	*Continuous Autonomous Radiometric Monitoring Assistant*
COPs	*Constraint Optimisation Problems*
COVID	*Corona Virus Disease*
CP	*Constraint Programming*
CPP	*Coverage Path Planning*
CSP	*Covering Salesman Problem*
CSPs	*Constraint Satisfaction Problems*
DD	*Displacement Damage*
DFJ	*Dantzig Fulkerson Johnson*
ILP	*Integer Linear Programming*
MILP	*Mixed Integer Linear Programming*
MTZ	*Miller Tucker Zemlin*
NP	*Non – deterministic Polynomial Time*
OSP	*Optimal Search Path*
SEE	*Single Event Effect*

SLAM	*Simultaneous Localization and Mapping*
TSP	*Traveling Salesman Problem*
UAV	*Unmanned Aerial Vehicle*

Chapter 1

Introduction

1.1 Introduction

The world is in the throes of technological revolution and *robots* have taken a center stage. Since its inception, robotics has managed to emerge from a niche technology to a new dominant socio-technical system. As predicted by a man way ahead of his time, Nikola Tesla, *"In the twenty-first century, the robot will take the place which slave labor occupied in ancient civilization"* [1], autonomous robots have became *part and parcel* of our current sophisticated society. Autonomous robots are already playing a vital role in boosting of the economy during current COVID-19 pandemic and are stated to be a game changer in post-COVID transformation of industries like health care, commerce, manufacturing, transportation and agriculture.

The idea of automatic system can be traced to the texts dating to 9^{th} century in Greek and Arab civilisations. However, *Industrial Revolution* marked a new era in the field of automation by generating a global demand of fast paced production. This demand fuelled the enormous success of autonomous robotics. The first few prototypes of Robotics and Automation Systems (RAS) were constrained to development of robots for precision tasks like welding [2]. Due to their high production cost, these robots were restricted to research institutes like Stanford or large firms like Ford and Unimate [3].

The first large scale deployment of RAS was by a pioneering robotics firm, ABB, in 1978. Interestingly, ABB's parent company ASEA had built successfully nuclear power plant in 1972 and ABB's main focus was deployment of robot's in capital intensive projects like Nuclear Power Plants [3]. Even the though research had focused on application of robots in diverse sectors like fashion, health care, travel, space exploration. But, the commercial deployment of robots was constrained to sectors related to industrial and manufacturing [4]. However, recent advances in digital and hardware technologies have ushered humans in

fourth industrial revolution (INDUSTRY 4.0). INDUSTRY 4.0 has propelled robots to work at the intersection of digital, physical, and biological spheres [5]. Nowadays, RAS is a fusion of an array of technologies like artificial intelligence (AI), the Internet of Things (IoT), genetic engineering, 3D printing, quantum computing. These technical capabilities has seen emergence of new processes like lights out manufacturing [6]. Even the commercial deployment has diversified with service robots like Pepper, Dyson360, Keenon becoming a part of industries like domestic services, health care and catering [7].

Despite the enormous success which autonomous robotics has achieved in last few decades, there is a plethora serious challenges like trust, safety compliance, carbon footprint reduction, resilience, scalability that confront this inter-disciplinary field. However, this thesis attempts to solve the most significant challenges in the sub-field of *motion planning*.

1.2 Motion Planning

Motion planning or path planning is a computational problem of finding a sequence of way-points that a robot needs to traverse in order to accomplish the task of reaching its goal from the current position. From its prototypical mapping as a piano mover's problem, path planning has advanced to address diverse range of variations on the problem, finding applications in domains like surgical planning, drug design, animation cartoons, SLAM, and assembly sequencing [8].

Every new real-world problem imposes a set of constraints and optimized objectives. For instance, some problem may demand the decision maker to minimize the total distance robot travels or the threat exposure during the robot's journey, search for an evasive object during the path traversal, map an unknown environment or update the existing map of the operating environment. There are scenarios where the decision makers have a priori knowledge of the start and goal location but in many cases, end goal is not known due to non-deterministic nature of the environment. However, what attracts the interest of research community is that the path traversal in real world is subjected to physical laws, uncertainty, and geometric constraints.

This gives rise to an array of questions which trace their origin in diverse fields like control theory, mechanics, computational and differential geometry, and computer science. Therefore, the impact of path planning is not only restricted to its utility in application but extends to the development of our science and math base by posing fundamental theoretical questions which otherwise would not have been asked. The author's research interest and scope of this thesis is addressing the theory and practice around Coverage Path Planning

(CPP) [9], which is a family of path planning problems, with an eye toward contemporary applications.

To focus on the discussion, and to point out some of the important concepts in coverage path planning, this chapter introduces CPP problem. The various challenges within CPP, applications and theory behind CPP are presented here.

1.3 Coverage Path Planning (CPP)

Coverage Path Planning (CPP) problem can be defined as the computational task of determining minimal length path which allows the robot to map/scan/cover the given area or volume while avoiding obstacles. This computational task is integral to diverse robotic applications, for instance floor cleaning [10–12], spray painting [13–15], underwater habitat exploration [16–18], demining [19–21], lawn mowers [22, 23], precision agriculture [24–26] and window cleaning [27, 12]. Fig. 1.1 visually represents common area coverage problems for which robots are deployed.

The pioneering work [29] on CPP laid down the ground rules to that the robot must adhere to while performing the task of area coverage. The aforementioned work assumes that a mobile robot is operating in a 2D environment. This criteria can also be extended to an aerial robot navigating at a fixed altitude. The rules are as follows:

1. After traversing all the waypoints, robot must cover the target area completely.
2. While covering the region, paths must not overlap.
3. The coverage operation must be continuous and sequential without any repetitive paths.
4. In case the target area contains obstacles, coverage path must avoid all the obstacles.
5. The robot must execute simple motion trajectories (e.g., straight lines or circles) to ensure simplistic control design.
6. Coverage path planner should output an “optimal" path within the given constraints.

However, in the real-world coverage operations, due to complexity of environment, it is not possible to satisfy all the rules stated above. Therefore, the decision makers are required to design coverage path planner based on priorities and trade-offs.

CPP problem is mapped to the Covering Salesman Problem (CSP) [30], a variant of the famous Traveling Salesman Problem (TSP) [31]. Unlike TSP, where the salesman visits each city with minimal tour length, CSP computes minimal tour length, where salesman must visit

Fig. 1.1 Inforgraphic illustrating common applications of CPP [28].

the neighbourhood of each city. However, in CPP, the robot must visit a set of waypoints in the target area. When traversing a waypoint, the area covered by robot's sensor footprint is mapped, thus by visiting all the waypoints sequentially the entire target area is mapped by the agent. Both TSP and CSP are NP-hard and suffer from the curse of dimensionality, i.e., the time required to generate solution to the problem increases drastically as the dimension of the problem increases.

One of the early applications of CPP, the "lawnmover problem", where the path planner computes the coverage path to cut all the grass in the target region ('lawn') has proven to be NP-hard [32]. However, it is worth noting that the lawnmower problem does not consider obstacles. Even the "piano mover's problem", which computes collision free path from current position to the goal is proven to be NP-hard [33, 34]. Two additional computational geometry NP-hard problems similar to CPP, are the *art gallery problem* and the *watchman route problem*. The art gallery problem is a visibility problem that computes the minimum number of guards required to guard and observe a polygonal gallery [35]. This problem corresponds to multi-robot CPP problem. On the other hand, the watchman problem is an optimization problem that computes the shortest route a watchman must take to guard a target area with obstacles [36]. The NP-hard nature of the CPP, has made the proposed heuristic solution highly dependent on the application. In the next subsection, the set of constraints and challenges that the robot will face while covering challenging environments are discussed.

1.4 Problem Statement and Challenges

Real-word applications of CPP pose prolific research challenges. This subsection identifies some of the crucial challenges, which have also been addressed in the later chapters of this thesis.

1.4.1 Goal-Oriented Coverage Path Planning

The baseline requirement for a CPP algorithm is to successfully map the entire target space. As discussed previously, CPP algorithms are often NP-hard in nature and the proposed heuristics are guided by the considered constraints and optimized objectives imposed on the path planner by the real-world applications. However, recent past has witnessed few peculiar cases, where the coverage is uncertain and the number of times an area needs to be visited varies [37, 38]. These cases intertwine the problem of CPP with search theory.

The first problem that is tackled, dives deep into area coverage in tandem with reaching a goal. This problem is known as *Goal-oriented Coverage Path Planning* and has arisen

recently, when robots have been deployed to perform daunting tasks like Nuclear Decommissioning, Oil and Gas Pipeline Inspection. This problem expects the robot to autonomously cover an abandoned environment with lurking dangers but at the same time achieve the goal to reach a static target in minimal time.

What makes Goal-oriented CPP interesting is that it combines two different problems of path planning: 1) CPP and 2) Optimal Search Path (OSP) problems which trace their origin to two different domains, robotics and operation research, respectively, with different formalisms. OSP at its roots aims to achieve the objective of maximizing the probability of finding the goal in the target area. The defined goal in the application mainly dictates the hypothesis of the decision maker. Some of recent goals in the OSP literature have been survivor, lost vessel and crashed plane [39–41]. At an abstract level, the goals can be classified based on following properties:

1. Mobility: static or mobile
2. Prior knowledge: a priori or unknown
3. Environment constraints on sensors and actuators
4. Cooperation nature of search object: adversarial or non-adversarial

The intended application of our Goal-oriented CPP allows us to limit the scope of our problem by characterising our goal, discussed in detail in Chapter 2 of this thesis.

1.4.2 Multiple Distributed Regions

The advances in the field of technology have ushered the deployment of Unmanned Aerial Vehicles (UAVs) for area coverage. UAV based CPP finds diverse applications, such as search and rescue, 3D mapping and land monitoring. Few applications expect UAV to visit multiple distributed regions introducing significant challenges in optimal path design. One of the major challenge introduced is additional computation of minimal length inter-region path covering a set of regions. Despite, the technological progress in the field of aerial robotics, area coverage of distributed regions is subjected to multiple constraints, due to limited resources and target area. The following subsections consider the constraints and provisions that an aerial robot will encounter, when trying to devise an optimal path for area coverage of distributed regions.

1.4.3 Extreme Environmental Conditions

The groundbreaking scientific discovery of nuclear energy generation has had far reaching consequences. At one end, it has permitted production of vast amount of energy using limited nuclear fuel. But at the same time, this generates tonnes of extremely hazardous nuclear waste. Recent accidents at such sites like the Fukushima Daiichi have resulted in a global demand for periodic inspection of sites with extreme environment. Human inspection of these sites is hazardous and requires a survey robot to perform this daunting task. One of the architectural norms followed in design of a nuclear facility is to isolate nuclear reactor cores from other service zones (control and service rooms, turbine building and cooling towers). Thus, the robot has to perform area coverage of multiple distributed regions. Apart from multiple distributed regions, prevailing environmental conditions, primarily presence of radiation, add to the complexity of CPP task as access to such sites for experimentation is difficult. Chapter 4 of thesis is dedicated to address this challenge of extreme environmental conditions during area coverage.

1.4.4 Energy Constraint

As discussed in previous section 1.4.2, area coverage of multiple distributed regions is a cumbersome task subjected to multiple constraints. One of the limitations which mobile robot suffers while covering distributed multiple regions is energy constraints. This problem intensifies for UAV due to much more limited on-board energy. Apart from multiple distributed coverage, even the extreme environmental conditions also reduce the lifetime of the robot. Due to the energy constraint, complete area coverage is not possible. Nearly all the prior works proposed in the literature have assumed that an aerial robot has sufficient on-board energy to perform the task of area coverage of distributed regions [42, 43]. Given the current energy models, aerial robots tend to have short battery life with an average of 20 minutes flight time, this consideration gains importance. Thus, one of the recent works has proposed multiple tours with stop over at onsite recharging stations [44]. This approach is not practical for many applications especially disaster management, where hard deadlines and multiple charging stations are not feasible. Chapter 5 of this thesis is dedicated to address the challenge of energy constraint during area coverage of multiple distributed regions.

1.4.5 Heterogeneous Fleet

Another set of important applications of distributed regions area coverage are post-disaster relief, military surveillance, search and rescue missions. These operations may often be con-

ducted over regions which are spatially separated with extreme terrains that are very difficult or completely unmanoeuvrable by terrestrial robots. Hence, such scenarios necessitate the deployment of UAVs for area coverage. Further many a time, these operations are carried out in critical emergency situations with stringent constraints on the available time within which the coverage must be accomplished. This generates a demand for deployment of all the resources at hand. Thus, a fleet of UAVs which are heterogeneous in terms of their technical specifications, like sensor footprint size and power rating to name a few, are entrusted to perform such missions. However, their diverse technical specifications pose a challenge for optimal coverage path design. Chapter 6 of thesis is dedicated to address this challenge of optimal coverage path for heterogeneous fleet of UAVs.

1.5 Thesis Contributions

The main contributions of this thesis are fourfold and centre around addressing the research challenges identified in section 1.4. These contributions are enlisted below.

1. The first contribution of this thesis is the proposal of a novel strategy to design an optimal coverage path with a static goal search in tandem while minimising coverage time. This framework, titled, 'Offline-Online', optimises the coverage time by equipping the robot with geometrically-awareness during coverage task. The efficacy of this strategy is evaluated by various experiments, against two baseline search approaches in three simulated environments.

2. The second contribution of this work is the coverage path design for area coverage in extreme environmental conditions. Extreme environmental conditions like presence of radiation triggers degradation of sensors in robot. Experiments were conducted at Rutherford Appleton Laboratory, UK to identify the provision needs that coverage path design must follow in order to successfully cover entire target site.

3. The third contribution of this thesis is a novel near optimal coverage path design for area coverage of multiple distributed regions given the energy constraint. The strategy maximises the coverage area for two inter-dependent coverage task by imposing a strict energy constraint. The performance of the coverage path strategy is evaluated by analysing its properties over an exhaustive set of test case scenarios and comparing it against two state-of-the-art area coverage approaches.

4. The fourth contribution of this work is a novel coverage strategy, which allows decision makers to deploy a fleet of heterogeneous aerial robots to perform area coverage

of distributed regions in critical emergency situations. A Simulated Annealing inspired heuristic solution solves a two-tier coverage problem of mapping heterogeneous resources to region and minimising energy cost. The results manifest a significant reduction in cost incurred in terms of energy to cover the complete target area compared to the state-of-the-art heuristic based on Genetic Algorithm.

1.6 Thesis Structure

The structural layout adopted in the remainder of this thesis is as follows:

Chapter 2: The general objective of this thesis is to consider contemporary research challenges in CPP. With this goal in mind, important concepts from combinatorial optimization are reviewed first. These concept are adopted later on as a solving tool in the thesis. To stay as close as possible to the theme of CPP, TSP is taken up as an example to illustrate optimization concepts on NP-hard problems. Finally, existing literature in the domain of CPP is reviewed, exploring the various taxonomies proposed in literature along with their positioning in regards to the contemporary research challenges.

Chapter 3: Chapter 3 explores the problem of goal-oriented CPP that traces its origin from two different set of problems, 1) CPP and 2) OSP. Both the problems require the deployed robot to perform area coverage. However, area coverage is a NP-hard problem and the varied objectives of CPP and OSP play a crucial role in optimal path design. The objective of CPP is to minimize the expense under the constraint of guaranteed complete area coverage. OSP tends to aim at maximizing the effectiveness of reaching the goal under the resource constraints. This chapter aims to study the combination of two problems through the lens of a proposed strategy, where deployed robot searches for the static goal, while mapping the environment along the path traversal.

Chapter 4: In this chapter, the focus is CPP problem in the context of deployment of UAV for inspection of an extreme environment site. The application specifically consists of designing optimal coverage path of a UAV for area coverage of a nuclear site. First, the underlying limitations that a commercial robot will face in extreme environment are identified through intensive experiments at radiation laboratories. In corroboration with underlying theme across gamma radiation and fast neutrons, the coverage path planner is fitted with a precedence provision to initiate an annealing process in robot's sensor. The chapter first formulates the problem mathematically and goes on to build the heuristic solution to this problem.

Chapter 5: In continuum, another interesting problem of area coverage of distributed regions is closely examined. An aerial robot while performing inspection of sites with

spatially distributed regions is subjected to multiple constraints. One of the major bottlenecks during aerial robot CPP is limited on-board energy. This problem intensifies, when an aerial robot needs to perform area coverage of distributed regions. In this chapter, CPP algorithm is designed where aerial robot is subjected to energy constraints, where complete area coverage is not often possible. The proposed algorithm treads a different route from existing approaches in literature by proposing a strategy of computing a near-optimal coverage path.

Chapter 6: This chapter investigates another challenge of area coverage of distributed regions. The CPP of distributed regions has been actively used in scenarios that are characterized as critical. Such scenarios, envisage a requirement to deploy all the available resources for the area coverage. Given, the diversity of resources in terms of heterogeneous fleet of UAVs, a resource allocation problem also gets associated with the coverage path design. This chapter aims to solve the problem of CPP over a set of disjoint regions using a fleet of UAVs which are heterogeneous UAVs in terms of their sensor footprint sizes, power rating and manoeuvring speeds.

Chapter 7: This chapter concludes by summarising the contributions detailed in the thesis. Finally, the future of research in the field of CPP and the role which this thesis will play is highlighted.

1.7 List of Publications

The contribution of each chapter has been published as following papers during the time duration of this doctoral thesis:

1. **Chapter 3**: **Z. Khanam**, S. Saha, D. Ognibene, K. McDonald-Maier and S. Ehsan, "An Offline-Online Strategy for Goal-Oriented Coverage Path Planning using A Priori Information," *2021 14th IEEE International Conference on Industry Applications (INDUSCON)*, 2021, pp. 874-881, doi: 10.1109/INDUSCON51756.2021.9529583.

2. **Chapter 4**:

    - **Z. Khanam**, S. Saha, S. Ehsan, R. Stolkin and K. Mcdonald-Maier, "Coverage Path Planning Techniques for Inspection of Disjoint Regions With Precedence Provision," in *IEEE Access*, vol. 9, pp. 5412-5427, 2021, doi: 10.1109/ACCESS.2020.3044987.
    - **Z. Khanam**, B. Aslam, S. Saha, X. Zhai, S. Ehsan, R. Stolkin and K. McDonald-Maier, "Gamma-Induced Image Degradation Analysis of Robot Vision Sensor for Autonomous Inspection of Nuclear Sites," in *IEEE Sensors Journal*, 2021, doi: 10.1109/JSEN.2021.3050168.

- B. Aslam, S. Saha, **Z. Khanam**, X. Zhai, S. Ehsan, R. Stolkin and K. McDonald-Maier, "Gamma-induced Degradation Analysis of Commercial off-the-shelf Camera Sensors," *2019 IEEE SENSORS*, 2019, pp. 1-4, doi: 10.1109/SENSORS43011.2019.8956620.
- **Z. Khanam**, S. Saha, B. Aslam, X. Zhai, S. Ehsan, C. Cazzaniga, C. Frost, R. Stolkin and K. McDonald-Maier, "Degradation Measurement of Kinect Sensor Under Fast Neutron Beamline," *2019 IEEE Radiation Effects Data Workshop*, 2019, pp. 1-5, doi: 10.1109/REDW.2019.8906531.

3. **Chapter 5**: **Z. Khanam**, K. McDonald-Maier and S. Ehsan, "Near-Optimal Coverage Path Planning of Distributed Regions for Aerial Robots with Energy Constraint," *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2021, pp. 1659-1664, doi: 10.1109/ICUAS51884.2021.9476696.

Chapter 2

Background Theory

2.1 Introduction

This chapter aims to familiarise the readers with prerequisite background in optimisation, specifically sub-field of combinatorial optimisation, that is effectively used as a solving and formulation tool for area coverage problems. The readers are first introduced to the fundamental concepts of optimisation in Section 2.2. Further, this chapter branches out to the allied research topics like combinatorial optimisation in Section 2.3 and Constraint Programming in Section 2.4. To elucidate a clear picture of the central theme of this thesis, CPP, TSP is used as an example due to its close relationship with classical coverage problem in Section 2.5.

After describing background theory of optimisation, a detailed overview of CPP literature is provided in Section 2.6. The chapter first focuses on classic approaches and taxonomy followed by CPP literature. Then, CPP problem in context of the aerial robots is introduced in Section 2.7 narrowing down to the research work exploring CPP of multiple distributed regions.

2.2 Optimisation

Optimisation is a family of mathematical development methods that have pervaded all walks of human life. Generally speaking, optimisation can be described as a decision making framework that allows identification of 'best available' solution for a problem constrained by a defined input domain [45]. It finds extensive application in multiple quantitative disciplines, such as computer science, operations research, engineering and economics. Despite its

ubiquitous nature, optimisation is reviewed through the lens of CPP problems from the robotics literature in this thesis.

Mathematically, an optimisation problem $O(.)$ for a given objective function $f : A \rightarrow \mathbb{R}$ can be formulated in the following way for minimisation problem:

$$O(f(x)) = \{x_0 \in A \mid f(x_0) \leq f(x)\ \forall x \in A\} \tag{2.1}$$

For the maximisation problem, the problem is formulated as:

$$O(f(x)) = \{x_0 \in A \mid f(x_0) \geq f(x)\ \forall x \in A\} \tag{2.2}$$

The *objective function* $f(.)$ can be considered as a goal that needs to be achieved. The optimisation can either be a process of maximisation or minimisation of objective function. The decision points and choices which have direct impact on the output of objective functions are known as *decision variables*. For instance, the start position of robot in an OSP problem. The set of values which the decision variable can take up is known as domain of the variable. These variables are either mutually exclusive or dependent on other variables. The variables which are dependent on other variables are known as *implicit* variables.

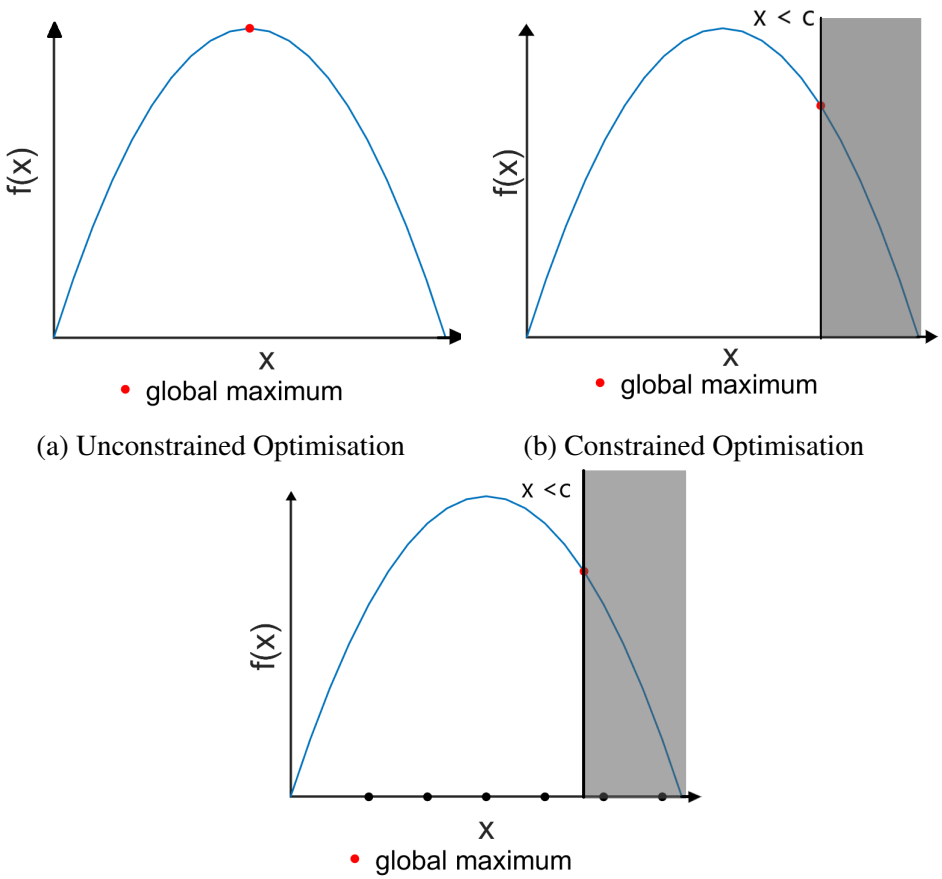
Optimisation problems can be categorised depending upon practical context of the problem at hand. Fig. 2.1 presents four categories of optimisation problems enlisted below.

1. **Unconstrained Optimisation**: It is a problem of optimising an objective function that depends on a single real-valued variable x. The domain of x is *continuous, infinite and uncountable* [46]. Mathematically, the unconstrained optimisation for maximising objective function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ and variable $x \in \mathbb{R}^n$ can be formulated as:

$$max_x \ \ f(x) \tag{2.3}$$

   Fig. 2.1a represents an unconstrained problem with a variable x for maximising objective function $f(.)$. In the given example, global maximum point (plotted as a red point in Fig. 2.1a of the quadratic function ($f(.) = -x^2 - 100$) is computed using a simple derivative.

2. **Constrained Optimisation**: Moving forward from unconstrained optimisation described above, constrained optimisation is introduced. In this set of problem, the input variable x is subjected to constraints [47]. In case the imposed constraints are hard in nature, generated solution is considered unfeasible, if constraints on the variables are not satisfied. On the other hand, if the variables are subjected to soft constraints then



(a) Unconstrained Optimisation

(b) Constrained Optimisation

(c) Combinatorial Optimisation

Fig. 2.1 Different types of linear optimisation problems.

the variable values are penalised in the objective function, when the constraints are not satisfied. Mathematically, a generic optimisation problem that minimises an objective function $f(.)$, can be formulated as:

$$\begin{aligned} \min \quad & f(x) \\ \text{subject to} \quad & g_i(x) = c_i \quad \text{for } i = 1, \ldots, n \end{aligned} \tag{2.4}$$

The optimisation problems can further be categorised based on type of constraints on the variables. For instance in Fig. 2.1b, the constraint $x < c$ is introduced. This additional constraint translates the problem into a convex optimisation problem. Similarly, if the objective function and all the constraints are linear in nature then the problem is termed as a linear optimisation problem.

3. **Combinatorial optimisation**: This is a set of optimisation problems where the domain of decision variables is a discrete number, i.e. countable [48]. Fig. 2.1c illustrates an example of combinatorial optimisation with a single variable. As the number of variables increase, the complexity of the problem increases exponentially, such that exhaustive search is not tractable. There are also scenarios where gradient descent does not converge to global optimum, but to local optima.

4. **Nonlinear optimisation**: Nonlinear optimisation is the family of optimisation problems, where one or more constraints or the objective function are nonlinear [49]. A nonlinear optimisation problem which aims to minimise an objective function $f(.)$, can be mathematically formulated as:

$$\begin{aligned} \min \quad & f(x) \\ \text{subject to} \quad & g_i(x) \leq 0 \quad \text{for each } i \in \{1,\ldots,n\} \end{aligned} \tag{2.5}$$

where n, m, and p are positive integers and $g_i(.)$ and $f(.)$ are real-valued functions with at least one of them being nonlinear. Fig. 2.2 depicts the following example of nonlinear optimisation problem:

$$\begin{aligned} \min \quad & 2 \times x_1 - x_2 \\ \text{st} \quad & x_1 \times x_2 \geq 25 \\ & x_1^2 + x_2^2 = 40 \\ & 1 \leq x_1, x_2 \leq 10 \end{aligned}$$

The subfield of combinatorial optimisation is described in detail in upcoming sections. The readers are given flavour of mathematical tools like constraint programming and exact and approximate algorithms using Travelling Salesman Problem.

2.3 Combinatorial Optimisation

As defined in previous Section 2.2, combinatorial optimisation is a type of optimisation where discrete choices are of major essence, i.e., the domain of decision variables is discrete. Thus, combinatorial optimisation can be stated as minimisation or maximisation of a objective function, that is subjected to constraints under discrete choices. The combinatorial

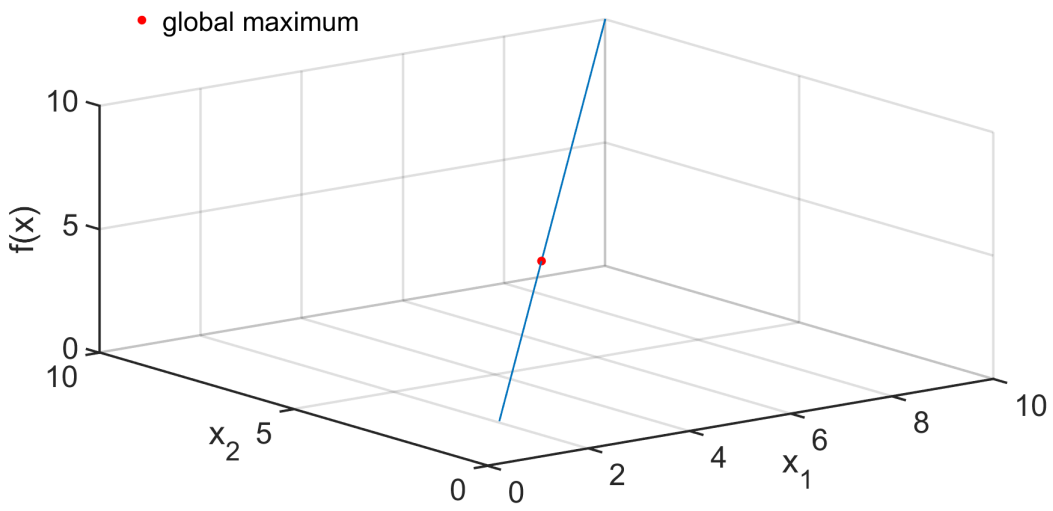


Fig. 2.2 Example of nonlinear optimisation problem.

optimisation finds application in diverse fields varying from logistics and supply chain to transportation, job allocation to network distribution.

The combinatorial optimisation problem suffers from the curse of dimensionality. As the number of decision variables increases, the discrete search space increases making the solution search intractable. Thus, the computational complexity class owing to large discrete space for real-life combinatorial optimisation problems are either NP-hard or NP-complete. In compliance with the central theme of this thesis, CPP, hard combinatorial optimisation problems which lie in the domain of NP-hard complexity are considered. An intuitive summary and definitions of fundamental terms and concepts are introduced using following definitions.

Definition 2.3.1 (Problem Statement) *Any given combinatorial optimisation problem can be defined by an objective function representing a goal subjected to a collection of constraints that is defined using a set of decision and implicit variables and constants.*

Definition 2.3.2 (Assignment) *A constant or a variable assignment is instantiated when their value is fixed.*

Definition 2.3.3 (Problem Instance) *The instance of any combinatorial optimisation problem consists of instantiated parameters.*

Definition 2.3.4 (Feasible Solution) *The feasible solution of a combinatorial optimisation problem is a set of values for decision variables that satisfies all the constraints in an optimisation problem.*

Definition 2.3.5 (Optimal Solution) *An optimal solution to a combinatorial optimisation problem is an element of the set of feasible solutions where the objective function attains its minimum or maximum value.*

Definition 2.3.6 (Global Optimum) *A global optimum solution to a combinatorial optimisation problem is an element of the set of optimal solutions where the objective value is atleast as good as any other feasible solution.*

Definition 2.3.7 (Local Optimum) *A local optimum solution to a combinatorial optimisation problem is an element of the set of optimal solutions where the objective value is best in the neighbourhood.*

It is often seen in case of highly complex combinatorial optimisation problems, that solution design targets a specific subproblem for effective and efficient solutions. In the context of area coverage problem, the objective focuses on determining an optimal solution to an applied problem or a better solution to an existing practise, has a prolific impact on the performance of the robot. Next section presents an introduction to Constraint Programming (CP), a generic combinatorial optimisation framework where the focus is laid on finding solutions to subproblems through specific algorithms and heuristics. This is part of the larger scheme of finding efficient solutions to complex and large combinatorial optimisation problems.

2.4 Constraint Programming

Constraint Programming (CP) is a mathematical paradigm used to formulate optimisation problems using a specific programming language. In CP, the constraints are implicitly stated [50]. Unlike imperative programming language, constraints are not specified as a sequence of steps to be executed. The combinatorial optimsation problems formulated in CP are referred to as models. Models comprise parameters, constraints and variables. There are array of solvers available to read the CP models to find a global optimum solution.

CP models can be classified either as: *1) Constraint Satisfaction Problems (CSPs), and 2) Constraint Optimisation Problems (COPs)*. The aim of CSPs is to determine a feasible solution, i.e., a solution which satisfies all the constraints. CSPs are devoid of objective function optimisation. COPs on the other hand can be seen as CSPs with an addition of constraints and variables to define the objective function.

Definition 2.4.1 (Domain) *The domain of any variable X is the set of values which the solver can assign to the variable.*

Definition 2.4.2 (CSPs) *Any CSPs can be denoted as a triplet* $< \chi, \mathbb{D}, \mathbb{C} >$ *where* $\chi = \{\chi_1, \ldots, \chi_n\}$ *is a set of variables,* $\mathbb{D} = \{\mathbb{D}_1, \ldots, \mathbb{D}_n\}$ *is the set of domains of the corresponding variables and* $\mathbb{C}$ *is a set of constraints.*

CPP applications tackled in this thesis have discretised the target area into a grid. Thus, domain of a decision variable is a set of integers and solver used is IBM ILP CPLEX[1].

Optimisation problems in CP are formulated as COPs. As mentioned before, COPs differs from CSPs by addition of an objective function. The COPs is elucidated using instance of a NP-hard problem, TSP, as it is closely related to CPP methodologies in next section.

2.5 Travelling Salesman Problem

Traveling Salesman Problem (TSP) is a classical mathematical problem which came to surface in the 19^{th} century [51]. It appeared as a mathematical puzzle which tried to find an answer to a question: *'Given a set of cities and inter-city distances, what will be the shortest route that a salesman must take and return to the depot such that each city is visited only once?'*.

2.5.1 ILP Formulation

Mathematically, the TSP can be modelled as an undirected weighted graph $G =< V, E >$ where $V = \{V_1, \ldots, V_n\}$ is a set of vertices representing cities and $E = \{(v_i, v_j) \mid (v_i, v_j) \in V^2$and $i \neq j\}$ are set of edges representing inter-city path. For any given instance, TSP can be represented as a graph in Fig. 2.3. In literature, TSP is often formulated as an Integer Linear Program (ILP) [52, 53]. The two most notable ILP formulations are discussed in upcoming sections.

Miller–Tucker–Zemlin

Miller–Tucker–Zemlin (MTZ) [54] is one of the earliest well known formulation. Even though, it is not computationally efficient but serves as an easy introduction to the readers for ILP formulation.

Each city is labeled as an integer using an enumerated list $1, \ldots, n$ such that n is the total number of cities and 1 denotes the city of origin and final return. This following decision

[1]IBM ILP CPLEX (also known as CPLEX) is an optimisation studio and software package.

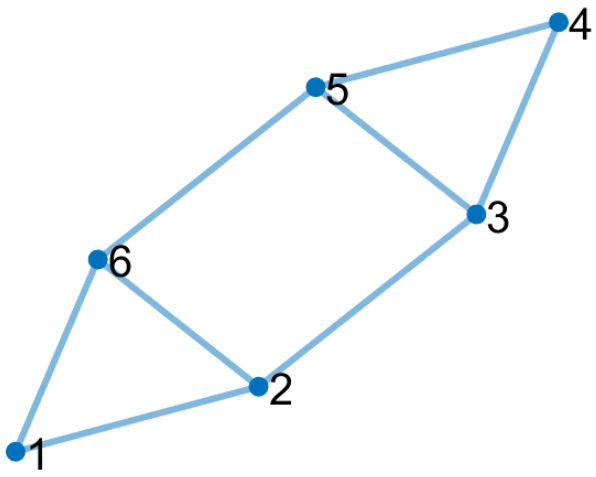


Fig. 2.3 A TSP Instance

variable can be described as:

$$x_{ij} = \begin{cases} 1 & \text{if there exists path between city } i \text{ and } j \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

For each city $j = 2,\ldots,n$, an auxiliary variable $u_j \in \mathbb{R}^+$ and a distance matrix c_{ik} represents distance between city i and k are introduced. The MTZ approach for ILP formulation of TSP is presented below. The objective function Z can be stated as:

$$Z = min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{2.7}$$

Subjected to :

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i = 1,\ldots,n; \tag{2.8}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j = 1,\ldots,n; \tag{2.9}$$

$$u_i - u_j + n x_{ij} \leq n-1 \qquad 2 \leq i \neq j \leq n; \tag{2.10}$$

$$1 \leq u_i \leq n-1 \qquad 2 \leq i \leq n; \tag{2.11}$$

$$x_{ij} = 0,1 \qquad i,j = 1,\ldots,n; \tag{2.12}$$

$$u_i \in \mathbb{Z} \qquad i = 2, \ldots, n. \tag{2.13}$$

Constraints enlisted in Equations 2.8-2.9 ensure that the salesman departs and arrive in each city only once. Equation 2.10 presents a constraint to eliminate subtour formation.

Dantzig–Fulkerson–Johnson

The major bottleneck in MTZ formulation is the high computational complexity due to the subtour elimination. This was improved by Dantzig–Fulkerson–Johnson (DFJ) [55] formulation that is presented below.

$$Z = min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{2.14}$$

Subjected to :

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i = 1, \ldots, n; \tag{2.15}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j = 1, \ldots, n; \tag{2.16}$$

$$\sum_{i \in Q} \sum_{j \in Q, i \neq j} x_{ij} \leq |Q| - 1 \qquad \forall Q \subseteq \{1, \ldots, n\}, |Q| \geq 2 \tag{2.17}$$

$$x_{ij} = 0, 1 \qquad i, j = 1, \ldots, n; \tag{2.18}$$

The DFJ formulation eliminates the subtour formation by adding constraint Equation 2.18, such that no proper subset Q is able to form a subtour. ILP formulations, no doubt, generate optimal solutions for TSP problems. However, the exponential increase in computational complexity as the number of decision variables increases, make them impractical for real world application. In the upcoming sections, practical approaches adopted in context of TSP that are also applicable to generic COPs are discussed.

2.5.2 Branch-and-Bound

Branch-and-Bound (B&B) [56] is a family of algorithms that apply bound on the domain of the variable depicting the objective value of the optimisation function, i.e., value Z in Equation 2.7. To achieve this, a search tree is constructed which is an exhaustive enumeration of all the solutions for the given problem. B&B prunes branches of the search tree using an implicit enumeration of all the solutions. The algorithm first estimates the attainable

objective value (called as bound) for current partial solution. The decision of branch pruning is taken by comparing incumbent bound and bound of the current solution. The major trade-off decision which algorithm designers take is achievement of a tight bound and usage of computational resources. B&B algorithm also inspires the popular A* algorithm [57] used in motion planning.

2.5.3 Dynamic Programming

As previously mentioned in Section 2.3, combinatorial optimisation problems exhibit optimal substructures which are considered during solution search in methods like CP. However, many COPs have properties which allow them to be divided into a set of recursively subproblems. Dynamic programming exploits this property of overlapping subproblems to generate computationally efficient solutions to COPs.

One of the prominent dynamic programming algorithm is Bellman–Held–Karp algorithm [58] to solve TSP. For set of cities $i = 1,\ldots,n$, let $S = 2,\ldots,n$ be a subset of cities. For each city $j \in S$, cost is defined as $cost(j,S)$ as the cost incurred when taking the shortest path from 1 to j. The recursive subproblem structure can be stated as:

$$cost(j,S) = min_k \{cost(k, S \setminus \{j\}) + c_{kj}\} \tag{2.19}$$

The time complexity of this dynamic programming algorithm is $O(n^2 \times 2^n)$.

2.5.4 Approximate Algorithms

In previous Section 2.5.2 and 2.5.3, exact algorithms were presented which determined the optimal solution working reasonably fast in smaller solution space. However, it has been noted that for large volume of scientific problems, the cost of optimal solution search is exponentially high [52]. This has ushered the researchers to devise standalone heuristics or approximation algorithms that yield into *'good'* solutions. Proposed approximation algorithms to TSP range from greedy algorithm to nature inspired algorithms [59]. The following subsections present a brief description of known TSP heuristics.

Greedy Algorithm

The greedy algorithm is a heuristic that searches for local optimum solutions (refer Definition 2.3.7). However, for few TSP instances, greedy algorithm can even converge to a global optimum. In greedy algorithm, starting from origin city, the salesman chooses nearest unvisited city as next move. This process is repeated unless and until all the cities have been

visited. This algorithm is also called as nearest neighbour algorithm [60]. On average for n cities randomly distributed, this algorithm determines a path 25% longer than global optima (shortest path) [61].

Perturbative Approaches

The greedy algorithm discussed in previous section starts from an empty solution and iteratively converges to a solution. On the other hand, there exist approaches which start with already existing feasible solutions initially and iteratively search the solution space to improve performance of an existing solution. These approaches are known as the perturbative approach [62]. One of the famous perturbative approach is local search. Local search algorithm takes the initial solution as a point in search space and successfully finds '*good*' solutions in neighbourhood space until a termination criterion is met. Ant colony algorithm [63], Simulated Annealing [64] and Genetic Algorithms [65] are well-known perturbative approaches used for finding optimal solution to a TSP instance. The algorithms in such cases require extensive experiments and ablation studies so that sub-optimal solutions are avoided. After providing the background of optimisation theory, the next section presents a detailed literature review of the existing research in the domain of Coverage Path Planning.

2.6 Coverage Path Planning Problem

CPP has carved a separate niche in the family of motion planning problems. This is alluded to by CPP's ability to be at the core of diverse robotic applications such as floor cleaning [66], minesweeping [67], precision agriculture [68], seabed survey [69] and painting [70] to name a few. However, CPP by an autonomous robot, to date, poses a considerable challenge and can be classified as NP-hard. An array of literature works [9, 71, 72, 11] has focused on surveying the coverage algorithms, mainly due to wide industrial applications. Apart from presenting a condensed overview, each survey has presented a taxonomy of CPP algorithms. The factors used for classification of coverage algorithms are as follows:

1. Intrinsic parameters of robot: sensing and localisation capabilities, and hardware-on board.

2. Extrinsic parameters: environmental conditions, and prior knowledge.

Fig. 2.4 illustrates the factors used for classification of the coverage algorithms. Each of these factors are discussed in-depth in following sections to provide an overview of classical CPP approaches.

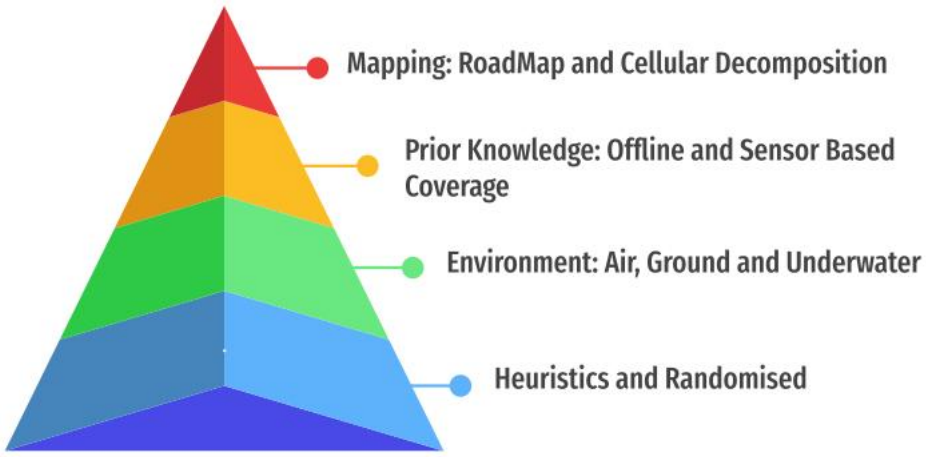


Fig. 2.4 Factors for classification of coverage algorithms.

2.6.1 Heuristic and Randomised Approaches

Early work in the CPP were classified into two categories: heuristics and randomised approaches. The approaches in the first category equip mobile robot with a set of simplistic behaviours (e.g., a wall follower) [73], but do not provide any guarantee of a complete coverage. However, early applications indicated utility in multi-robot coverage [74, 75]. Heuristic approaches ensure that robots do not collide with each other and also obstacles.

The approaches in the second category employ randomisation. These approaches rely on the fact that if a robot covers an area randomly for a long interval then a complete coverage is guaranteed. The major advantage of these approaches is commercial viable cost/benefit ratio. This is the reason that commercial vacuum cleaning robots like RC3000, Trilobite and Roomba adopt randomisation approach [76]. Due to randomisation approach, no complex or expensive sensors and computation resources are required for localisation and mapping. However, in case of coverage of 3D space, and specifically underwater and aerial robot coverage, randomised algorithms are not viable as the cost of operation in terms of energy and time grows exponentially as the size and complexity of solution space increases.

2.6.2 A Priori Information

If the robot has a priori information of the target site while planning the coverage path, the algorithm is known as a priori or an off-line CPP problem. Otherwise, if the robot does not have any prior algorithm, it is known as an on-line CPP problem or sensor-based CPP. Sensor-based CPP deploys fast and simple heuristics algorithms in planning [9]. However, off-line CPPs use combinatorial optimisation problems for path planning.

2.6.3 Mapping

Mapping the environment plays a major role in all path planning algorithms and CPP is no exception. This is the reason that taxonomy proposed in prominent surveys [71, 9] have emphasised on environment discretisation and mapping. The discretisation methods [77] can further be categorised as: *roadmaps* and *cellular decompositions*.

The fundamental concept of a roadmap is to generate points in the (obstacle-)free space and create connection between them on the basis of spatial occupancy and traversability. Two points are connected if they are neighbours and there exists a path between both of them which is traversable in nature. The description of neighbourhood of a point is problem dependent. The popular approaches of roadmap generation used in off-line CPP algorithms are voronoï diagrams [18], probabilistic roadmaps [77], and visibility graphs [78]. The sensor based coverage requires adoption of sophisticated roadmap generation techiniques, like the rapidly-exploring random trees [79].

However, due to the complexity and vast nature of target site, the discretisation of target site into regions is frequently used. This technique of region based decomposition of target site is called cellular decomposition. The approaches in the field of cellular decomposition further branch-out as uniform decomposition (set of uniform cells) and non-uniform decomposition (set of regions or subregions) [77]. Uniform cellular decomposition mainly deals with discretisation of the continuous target site into a grid of cells of same size. If the target site contains obstacles, the preferred approach is to build an occupancy grid [80]. These decomposition techniques are also known as approximate cellular decomposition due to loss of information during the grid mapping.

The same cell size constraint is relaxed in a non-uniform cellular decomposition. The popular exact non-uniform cellular techniques include trapezoidal decomposition [81] that handle polygonal target site, triangulations decomposition, such as Delaunay [82], Boustrophedon decomposition. The decomposition of a continuous space using an exact cellular decomposition is considered NP-hard [83]. A tractable cellular decomposition method

was presented by Huang [84] that minimises the number of turns that a robot takes while performing coverage operation.

Due to NP-hard nature of exact cellular decomposition, decompositions are performed using some optimality criterion [9]. Moreover, non-uniform exact decompositions are preferred over uniform decompositions in presence of obstacles [9]. Generally, the large cells generated by exact decompositions, are covered using heuristics. Two other intrinsic parameters of robot, shape and the orientation of the robot, play an important role in the discretisation of continuous environments. The incorrect positioning of robot can lead to collision with the obstacle [77]. However, the orientation of robot drives decision makers to plan path in configuration spaces. The configuration space is the collection of all the possible configuration of robot's possible configurations depending upon the total number of degrees of freedom of the robot [33, 77].

Finally, it is noted that the literature classifies algorithms into two classes, roadmaps and cellular decompositions. However, algorithms in these two classes are not completely independent. For instance, a grid decomposition is viewed as a roadmap [81] where the points are distributed uniformly over the target site. Similarly, a voronoi diagram is generated using the random points in a probabilistic map.

2.6.4 Sensing Capability

When CPP is executed by an autonomous robot equipped with a perfect sensor, a cell is considered fully covered after a single visit to the centre of the cell and no further visits are required. This problem of CPP with the perfect sensor is also called as an area covering problem [85] and a region filling problem [86]. If the target space is larger than the range of robot's sensor and can be decomposed into a set of cells then CPP translates to computation of ordering of the decomposed regions [9].

If the autonomous robot has an imperfect sensor then CPP algorithm can no longer provide a guarantee of complete coverage. The approach aims to add an additional constraint on minimal coverage required to be achieved for each cell. The problem of coverage with imperfect sensors has been compared with goal- oriented search in the target area [87]. The minimal required coverage constraint often requires robot to cover the target site multiple times leading to multiple overlapping passes. The applications of area coverage robots with"imperfect sensor" have recently emerged mainly in the field of extreme environments [88, 89]. However, these applications also assume that coverage with imperfect (uncertain) sensor is possible but only adds additional computational constraints.

Another important factor for classification of CPP algorithms is the operating environment (air, ground and underwater) of the robot (see Fig. 2.4). In this section, the focus was mainly

on the ground-based robots. Since, the operating environment of the autonomous robot in Chapter 4-6 is air, the next section describes the literature in the domain of aerial robot based CPP.

2.7 Coverage Path Planning for Aerial Robots

Aerial robots find application in a wide range of domains, like surveillance [90], disaster management [91], precision agriculture [92], security [93], structural health monitoring [94], photogrammetry [95], wildfire tracking [96], and cloud inspection [97]. The aerial robots or Unmanned Aerial Vehicles (UAVs) comprises aerial platforms devoid of on-board pilots. These platforms are either operated by humans remotely or automated by pre-programmed flights. Intelligent systems integrated with on-board sensors play a major role in execution of autonomous flights.

The technological progress in the development of aerial platforms pertaining to autonomous flight has seen rapid deployment of UAVs for area coverage [98]. The upcoming section discusses type of UAVs and further branches to the area discretisation techniques and recent CPP algorithms.

2.7.1 Types of Aerial Robots

The UAVs can be classified into two types depending upon the configurations, fixed-wing and rotary-wing. Given the control and guidance system, both the types of UAVs have their own set of advantages and challenges [99]. The fixed-wing UAV comprises rigid wings with an airfoil that allows UAV to fly using the lift generated by the forward airspeed. The control surfaces in the wing allows the navigation control. The aerodynamics of fixed-wing UAV provides the advantages of longer endurance flights and high-speed motion. The other advantage is ability to carry heavier payloads as compared to rotary-wing UAVs. However, the challenges of fixed-wing UAVs are additional requirement of runways for flight take-off and landing, and inability to hover [100]. On the other hand, the rotary-wing provides advantage of maneuverability using rotary blades. The other advantages include ability to perform vertical take-off and landing, low-altitude flights, and hovering [99]. The rotary-wing UAVs can also be classified as a single-rotor UAV and a multi-rotor UAV.

The single-rotor UAV is actually equipped with two rotors. The main rotor is used for navigation and the rotor in the tail is used for control. This type of UAVs have an ability to vertically take-off and land. As compared to their counterpart multi-rotor UAV, they have a longer endurance flight and can carry high payloads. The major disadvantages of

such platforms are mechanical complexity and elevated cost [100]. The multi-rotor can be classified based on the number of rotor blades. The most common type of multi-rotor UAVs are the quadcopter and the hexacopter. Recently, tricopters and octocopters have also been developed. Multi-rotors UAVs have the advantages of agile platforms and ability to maneuver and hover. Nonetheless, multi-rotor UAVs have limited payload and shorter flight endurance. These UAVs have simplistic mechanical and electrical design [100].

Another type of UAVs of interest is the hybrid UAV. The hybrid UAV is a specific aerial platform with the advantages of both types of UAVs, fixed-wing and rotary-wing. These UAVs have the capability of vertical take-off and landing, high-flight speed and longer flight time [101]. Finally, other factors for UAV classification in the literature are altitude and endurance. In such cases, the UAVs can be classified based on low, medium, and high altitude, and short and long endurance [102].

2.7.2 Area Discretisation

Given the target area that comprises the free space and the boundaries, the CPP problem translates to computation of path that can cover the entire area of interest considering the motion restriction and sensor characterisation of the robot. In the context of aerial target environment, the obstacles in the space represent the no-flight zones.

As mentioned in Section 2.6.3, the target site is decomposed into a set of non-intersecting regions also known as cells. There are an array of decomposition techniques that are applied to determine the set of cells. The resolution and size of the cells and decomposition techniques are problem dependent. The path for coverage of the cell is highly dependent on the cell size. For cells of large size, several passes are necessary for complete coverage of the area. While in the smaller cells, a single pass is enough for complete coverage. Generally, the size of cell is proportional to the footprint of the sensor. In the next section, the target area and the cellular decomposition techniques are explored.

Target Area

The target area for area coverage in literature is represented as a set of n vertices $V = \{v_1, \ldots, v_n\}$. Each vertex $v_i \in V$,is described as a coordinate pair $(v_x(i), v_y(i))$ and the internal angle γ_i. If v_i is the current vertex then the next vertex $v_{next(i)}$ of the polygon target area can be computed as $next(i) = i(modn) + 1$. The edge connecting between two vertices v_i and $v_{next(i)}$ is denoted by e_i. The length of edge e_i is $l_i = ||v_i - v_{next(i)}||$. Furthermore, the target areas can contain no fly zone within the region and are depicted as a sequence

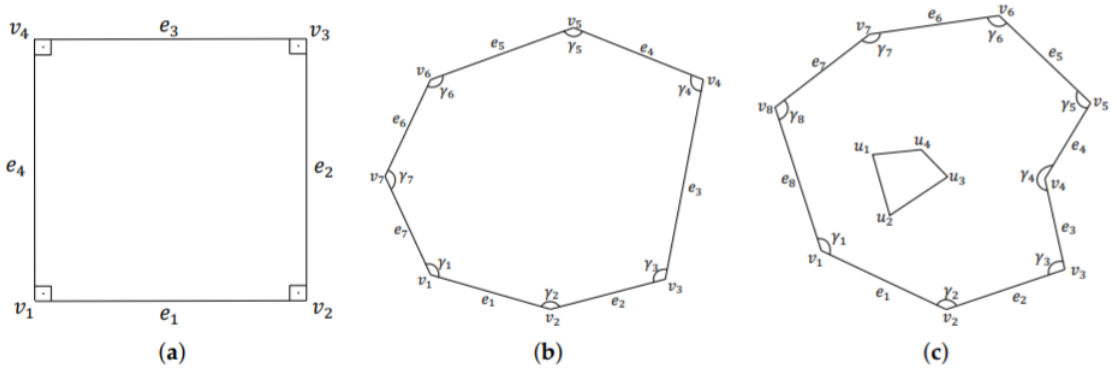


Fig. 2.5 Different areas of interest explored during CPP missions: (a) Rectangular; (b) Convex Polygon; (c) Concave Polygon with No-Fly Zones [72].

of obstacle-points $\{u_1, \ldots, u_n\}$. Fig. 2.5 shows examples of three target areas of different polygons.

The shape of the target area also plays an important role in designing coverage path. A majority of the CPP approaches [72] consider diverse polygon shapes including concave, convex and irregular shapes. However, a few of the approaches have restricted the coverage path design to polygons that are rectangular in shape [42]. Recent approaches [103, 104] have also considered no-fly zones while designing coverage path. These no-fly zones usually indicate the region where coverage is not allowed or unnecessary.

Cellular Decomposition

The major requirement for the success of the CPP algorithm is to provide feasible solution in acceptable time limit. The successful coverage is achieved by decomposing the target site using cellular decomposition techniques. This is usually achieved by applying cellular decomposition [9]. In context of UAVs, literature contains ample of cellular decomposition techniques [72]. But, the most common used technique is approximate cellular decomposition.

Exact cellular decomposition process decomposes the target site into subareas, also called as cells. Usually, the coverage of the cells is performed by simple set of motions like back-and-forth. This allows the CPP problem to be reduced to a path planning from one cell to another [9]. The robot traverses between spatially adjacent cells. The global inter cell path is designed using an adjacency graph where cells are the nodes and edges connect neighbouring cells. An adjacency graph is depicted in Fig. 2.6. After computing the global inter-cell coverage path, the final coverage path comprises intra-cell coverage and inter-cell connection path [71].

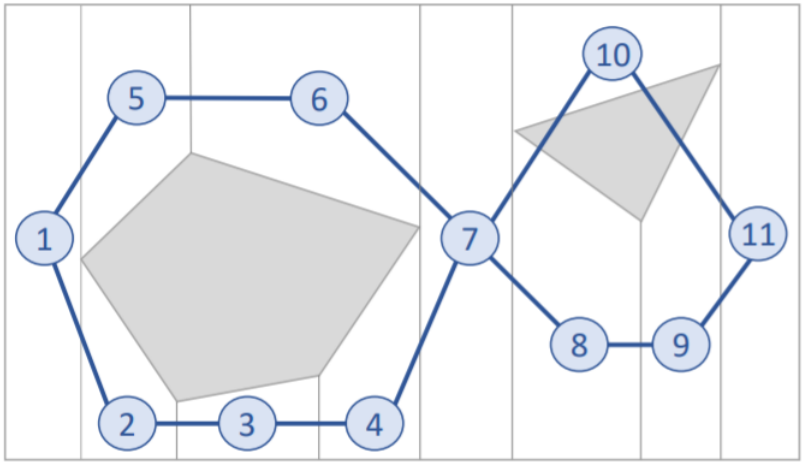


Fig. 2.6 Adjacency graph representing the workspace splitted into cells [72].

The two relevant exact cellular decomposition techniques commonly used for aerial surveys are: 1) trapezoidal decomposition and 2) Boustrophedon decomposition (Boustrophedon refers to "the way of the ox"), as shown in Fig. 2.7. The first decomposition technique divides the target area into a set of convex trapezoidal cells, and performs intra-cell coverage using back-and-forth motions. The second decomposition technique divides the target area into a set of non-convex cells using vertices of the obstacles. The critical points are determined by inserting a vertical sweeping line along the selected obstacles vertices. The Boustrophedon decomposition in comparison to trapezoidal decomposition generates fewer cells in order to minimise the coverage path length.

The approximate cellular decomposition technique decomposes the target area into a set of regular cells [9]. The regular cells are usually of square shape, but they can also be of triangular or hexagonal shape. Several grid-based methods are applied after approximate cellular decomposition to compute coverage paths [71]. The size of cells is proportional to the footprint of the UAVs, as the main aim of CPP is to map the target area. The resolution of cell is obtained using sensor requirements, resolution and overlapping rates, and the sensor characteristics, as illustrated in Fig. 2.8(a).

The computed coverage path is a set of m waypoints $W = \{w_1, ..., w_m\}$. Each waypoint $w_i \in W$ denotes a navigation command to the UAV. The waypoints also contain all the necessary localisation information. Since, the cells are proportional to the footprint, the center of each cell is referred as a waypoint, as shown in Fig. 2.8(b).

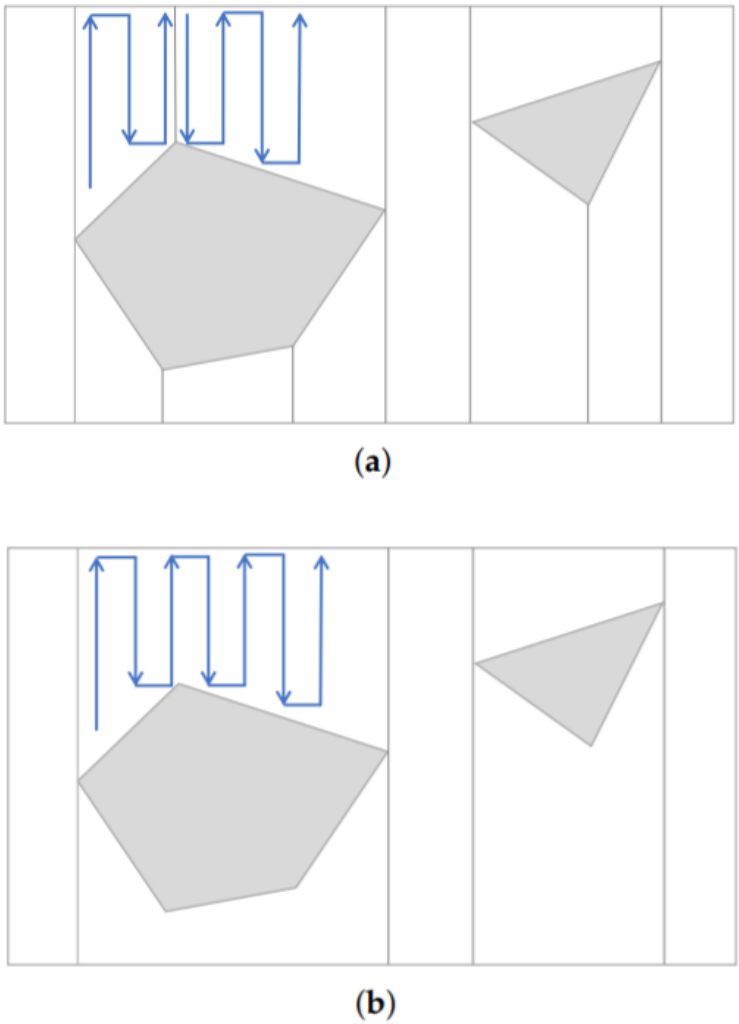


Fig. 2.7 Two types of exact cellular decomposition: (a) Trapezoidal decomposition; (b) Boustrophedon decomposition [72].

No Decomposition

The area coverage of regular-shaped polygons and non-complex areas using a single UAV generally does not require any type of decomposition. The area coverage is performed by the UAVs through simple geometric patterns. The most common coverage patterns are the back-and-forth and the spiral. The back-and-forth motion is used in the Mission Planner, the popular flight-control software [105]. In the back-and-forth coverage pattern, the UAV moves along the straight lines crossed in both directions with closed-angle maneuvers at the end of each round. In the spiral coverage pattern, the UAV covers the area by passing

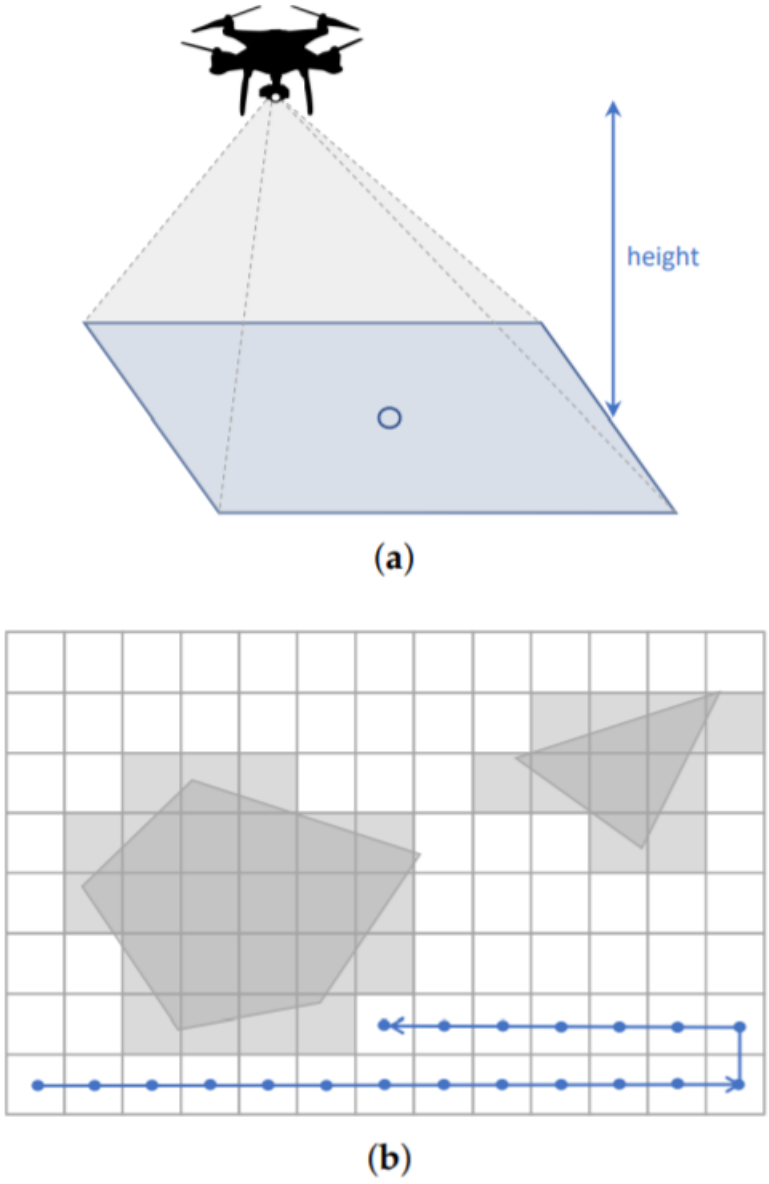


Fig. 2.8 Approximate cellular decomposition: (a) Projected area; (b) Regular grid with waypoints [72].

through the external vertices of the area while reducing the radius towards the central point after every complete circle.

Andersen [106] has proposed different types of coverage patterns for a rectangular area and compared them in detail. The author has classified back-and-forth pattern as parallel and creeping line, as illustrated in Fig. 2.9(a)-(b) and the comparative study finds its utility in application where target area is large. Another interesting pattern is the square flight pattern which is similar to spiral pattern. Unlike spiral pattern, it contains straight lines and 90°

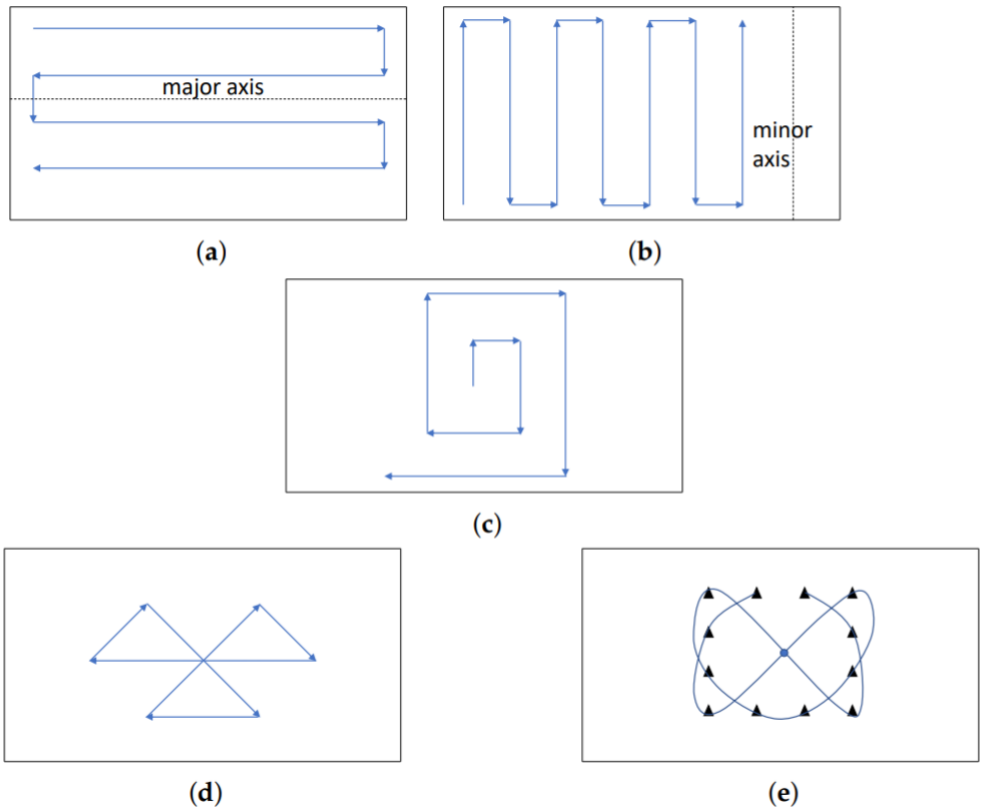


Fig. 2.9 Simple flight patterns in rectangular areas with no decomposition: (a) parallel; (b) creeping line; (c) square; (d) sector; (e) barrier [72].

turning maneuvers towards the right side. Also, the pattern starts at the centre and moves towards the circumference. It is usually used when the application requires a uniform area coverage, as shown in Fig. 2.9(c).

The sector flight pattern, illustrated in Fig. 2.9(d), contains a straight line with 120° turning maneuvers towards the right. The UAV returns to its starting point after completing first batch of three sectors. Then, the same batch of three sectors is repeated but with the displacement of 30°. The last pattern of coverage used for area coverage is barrier pattern. In the barrier pattern, the UAV executes a motion by visiting 12 spatially distributed waypoints in the search area, as illustrated in Fig. 2.9(e). The trajectory between two points is circular in nature. After providing the overview of the target area and area decomposition, the next section dives deep into research work exploring the aerial coverage of distributed regions.

2.7.3 Multiple Distributed Regions

A vast majority of prior works have focused on computation of coverage path for a single region [107, 108]. However, there can exist scenarios such as post-disaster relief, environmental monitoring, military surveillance, search and rescue missions, where the UAV is required to cover multiple regions. There are several works which explore the problem of computation of coverage path for multiple disjoint regions. In addition to the need of coverage path computation for multiple regions, instead of a single one, the problem with disjoint regions introduces the additional challenge of generating least cost tours covering a set of regions, also known as the Travelling Salesman Problem (TSP). The combination of CPP and TSP, both of which are classified as NP-hard, introduces significant challenges.

This problem was first introduced by Xie et al. [42] who proposed a dynamic programming solution to compute coverage path for multiple disjoint rectangular regions. The authors further extended this algorithm for multiple convex region by proposing a genetic algorithm solution [43]. Vasquez-Gomez et al. [109] attempted to solve this problem by proposing a two step path planning algorithm. All these works have assumed that the UAV had sufficient energy to cover all the regions. But only recently, potential energy limitation and constraints have been considered when computing coverage path over multiple disjoint regions [44]. This work allowed the UAV to return to the depot to change its battery. However, this might not be a realistic scenario in many cases and limited energy may lead to partial area coverage. Nevertheless, appropriate execution of partial coverage can provide useful information of the site. Recently, few works have explored optimisation of coverage path for partial coverage of a single region [110–112].

2.8 Summary

In this chapter, readers are introduced to the optimisation problem and subtopics pertaining to area coverage problem. The subfields of combinatorial optimisation are explored in depth. The mathematical modelling tools like constraint programming that have been used later on are discussed in detail to illustrate the coverage path problem formulation. Since, these tools generate computationally expensive solutions, the exact algorithms and approximate algorithms are explored to find optimal and suboptimal solutions. Furthermore, this chapter acquainted the readers with a detailed overview of CPP literature. The chapter first explained the classic approaches and taxonomy followed by synopsis of CPP literature. Finally , CPP in the context of aerial robots is described by narrowing down to the research work exploring the problem of CPP with multiple distributed regions.

The gaps identified in prior works can be clustered into four major challenges. The first two major challenges that autonomous robots face stem from imperfect sensing capabilities. These challenges are further intensified in an extreme environment. Due to the imperfect sensing capabilities of the autonomous robot is imperfect, robot needs to explore an area more than once for complete coverage. There arises an interesting challenge of designing an algorithm where trade off between coverage area and time is critical for success for the coverage task. This research challenge is explored in Chapter 3 and 4. Finally, the last two challenges stem from the adverse impact of limited energy and varying footprint,speed, and power on long term autonomy. Chapter 5 and 6 explore the possible solutions to overcome these challenges.

Chapter 3

Goal-Oriented Coverage Path Planning

3.1 Introduction

One of the key challenges for CPP, as identified in Chapter 1, is to equip mobile robots with the ability to autonomously plan a coverage path to reach the static target effectively and efficiently. The current approaches to achieve such tasks, however, are often time-consuming. Therefore, in this chapter, an offline-online strategy is proposed to meet the speeding-up challenge by efficiently modelling the environment using a priori information. In the 'offline' stage of the strategy, environment layout is segmented into a set of regions. The corners and dead-ends are identified based on spatial mobility of the regions. The global path is then computed by deriving a graph-structured, road map, using segmented regions. In the 'online' stage, the global path is traversed by selecting frontiers which concurrently minimise the covered area and time. In case the path is obstructed, a re-planning strategy is deployed. The proposed strategy is evaluated by various experiments against two baseline search approaches in three simulated environments. The results manifest a significant reduction in time to reach the goal and coverage area which caters to the objective set by decision makers given time criticality of the coverage task. This chapter presents all the details of the developed framework and an extensive analysis of this framework.

The remainder of the chapter is organised as follows. The overview of the problem is presented in Section 3.2. This is followed by mathematical formulation of problem in Section 3.3 to present a clearer picture of GO-CPP as a problem to the readers. In Section 3.4, the proposed offline-online strategy is discussed in detail. The experimental evaluation of the proposed method is performed in Section 3.5. Finally, the conclusions are summarized in Section 3.6.

3.2 Problem Overview

Chapter 1 discussed two problems, CPP and OSP. Due to the different objectives, both these problems have different formalism and application. Recently, applications have emerged where a robot is required to reach a goal in tandem with sensor based area coverage. This has given birth to challenging CPP problem known as Goal-Oriented Coverage Path Planning (GO-CPP). A literature review in Chapter 2 revealed that intensive research has been carried out in the field of CPP. However, a combination of both the problems, GO-CPP, remains unexplored. This sets up the stage for GO-CPP framework to be presented in this chapter.

CPP is a canonical problem in the field of mobile robotics, finding applications in diverse fields like search and rescue missions [113–115], surveillance and monitoring [116–118] and habitat conservatory programs [119, 120]. However, recent times have witnessed an advent of challenging coverage tasks like *Nuclear Decommissioning* and *Oil and Gas power plant Inspection* where the target (such as reactor core, pressure vessels, turbine) is stationary and their location can be estimated using a priori knowledge. However, the main challenge stems from strictly minimising the coverage time associated with the task [121, 122, 88].

Traditionally, a frontier exploration strategy [123] is used to exhaustively cover the environment. The goal is perceived as a part of the environment and the robot is expected to locate target by referring to occupancy grid that is incrementally built during coverage of the environment. To speed-up this exhaustive coverage, the static targets are spatially modelled [124] as a subspace of the site. In order to do so, a priori information of indoor environment in the form of an architectural layout of the site (metric map or floor plan) is used. However, any architectural layout is regarded as a non-deterministic environment for path planning, due to the inconsistencies in the form of scale (between online grid map and floor plan) and presence of dynamic or unmapped obstacles like debris or radiation. This has been the reason that heuristic based GO-CPP methods (like branch and bound [125] or A* [126]) have failed to perform efficiently in practical scenarios [127].

To overcome the challenges posed by GO-CPP with strict time constraint, a novel offline-online strategy is proposed (see Fig. 3.1) which performs a time-efficient geometrically-aware coverage task. The coverage strategy exploits underlying structure of the search site, spatially modelling the target using walls and openings which are typically not subject to any major change over the years and thus still can be classified as a relevant source of a priori knowledge [128].

The *'offline'* stage decomposes the layout into a set of regions using *'physical'* and *'virtual'* representative lines aligned along the walls and perpendicular to the openings, respectively. The segmented regions are clustered into three sets: *'junction'*, *'dead-end'* and *'corner'* according to their spatial mobility and connectivity with its neighbouring region.

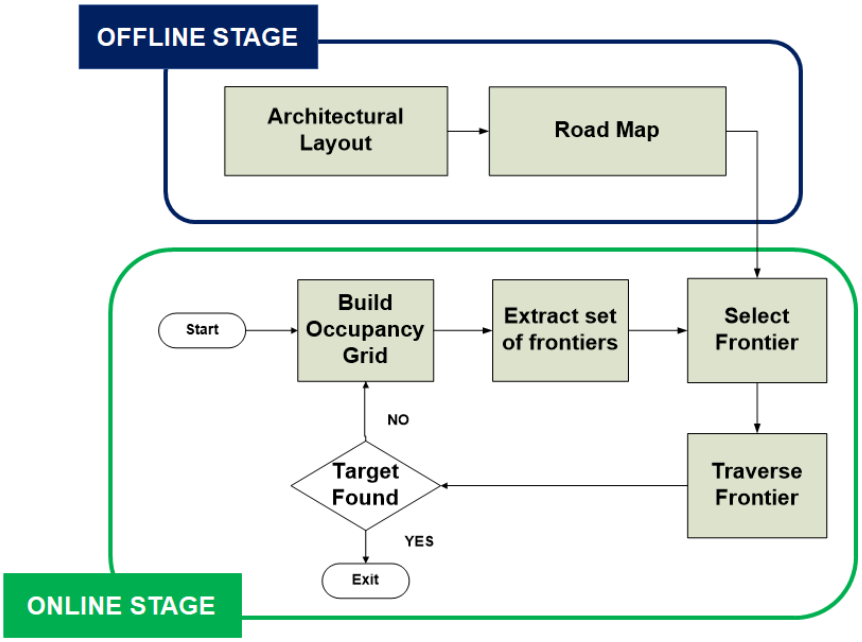


Fig. 3.1 Work flow of the proposed offline-online strategy.

After decomposing the layout as a set of regions, the goal is spatially modeled as a destination region. To keep-up with the scale related inconsistencies, the path is planned as a set of regions which the robot needs to traverse to locate the target. A graph structured road map is built, with each node as a region which contains all the possible paths from source (start location) to destination (target). Any node in the road map is either a *'corner'* or a *'dead-end'* only if it is the source or target region. Thus, planned path allows the robot to avoid getting stuck in a corner or dead-end while searching for the goal.

The *'online' stage* explores the site using frontier exploration [123]. The robot extracts a set of frontiers from the occupancy grid which it incrementally builds during exploration. From the detected set of frontiers, a frontier is selected for traversal based on the criteria which allows the robot to follow planned path while minimizing coverage area and maximizing information gain. However, if an obstacle is encountered at runtime, a re-planning strategy is deployed using the road map which improves robustness by making the robot aware of close alternatives. The re-planning strategy is tested by integrating random obstacles at runtime.

3.3 Problem Formulation

In the given problem, a robot stationed at a random location has to explore an indoor environment to reach a static target. Starting from its current position c_r, the robot explores

the environment $\Gamma \subset R^2$ while constructing the map as an occupancy grid until it finds the static target t_r. Taking the input from an architectural layout, static target location is modeled as a region γ_{target} of environment Γ, due to non-deterministic environment and uncertainties of layout. The aim of the robot is to reach target region by minimizing the exploration area. The problem is illustrated in Fig. 3.2. If P is the path robot executes to reach from c_r to γ_{target} then $\gamma_{known}(P)$ is the region it explores while traversing the path. Let $\chi(w)$ be a function that outputs the area under input region w and $F(P)$ be proportion of entire site explored during traversal of P.

$$F(P) = \frac{\chi(\gamma_{known}(P))}{\chi(\Gamma)} \tag{3.1}$$

The problem translates to finding optimal path P^* such that

$$P^* = arg\ min_P\ F(P) \tag{3.2}$$

subjected to,

$$\gamma_{target} \subseteq \gamma_{known}(P)$$

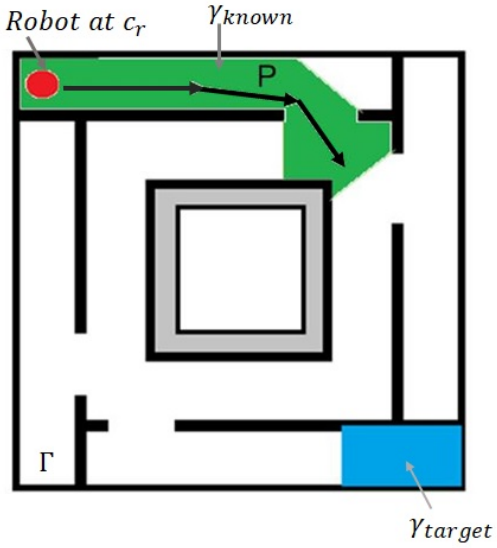


Fig. 3.2 The robot at c_r searches an extreme environment Γ to find a target region γ_{target} exploring γ_{known} region such that $\gamma_{target} \subseteq \gamma_{known}$.

3.4 The Proposed Strategy

This section presents an offline-online strategy for planning a path for GO-CPP. The offline stage exploits underlying structure of the site by constructing a road map of paths to static

target. In the online stage, each region of the planned path is covered with the aim of minimizing coverage area. Both the stages are discussed in detail below.

3.4.1 The Offline Stage

An architectural layout is a schematic projection of the site. Formally, layout X is a set of n disjoint rectilinear polygons $X_1, X_2, \ldots, X_n$ such that

$$X = \bigcup_{i=1}^{n} X_i \tag{3.3}$$

Each rectilinear polygon denotes an important planar landmark which is often referred to as *'room'*. All the rooms are enclosed by wall segments, which are the edges of rectilinear polygon in X denoted by bold lines. Rooms are interconnected using an *'opening'*, which can be a door or any other outlet in a wall segment of the room. Let $O = \{o_1, o_2, \ldots, o_w\}$ be a set of openings in X.

Map Segmentation

Often, the layout like floor plan and evacuation maps are images which contain symbols and markings to represent emergency exit, room names, fire extinguishers, furniture, etc. These input maps are pre-processed using template matching technique [129] to remove all the symbols and clean the layout. The 'clean' representation of layout is referred as map X'. Extraction of the wall segment from a layout is a well researched problem [130], [131]. However, the technique proposed by [132] is selected due to its robust nature to work well on a variety of layouts ranging from grid maps to floor plans.

The walls are represented as line segments in the layouts. The method detects line segments S using Canny Edge detector [133] and processes it further by Hough line transform [134] as shown in Fig. 3.3. The lines segments are then hierarchically clustered according to the angular coefficients into set C. Each cluster $C_i \in C$ is clustered according to the spatial proximity into $W_{i,k}$. The wall segment is considered as a 'physical' representation line which segments a layout into *'room'*.

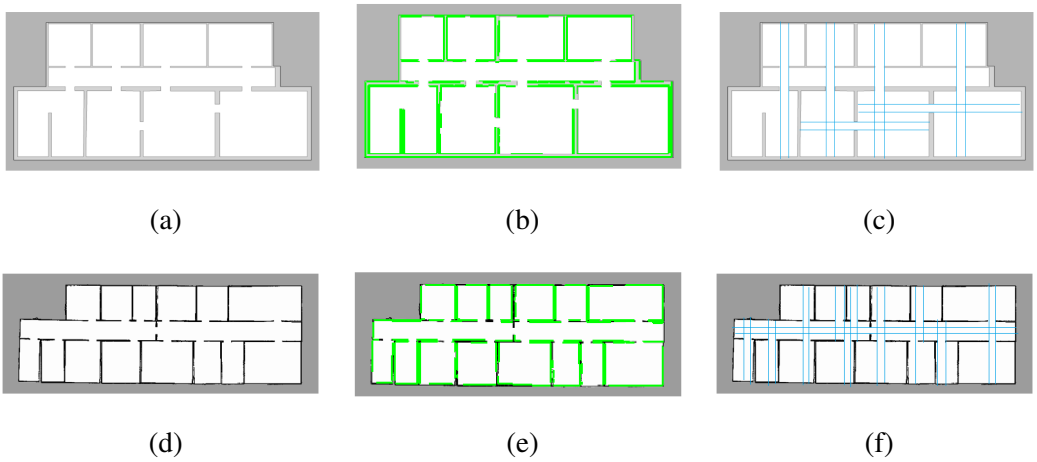
However, for a time-constrained GO-CPP, a robot does not need to cover all the rooms completely. A part of room is covered if it contains target or brings the robot closer to target location. Therefore, entity *'room'* must be segmented into a set of regions using openings. The spatial localization of *openings* define permitted movement in X'. The openings O are identified if there exists walls in the cluster $W_{i,k}$ which are collinear. This intuition drives Algorithm 1 to segment floor plan X into a set of closed convex regions, $Q = \{q_1, q_2, \ldots, q_m\}$

Algorithm 1: MAP SEGMENTATION

Input:
1. X' : Map
2. $O = \{o_1, o_2, \ldots, o_w\}$: Set of openings

Output: $Q = \{q_1, q_2, \ldots, q_m\}$: Set of Regions

begin
  Initialization:
    1) $K, Q = NULL$
    2) $X'' = X', \hat{O} = O$ ▷ X'' denotes modified floor plan and $\hat{O}$ is a set of opening in X'
  for *each* $\hat{o}_i \in \hat{O}$ **do**
    Draw prependiculars to opening $(\hat{o}_i)$ between two wall segments in X' ▷
      Prependiculars closes the existing gap of at least one opening
    Removes closed openings from $\hat{O}$
  Determine, $K = \{k_i \mid k_i$: closed polygon in X' $\}$
  $Q =$ *Region_Label* (K, Q) ▷ Q satisfies the condition stated in equation 3.4
Function `Region_ Label`**:**
  for *each rectilinear polygon* $k_i \in K$ **do**
    Assign label q_i to k_i
    $Q = Q \cup \{q_i\}$
  return Q



(a) (b) (c)

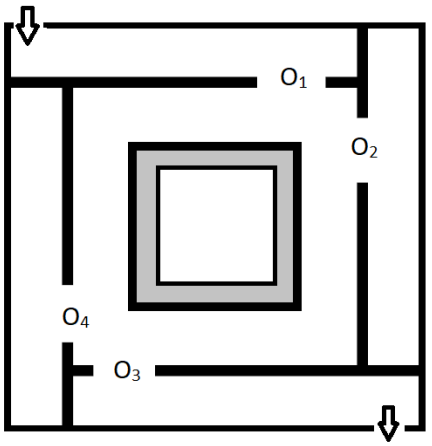
(d) (e) (f)

Fig. 3.3 Illustration of (a,d) layout map [135] (b,e) wall segments detection (c,f) segmentation of the map.

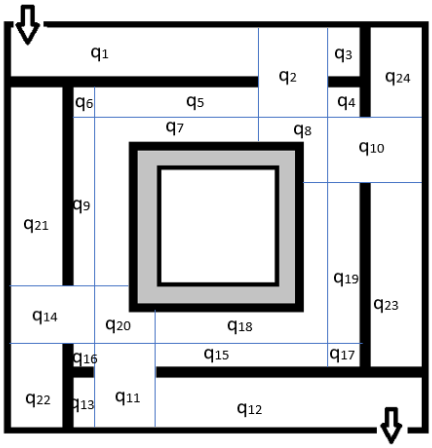
where

$$Q = \{q_i \mid \exists \text{ edge of } q_i \perp o_t; \text{ where } 1 < t < w\} \tag{3.4}$$

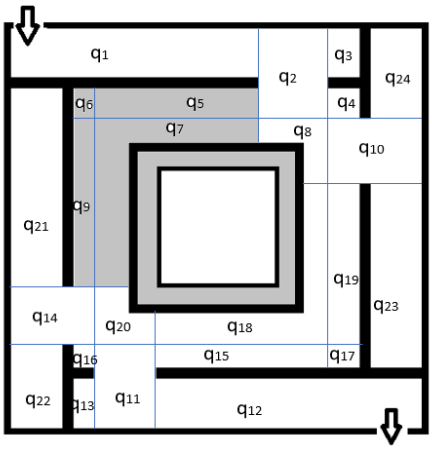
The perpendicular drawn at opening junctions are *'virtual'* representative lines. Fig. 3.4b illustrates map segmentation for the floor plan depicted in Fig. 3.4a.



(a)



(b)



(c)

Fig. 3.4 Site A represented as (a) a floor plan (b) a segmented map (c) with obstruction.

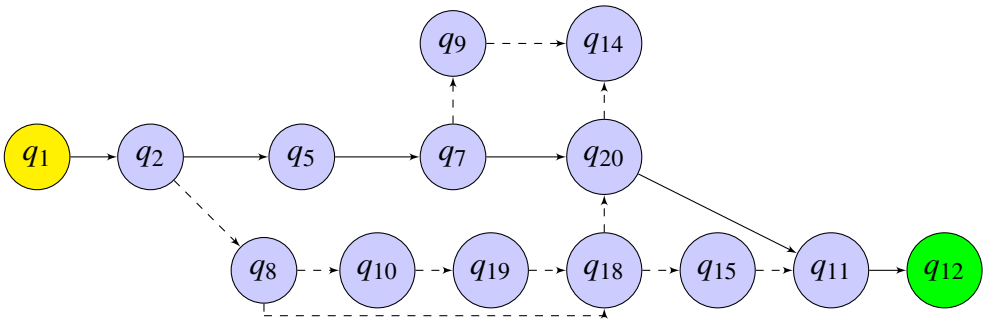


Fig. 3.5 Road map of site represented in Fig. 3.4

Road Map

Due to non-deterministic nature of the environment and uncertainty of scales, spatial occurrence of the goal is modeled as a region, q_d. For any given environment, while searching for the destination region, there can exist a set of paths $P = \{p_1, p_2, \ldots, p_z\}$ which the robot can follow. The offline stage of the framework targets to define all paths in P as traversal of regions in Q. A path $p_l \in P$ in $TG(V,E)$ with q_s as source region and q_d as destination region can be formulated as:

$$p_l = ((q_s, q_{k_1}), (q_{k_1}, q_{k_2}), \ldots, (q_{k_m}, q_d)) \tag{3.5}$$

All paths in P are represented as a directed graph, $TG(V,E)$, consisting of vertices $V = \{v_1, v_2, \ldots, v_b\}$, connected using edges $E \subseteq V \times V$. The vertices V represent regions in Q such that $V \subset Q$ and edges E indicate connection between them. Algorithm 2 constructs a graph given a set of regions Q. Two vertices v_i and v_j in $TG(V,E)$ are connected if, and only if, region corresponding to both the vertices are neighbours in modified map X'' and there exists a common edge between the regions which is a *'virtual'* representative line. $N(q_i)$ denotes neighbours of regions q_i which is determined using Manhattan distance. If two regions in X' share a common edge which is a *'virtual'* representative line, it implies that robot can move from one region to another and vice versa. Considering time constraints, it is judicious for a robot not to venture into corners and dead-end until they are modelled as target regions. As the robot uses frontier exploration [123] to cover the environment in online stage, to minimize coverage area, frontiers need to be selected in the regions which robot needs to traverse based on mobility. Regions are identified as *'corners'*, *'dead-end'* and *'junctions'* based on spatial mobility.

The spatial mobility of any region q_i is defined as a *degree* which is number of edges in the region q_i that are *'virtual'* representative lines. Any region with $degree = 1$ represents dead-end loops and is not added to $TG(V,E)$ unless the region is source or destination. For

Algorithm 2: ROAD MAP

Input:
1. $Q = \{q_1, q_2, \ldots, q_m\}$: Set of regions
2. q_s, q_d: Source and destination region

Output: $TG(V,E)$: Road Map

```
1  begin
2      Initialize: TG(V,E) = φ
3      TG(V,E) = Add_Node(q_s, q_d, TG(V,E)) ▷ Robot starts traversing from source region q_s
         to destination region q_d
4  Function Add_Node ( q_i, q_d, TG(V,E) )
5      Add q_i as a vertex to TG(V,E)
6      if q_i == q_d then
7          return TG(V,E)                          ▷ Robot reaches destination region q_d
8      for each region q_j ∈ N(q_i) do
9          if ∃ (common edge between q_j and q_i ≠ virtual) then
10             if degree(q_j) > 2 then
11                 Add_Node( q_j, q_d, TG(V,E) )          ▷ q_j is a junction region
12             else
13                 if degree(q_j) == 2 AND q_j is not a corner then
14                     Add_Node( q_j, q_d, TG(V,E) )
15                                                  ▷ Robot avoids traversing corners
16                 else
17                     if degree(q_j) == 1 AND q_j == q_d then
18                         Add q_d as a vertex to TG(V,E)
19                         return TG(V,E)
20                             ▷ Robot reaches destination region q_d and searches for the goal
21     return TG(V,E)
```

instance, regions q_{24}, q_{21}, q_{22} in map illustrated in Fig. 3.4b are considered dead-end loops. Similarly, corners are identified as regions with $degree = 2$ such that both the edges which are *'virtual'* are perpendicular to each other and have a common end point. Regions like q_6, q_4, q_{17} and q_{16} in Fig. 3.4b are examples of corners. All the remaining regions are labelled as *'junctions'*. The map of site A is used as a running example to elucidate Algorithm 2 as illustrated in Fig. 3.5.

Path Selection

The generated road map $TG(V,E)$ encompasses multiple paths, which the robot can follow to reach destination region from source region. The next objective of offline stage is to find a

path p_i that minimizes time to reach target and coverage area of the site. A utility function $F(p_i)$ is defined which tries to achieve the aforementioned objective:

$$F(p_i) = P_s(p_i) + \hat{p}_i \tag{3.6}$$

where, $P_s(p_i)$ is the probability that robot is successfully able to tread entire path p_i and $\hat{p}_i$ is the number of the regions which lie in path p_i. The probability ($P_s(p_i)$) of robot to reach the destination region q_d traversing a path (p_i) is highly dependent upon its ability to successfully traverse all the regions individually that belong to path p_i. It is assumed that robot's probability (η_{q_i}) to traverse a region q_i with obstruction is 0.5.

$$P_s(p_i) = \prod_{q_k \in p_i} (1 - \eta_{q_i}) \tag{3.7}$$

Thus, a global path p_k is selected using:

$$p_k = \min_{p_i \in P} F(p_i) \tag{3.8}$$

The selection of global path p_k is oblivious to presence of obstacles like debris. This information can be gathered by the robot using sensors at run time. This requires strategy to replan the global path in case of an obstacle blockage.

3.4.2 The Online Stage

The first stage accounts for planning global path as a set of regions using the layout. The second stage then targets to narrow down on local coverage of each region of selected global path p_k. The challenge of directing a robot in real-time can be formulated as iterative identification of prospective uncovered regions, selection of one of them and path planning for robot to reach the selected region. The identification of uncovered regions is conditioned to representation of the traversal site by robot. In this case, it has been achieved by employing frontier-based exploration, which relies on occupancy grid as a representation for navigation to uncovered regions. The occupancy grid maps embody a region of traversal site within a grid. Initially, all the cells are labelled as *'unknown'*. As the robot navigates, occupancy likelihood for each cell is updated using real-time sensor data. Using occupancy likelihood, the cells are labelled as *free, occupied or unknown*. A robot recursively seeks to reach an uncovered region by navigating to the *frontier cells* that are adjoining to free and unknown cells. The segments of *frontier cells* that are spatially adjacent are grouped into *'frontier regions'*.

Algorithm 3: SITE TRAVERSAL

Input:
1. c_r: Start robot position
2. p_k: Selected global path
3. $TG(V,E)$: Road Map

Output: Locate *Target*

begin
  Initialization:
    1) $r = c_r$ ▷ Current robot position, r as start location
    2)f_c: next frontier for coverage
  %FRONTIER SELECTION:%
  while $r \neq$ *STATIC TARGET* **do**
    Obtain an occupancy grid map G for robot position r
    Find t: entry point of next region in p_k
    Find $F = \{f_1, f_2, \ldots, f_y\}$, multiple frontiers in G
    if $F = NULL$ **then**
      All the area covered, No target found
      return FALSE
    $Obj = [\phi]$
      ▷ Store the values of objective function of all the frontiers in F in an array Obj
    for *each frontier* $f_j \in F$ **do**
      Calculate $C(f_j)$ and $O(f_j)$ using Equation 3.12 and 3.9, respectively.
      $Obj[j] = O(f_j)$
    $f_c = \max_{f_j \in F} Obj$
    Find a path $\hat{p_c}$ from r to reach f_c using G
    Move Robot from r to f_c following path $\hat{p_c}$
    $R = f_c$ ▷ Update current robot position
  if $r ==$ *STATIC TARGET* **then**
    Target found
    return TRUE

Region Exploration

The robot builds occupancy grid map G of the environment using GMapping SLAM algorithm [136]. At any given instance, there can exist multiple *frontiers* such that $F = \{f_1, f_2, \ldots, f_y\}$ in an occupancy grid G. The approach selects best frontier among available choices for robot using a Boltzmann-like objective function formulated as:

$$O(f_j) = e^{(-\beta C(f_j))} \tag{3.9}$$

where, f_j is a frontier in F, $C(f_j)$ is the cost function estimating the cost incurred by the robot when traversing from current position to frontier f_j and β is a constant. After selecting the frontier, robot navigates from current position to the selected frontier.

***Calculation of Cost Function* :**

Let $r,t \in \mathbb{R}^2$ be the current position of robot and intermediary target point (which is the entry point of next region q_i' in path p_k), respectively. The distance between two coordinates points x_1 and x_2 is calculated using $D(x_1,x_2)$. $D(x_1,x_2) : \mathbb{R}^2 \rightarrow \mathbb{R}$ is an input operator that computes the euclidean distance between x_1 and x_2. Now, the following parameters are introduced :

$$F_1 = \frac{D(f_j,t)}{\Sigma_{j=1}^{y} D(f_j,t)} \tag{3.10}$$

$$F_2 = \frac{D(f_j,t)}{D(r,t)} \tag{3.11}$$

F_1 and F_2 explore the affinity between frontier f_j and target t with respect to other frontiers and current robot position, respectively. A robot at the location r calculates the cost of all frontiers $\forall f_j \in F$ using:

$$C(f_j) = \frac{w_1.\hat{F_1} + w_2.\hat{F_2} - w_3.\hat{I.G}(f_j)}{1 - e^{-member(f_j)}} \tag{3.12}$$

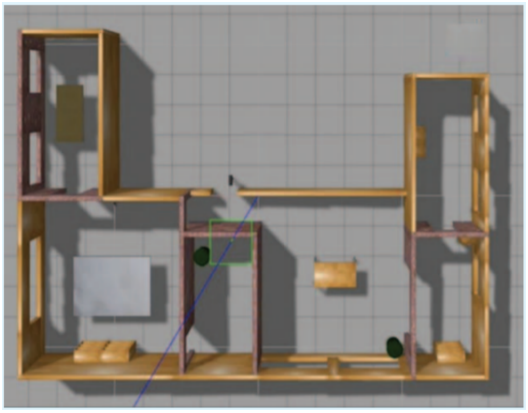
where, w_1, w_2 and w_3 are the user-defined weights that are used for tuning of normalized parameters F_1, F_2 and $I.G$ in cost function according to different layouts. The cost function $C(f_j)$ ensures that the robot selects only those frontiers which lie in the regions of selected path p_k. This restriction has been imposed by employing a membership function $member(f_j)$ as follows,

$$member(f_j) = \begin{cases} \infty, & if f_j \in (q_i \text{ OR } q_i') \\ 0, & otherwise \end{cases} \tag{3.13}$$

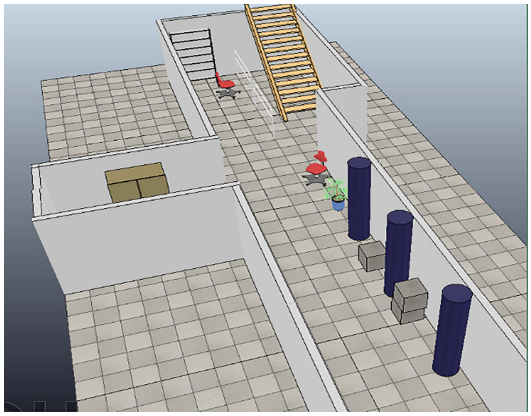
where q_i' is the next region to be traversed after q_i. Due to scale inconsistency, the layout is aligned with respect to grid map G manually[1]. From given set of frontiers, movement $\mathscr{M}$ is calculated to which frontier will yield highest value of information gain. $I.G(f_j)$ estimates amount of information the robot will gain on movement $\mathscr{M}$ to frontier f_j from r such that:

$$I.G(f_j) = H(G|\mathscr{M}) - H(G) \tag{3.14}$$

[1]This process requires human effort which is done only once and takes few minutes and has been adopted in past by many approaches [137]. However, it can be automated as proposed in [138].



(a)



(b)

(c)

Fig. 3.6 Simulated (a) Environment 1: Nuclear Site [139] (b) Environment 2: Debris prone extreme environment [140] (c) Environment 3: Industrial Environment [141].

The movement $\mathscr{M}$ comprises multiple steps spanning over the path ($\hat{p_c}$) [2] between f_j and r. $H(G)$ is the information entropy of probability distribution defined $\forall\ c_{i,j} \in G$

$$H(G) = -\sum_{c_{i,j} \in G} p_o(c_{i,j}) log(p_o(c_{i,j})) \tag{3.15}$$

3.4.3 Re-planning Strategy

The selection of a global path p_k minimises coverage area and time. If the selected path p_k is non-traversable, there is a need for the robot to detour to a different global path selected via following steps:

1. From pre-computed road map, trace path p_k backwards using the regions already covered till it reaches a region with its outdegree[3] greator than 1. This region is labelled as q_i''. Station the robot at entry point of q_i'' .

2. From the current region q_i'', robot can follow a set of possible paths (P') apart from currently traversed p_k to reach destination where:

$$p'_j = ((q_i'', q_{t_1}), (q_{t_1}, q_{t_2}), \ldots, (q_{t_m}, q_d)) \mid \forall p'_j \in P'$$

3. Calculate $F(p'_j)$ using Equation 3.6 and choose a new global path p_k using Equation 3.8, from the region q_i''.

4. Now, the robot selects the next frontier using Algorithm 3 with the updated path p_k.

If there exists no obstacle-free path, the robot returns back to entrance. For instance, if there is an obstruction in state q_5, q_7 and q_9, as illustrated in Fig. 3.4c. Then, the robot re-routes and selects a different path via region q_8.

3.5 Experimental Evaluation

Performance evaluation of the offline-online strategy has been carried out through a comprehensive set of simulation-based experiments. The main motivation behind offline-online strategy is to minimise coverage area and time in online stage for robot operating with time

[2] $\hat{p_c}$ represents the path between two frontiers which the robot follows in runtime while moving along the global path p_k.

[3] In a directed graph, outdegree of a vertex is the number of outward directed edges from a given vertex.

constraints. Thus, the strategy is evaluated using two metrics: coverage time (C_t) required to reach goal and covered area (C_a) which is defined as:

$$C_a = \frac{count(F_r \cup O_r)}{Cell_{tot}} \tag{3.16}$$

where, F_r and O_r are the set of free and occupied cells in the generated grid G, respectively. $Cell_{tot}$ is the total number of cells in the grid and $count(X)$ is a function that outputs cardinality of input set X.

3.5.1 Experimental Setup

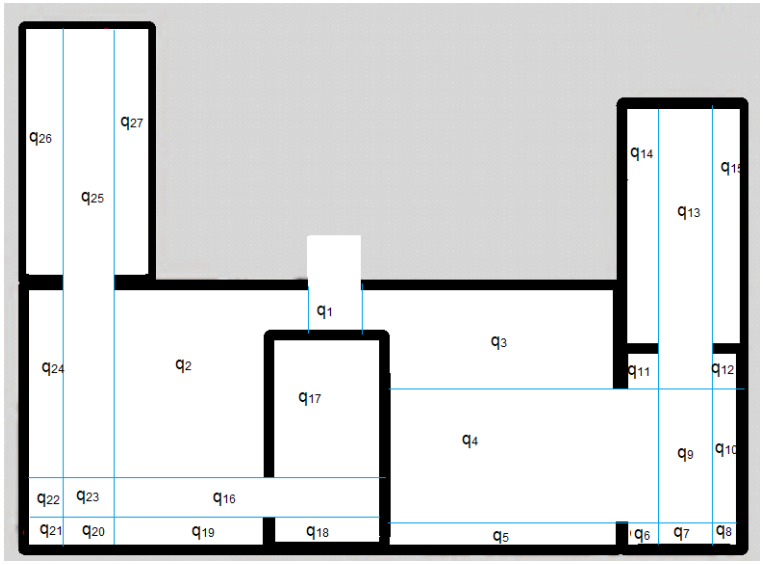
To ensure that the results complement existing evaluations in literature, industrial inspection robot, CARMA [139] is simulated in ROS Gazebo[4]. CARMA, a modified TurtleBot 2 robot, has conducted radiological inspection of largest nuclear facility in Europe, Sellafield as reported in [139]. CARMA is equipped with a sensor package containing floor monitoring and detection dosimeters. It builds occupancy grid map of resolution 0.05 m/grid using Kinect sensor mounted on the robot.

Three indoor environments with different characteristics are considered. The first environment [139] that was used to test radiological robotic inspection is illustrated in Fig. 3.6a. The second environment [140] was used to test a debris clearing robot in extreme environment shown in Fig. 3.6b. The third environment [141] was proposed for Agile Robotics for Industrial Automation Competition 2017 as shown in Fig. 3.6c. However, for the experimentation purpose, location of the goal is varied across the environment. This demonstrates the behaviour of the strategy across multiple scenarios with variation in dimensions of layout. There are twenty GO-CPP operations carried for experimental evaluation. To evaluate robustness of re-planning strategy, random obstacles in the global path are inserted.

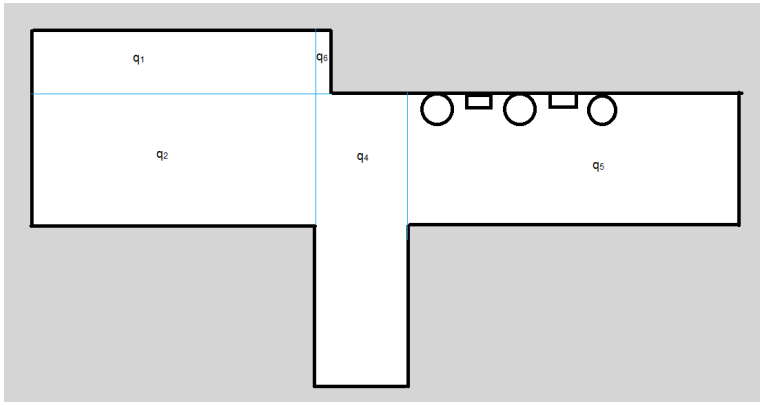
3.5.2 Offline Stage

The proposed approach uses layout as background information to pre-plan area coverage. The layouts of all three environments are segmented into a set of regions using map segmentation algorithm. For the given layouts (refer Fig. 3.6), map is segmented to derive a set of regions Q, illustrated in Fig. 3.7. The map segmentation details for all environments are tabulated in Table 3.1. For all twenty scenarios, a *road map* is computed with Algorithm 2 using the set of regions Q. Few of the computed road maps are illustrated in Fig. 3.8. From the derived road map, a global path is selected using a utility function described in Equation 3.8.

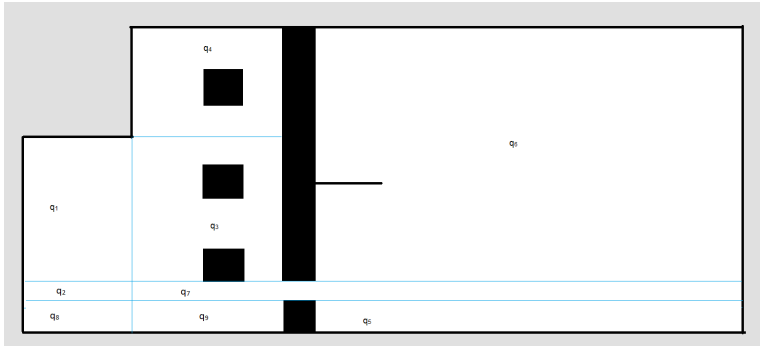
[4]Gazebo is a 3D dynamic simulator



(a)



(b)



(c)

Fig. 3.7 Region segmentation in layout of (a) Environment 1 (b) Environment 2 (c) Environment 3.

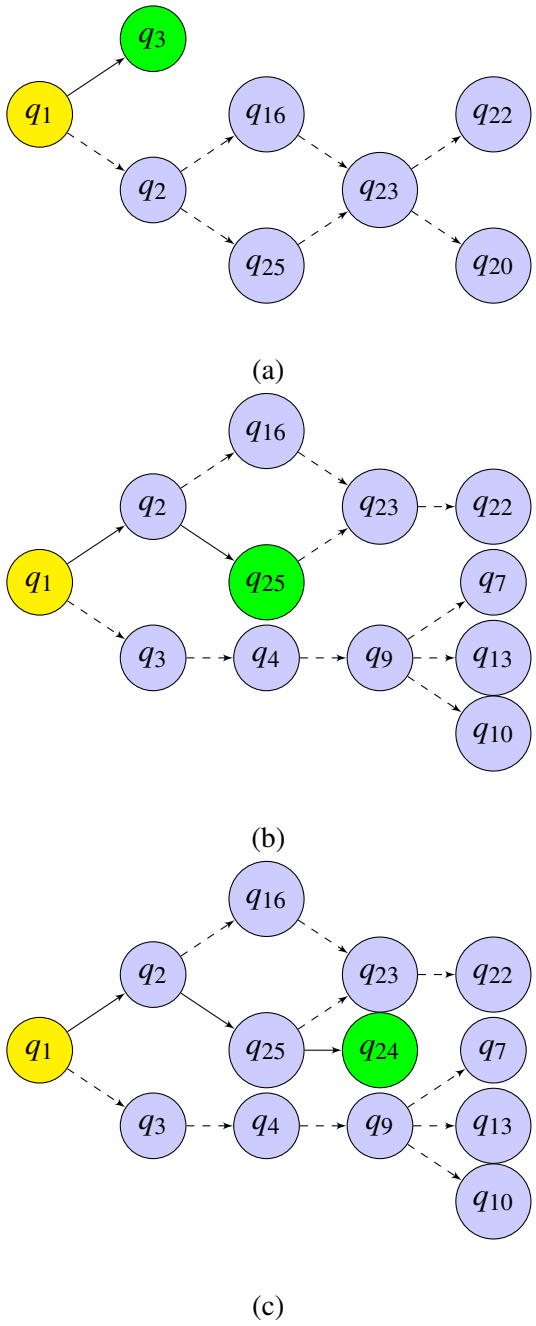


Fig. 3.8 Road map of static target search operation in Environment 1, robot stationed at source region q_1 and static target location in (a) region q_3 , (b) region q_{25}, and (c) region q_{24}.

3.5.3 Evaluation of Coverage Time

The main aim of offline-online strategy is to minimize coverage time (C_t) and coverage area (C_a) for the robot traversing the environments. The global path selected minimizes coverage time using utility function $F(.)$ in Equation 3.8. The utility function $F(p_i)$ for a

Table 3.1 Details of all three environments.

Environment	No. of Regions	Area (m^2)
#1	27	16×10
#2	6	50×12
#3	9	30×20

path p_i depends upon $\hat{p}_i$, number of regions in path p_i. Therefore, both the evaluation metrics are evaluated with respect to $\hat{p}_i$. A total of twenty GO-CPP operations are performed for given environments and for each experiment 20 instances of C_t are recorded. The results are reported after taking average over these 20 distinct runs.

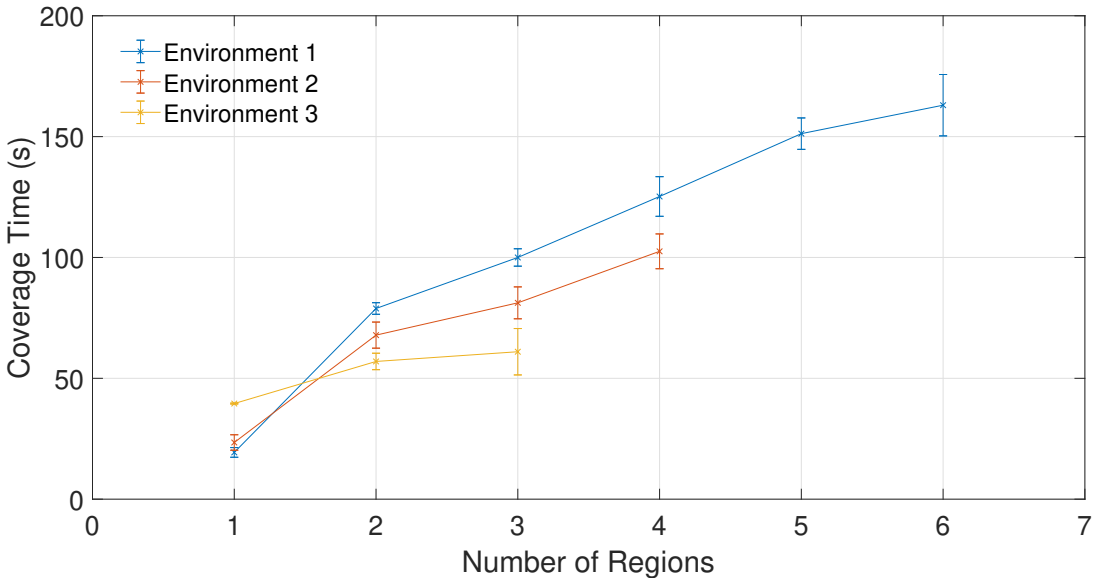


Fig. 3.9 Analysis of coverage time with respect to number of regions.

Fig. 3.9 depicts the experimental observation of coverage time recorded, while executing the proposed approach with respect to number of regions. From the illustrated figure, there is an observed trend that with an increment in number of regions in the global path, the coverage time C_t increases. However, it must be noted that all the regions in a global path can have different areas. For instance, region q_2 and region q_{16} in Fig. 3.7a, even though they are neighbours, but their areas are different. Therefore, it was decided to further analyse C_t in respect to another parameter, $area_{p_i}$.The parameter, $area_{p_i}$ is the ratio of area of global path p_i with respect to the total area of the site.

Fig. 3.10 illustrates C_t of the proposed approach with respect to area of all the regions in global paths. The area of global paths increase as the goal moves farther away from the start location of the robot. Therefore, coverage time to locate the target increases. The increase in $area_{p_i}$ also results in reduction of the number of frontiers penalised in Equation 3.12. To further elucidate the efficacy of offline-online strategy, it was compared with two categories of baseline techniques.

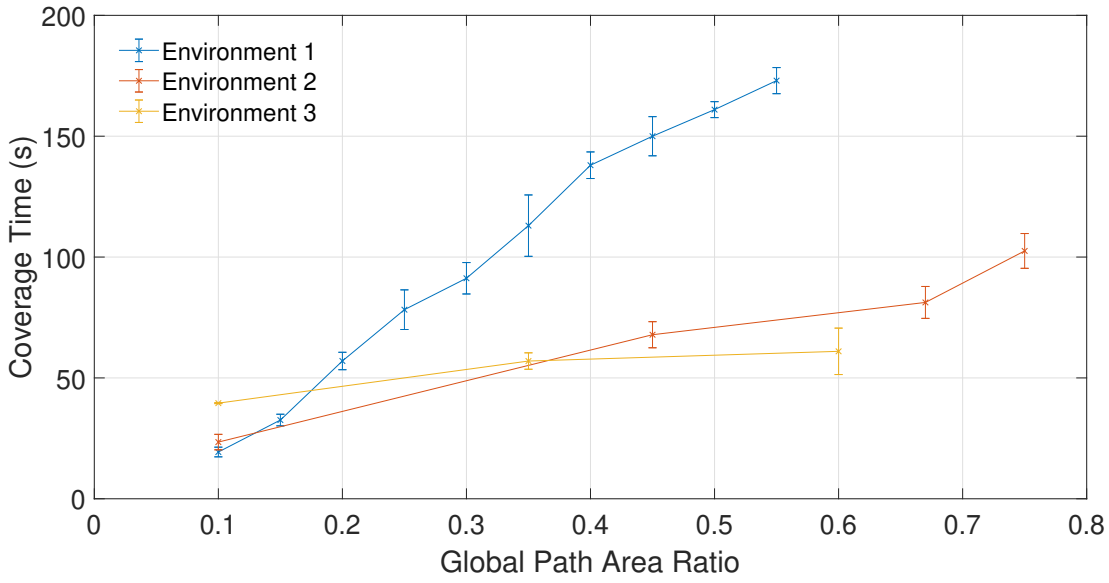


Fig. 3.10 Analysis of coverage time with respect to global path area ratio.

Comparison with existing works:
The performance of the proposed approach was compared with following methods widely used for GO-CPP.

1. Caley et al. [124]: This method spatially models the goal location (exit) and selects frontier closest to the spatial location of the goal.

2. Bird et al. [139]: This method covers the environment in a random fashion, a similar approach has also been implemented in [142].

The experiments are conducted in three different scenarios by varying goal location in the environments. In the first scenario, the robot is required to traverse two regions to reach the goal. As tabulated in Table 3.2, the proposed approach is able to perform significantly better than other two methods. However, the performance of nearest frontier method is comparable to ours. In the second and third scenario, to locate the target, the robot is required to traverse

four and six regions, respectively. As the distance between initial robot position and static target increases, the proposed methods outperforms both the baseline methods. This is due to the fact that geometric layout awareness in GO-CPP operations allows the robot to avoid traversing dead-end loops and corners in turn minimising the time.

Table 3.2 Performance of different methods for static GO-CPP.

Case	Methods	Coverage Time (s)	Coverage Area
1	Proposed	12.307	0.08205
	[124]	18.184	0.110
	[142], [139]	71.769	0.278
2	Proposed	25.023	0.279
	[124]	54.595	0.596
	[142], [139]	251.547	0.773
3	Proposed	45.429	0.601
	[124]	102.57	0.818
	[142], [139]	341.709	0.923

To put things in perspective, the experiments elucidate that minimal coverage time C_t is observed due to geometric-layout awareness in the form of a road map. As the distance between the static goal and starting position of robot increases, the robot is required to cover more area to reach the target location. The performance of proposed approach outperforms the baseline approach in such scenarios. This can be leveraged to the ability of our approach to identify regions as corners and dead-ends which equips the robot to avoid covering such regions. This reduces coverage area which in turn minimizes search time. This versatility of the proposed approach to minimize coverage area has been further emphasised with evaluation of map built during the GO-CPP operation in the next subsection.

3.5.4 Evaluation of Coverage Map

As the robot traverses an environment, it builds an occupancy grid map which is also used for frontier exploration. The objective of proposed approach is to minimize coverage area, thus the experiment use the environment map which robot builds using its sensors during the trip as one of the parameters to evaluate the performance of the proposed approach.

Fig. 3.11 depicts the experimental observations of C_a (refer Equation 3.16), recorded in the proposed approach with respect to varying number of regions. From Fig. 3.11, it is evident that as the number of regions in the global paths increases, the coverage area increases. The increase in number of regions indicates that the distance between static goal

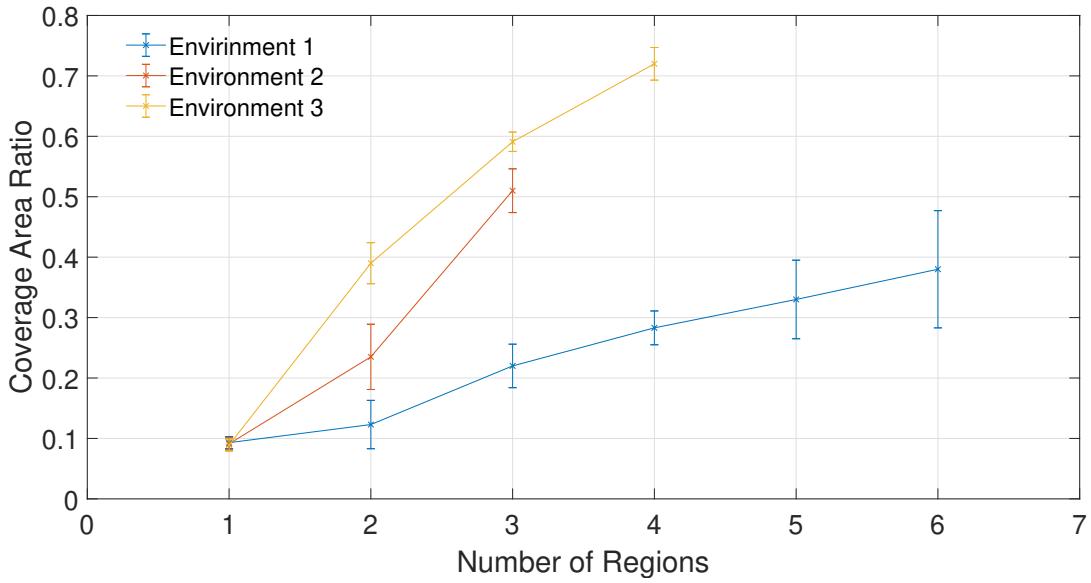


Fig. 3.11 Analysis of Coverage Area for Global paths with different number of regions.

and start position of robot increases which then requires the robot to cover more area. Further analysis of results in the light of ratio of area under global path with respect to total area, ($area_{p_i}$) (refer Fig. 3.12), shows that the increment in offline computed path area also leads to increase in coverage area.

However, if the map generated by the proposed approach is compared with the map generated by the other two methods, the proposed approach outperforms by covering minimum area as evident in Table 3.2. This can be attributed to the fact that selection of frontiers for coverage using the offline computed information allows the robot to traverse regions which are well connected spatially with respect to the coverage path.

3.5.5 Dynamic Obstacle

In previous Section 3.4.3, the re-planning strategy for the robot was discussed when it encounters obstacles in the pre-planned global path. There can exist a scenario where the door is closed or path is blocked due to presence of debris. In this section, the behaviour of the proposed approach in such a scenario was discussed.

An experimental study to evaluate the re-planning strategy is performed by comparing the re-planning strategy with other GO-CPP works where the robot deploys a tedious path re-planning from scratch [143], [138], [144]. The three cases of GO-CPP operations were considered by varying the number of regions in pre-planned path: 1) Case 1: $\hat{p}_i = 4$, 2) Case 2: $\hat{p}_i = 5$, and 3) Case 3: $\hat{p}_i = 6$.

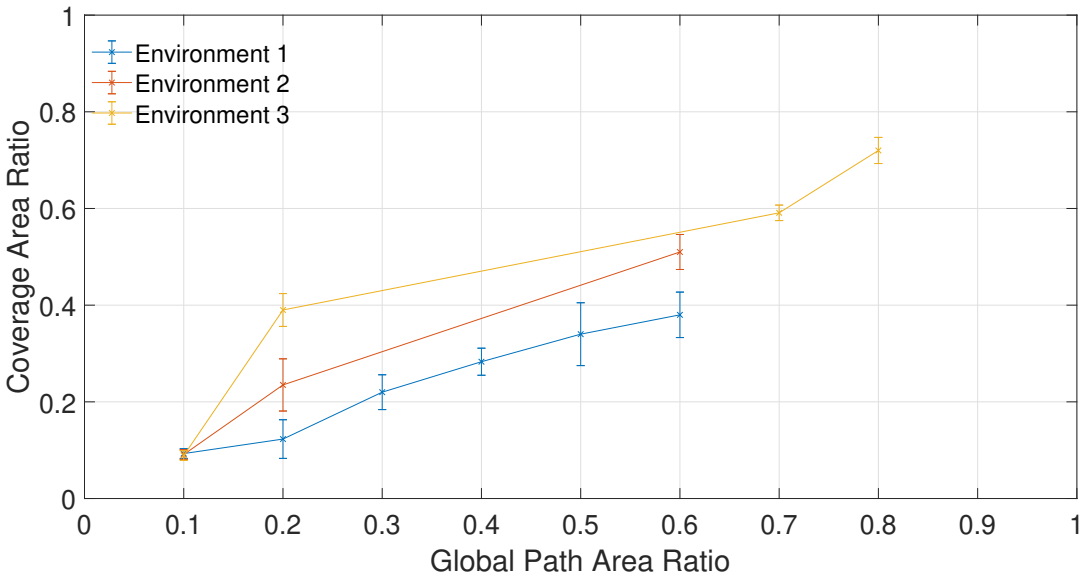


Fig. 3.12 Analysis of ratio of coverage area with respect to total area of global path.

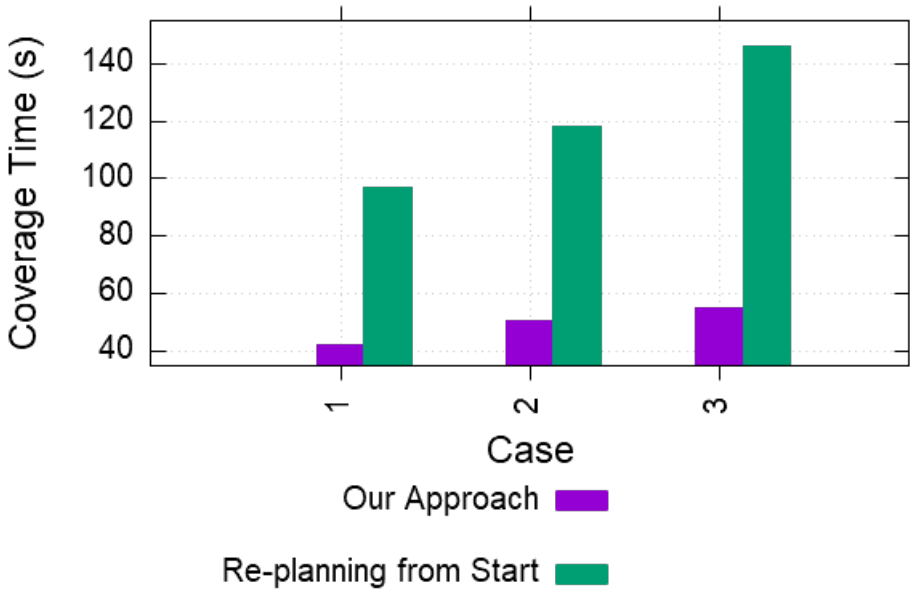


Fig. 3.13 Average coverage time with dynamic obstacles for three cases.

It is observed in Fig. 3.13 that after the robot encounters dynamic obstacles, there is an increment in coverage time C_t. However, the road map built offline allows the robot to traverse to the nearest region and re-plan a new optimal coverage path from a closed

alternative, rather then re-planning tediously from the start location. The experimental studies indicate the robustness of the proposed offline-online strategy for GO-CPP of a static target.

3.6 Summary

In this chapter, a novel offline-online strategy is presented which allows a robot to reach a static goal in any given indoor environment using the architectural layout as a priori information. The proposed approach in offline stage segments the layout into a set of regions and a global path is selected by generating a graph structured, road map, which prevents the robot wasting time traversing irrelevant areas like corners and dead-ends. The robot in online stage treads towards the target following the derived global path by selecting frontiers which minimize the coverage time by covering minimum area. If the robot encounters an obstruction, the re-planning strategy is deployed. The proposed approach was evaluated using three simulated indoor environments in Gazebo. The results show that the robot is able to reach the target in reduced time covering minimum area, which is in line with the requirement of strict time constraints. The derivation of a graph structured road map from layouts increases the geometric layout awareness and assists the robot to reach the target.

However, the proposed strategy's reliance on the architectural layout is a major bottleneck. If the static target is spatially displaced or a geometric error has crept in placement of walls and openings, the roadmap generated by offline stage misguides the robot during area coverage operation. Thus, there is a dire need to provide an updated map as an input to offline stage. The next chapter of this thesis explores techniques to map the environment.

Chapter 4

Coverage Path Planning of Distributed Regions with Precedence Provision

4.1 Introduction

As discussed in the previous chapter, an updated map of the site is a requirement for robot based area coverage. However, the present decade has witnessed an increment in number of sites hosting extreme environments that require immediate mapping. The conditions prevalent in such sites vary significantly and may be characterized by extreme temperature and radiation levels. Thus, the current approaches to compute coverage path in these hostile conditions are not feasible. The area coverage of such sites requires robots to consider the correlation of work providing precedence provision in visiting regions. In this chapter, coverage path planning strategies are proposed which provide precedence provision. To meet the challenges, the problem is divided into two phases: inter-region and intra-region path planning.

In the 'inter-region' path planning of the approach, the site comprising of multiple disjoint regions are modelled as a connectivity graph. Two novel approaches, a Mixed Integer Linear Programming (MILP) and heuristic based techniques, are proposed to generate the ordered sequence of regions to be traversed. In the 'intra-region' path planning of the approach, each region is decomposed into a grid and Boustrophedon motion is planned over each region. The ability of this combined approach to provide complete coverage is proved. An investigative study has been conducted to elucidate the efficiency of the proposed approach in different scenarios using simulation experiments. The proposed approach is evaluated against baseline approaches.

The remainder of the chapter is organised as follows. Section 4.2 provides a detailed overview of problem motivation. The sensor characterisation in the presence of radiation is presented in Section 4.3. This is followed by problem formulation in Section 4.4. Section 4.5 encapsulates a discussion on the proposed inter-region path planning technique. The intra-region path traversal technique is presented in Section 4.6. The experimental evaluation of the proposed method is given in Section 4.7. Finally, the research work is summarised in Section 4.8.

4.2 Problem Motivation

The previous chapters have established that CPP is a well researched problem in literature [9, 11, 71]. However, the research has typically focused on coverage of single region or multiple connected regions. Even though few research works have considered multiple distributed regions but failed to consider sites that can be characterized by multiple disjoint regions and require inclusion of precedence provision, due to the prevailing environmental conditions like high pressure, extreme temperature and presence of radiation.

Recent accidents at such sites like Fukushima Daiichi nuclear meltdown, and Gulf of Mexico oil spill etc. have resulted in environmental catastrophe. This has generated a global demand of periodic coverage of sites with extreme environment. Human inspection of these sites is hazardous and requires a robot to perform this daunting task. The inspection or coverage of an extreme environment site has following salient characteristics which need to be considered while computing coverage path:

1. Disjoint Regions: The precarious sites consist of regions which are spatially distributed and a robot is entrusted to cover each region. Some applications of coverage of disjoint regions are survey of post-disaster scenarios [145], and precision agriculture [109].

2. Correlated Nature of Work: Coverage of the site requires the robot to perform sensing and actuating actions like corrosion check and radiation monitoring [121, 122, 146]. There exists a certain degree of correlation between these actions which impose requirement of visiting regions with a precedence provision. An experimental study is conducted at a radiation laboratory to model effect of radiation on the sensors. The experimental studies revealed that the annealing process starts as soon as radiation exposure is removed. Thus, it can be concluded that after covering regions which have a presence of radiation, the next region must be a non-critical area.

The development of robots that can survey sites has revolutionized the field of surveillance and inspection. The recent [147] robot based fixed altitude CPP problem mapped environment

monitoring problem as a multiple target visiting problem. The multiple targets visiting problem was then formulated as Travelling Salesman Problem (TSP) [148]. TSP generated an optimal path where each target was visited once taking into consideration disjoint regions, however ignoring the constraint of correlated nature of work. Recently, robot *'CARMA'* (Continuous Autonomous Radiation Monitoring Assistance) [149] used sensor-based CPP to inspect Sellafield nuclear facility. However, it failed to consider distributed nature of regions. In other cases, where the robot was assigned to inspect any region [71], [150], CPP problem was formulated as derivation of the optimal path, so that one region was covered considering the precedence provision. Another recent aerial robot-based post-disaster survey had proposed a variant of grid based TSP solution for systematic investigation of site with distributed regions [42]. However, it failed to eliminate the sub tours [151] which is an integral problem of TSP.

Although, aerial robot based coverage has been studied extensively, to the best of the authors' knowledge, none of the works focus on the robot based coverage path planning techniques considering both the salient characteristics. The research in this chapter seeks to bridge this gap, by proposing a coverage path planning technique which covers any site with disjoint regions providing precedence provision.

In this chapter, a strategy is adopted that provides a solution to the problem of coverage of multiple non-overlapping regions with given precedence (see Fig. 4.1). The CPP problem for multiple disjoint regions is first presented as MILP formulation . As the problem is very complex in nature, use of MILP formulation makes the solution approach highly compute intensive, and restricts its scalability to problems containing only a small number of regions. However, the intended application of area coverage requires the robot to cover vast geographical areas, where both the number of regions and their average sizes may be pretty large. This demands a heuristic solution that generates a set of feasible solutions for the coverage path planning problem for disjoint regions.

A scenario is considered where the regions are far apart, therefore the distance between two regions can be approximated as distance between their centroid. The proposed scheme treads a route by initially dividing the problem into two phases: Inter-region path planning and Intra-region path planning. In the first phase, an attempt is made to answer a pertinent question: How can a robot cover the entire site by visiting each region while considering the specified order due to correlated nature of regions. A MILP solution and a construction heuristic, Inter-Region Path Planning (*'IRPP'*) are proposed that generate the order of inter-region traversal. This order serves as an input to the second phase of *'Intra-region traversal'*, where the entry and exit point for each region are calculated on the basis of preceding and succeeding region, in the order of generated inter-region traversal. Deriving this information,

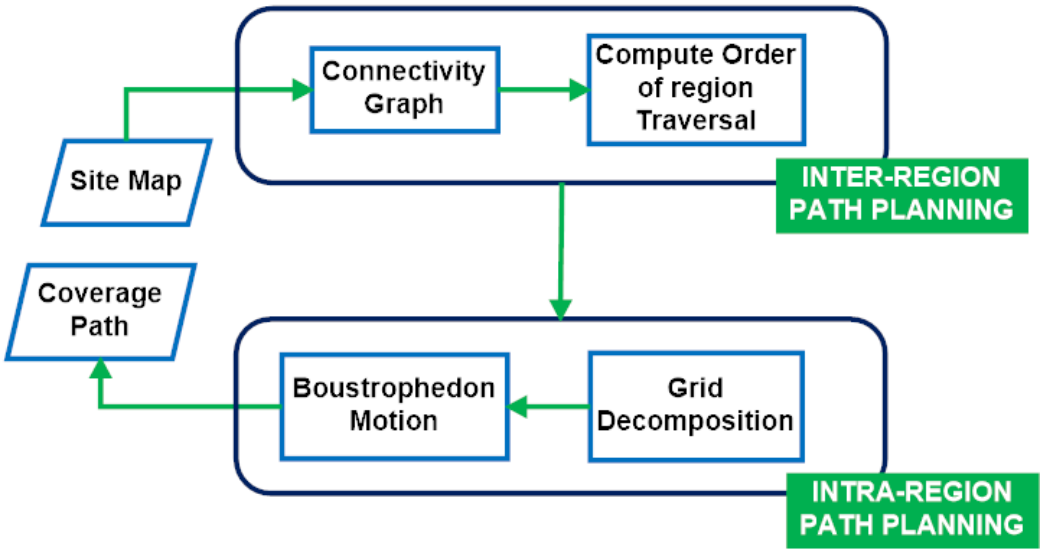


Fig. 4.1 Flowchart illustrating an overview of all the steps required to compute the coverage path.

the second phase first decomposes target site into a grid and then calculates Boustrophedon Motion between entry and exit grid cell for each region.

4.3 Sensor Characterisation

The area coverage of an extreme environmental site is daunting task. It is interlaced with the time intensive challenges and have complex environmental issues. In this section, author tries to address the environmental issue of degradation of sensors during the operation at target site. In turn, leading to breakdown of robots at the site and increasing the radioactive wastes [139]. Therefore, there is a dire need to further model the degradation induced in the sensors. An experimental study is conducted to characterize the radiation induced degradation in the depth sensors as a function of exposure time. The depth sensor chosen for irradiation is Microsoft Kinect which has been used previously by the aerial robots for area coverage [152, 153].

Firstly, the degradation of the Kinect sensor is observed as noise in acquired depth images during radiation experiments. Then, induced degradation is classified as Displacement Damage (DD) and Single Event Effect. The experimental evaluations have been illustrated in Fig. 4.2.This work serves as a building block in assisting aerial robots to perform area coverage in a nuclear sites with disturbed regions in a better fashion.

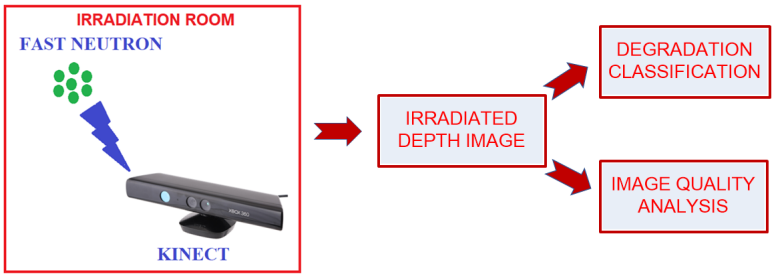


Fig. 4.2 Experimental Evaluation.

4.3.1 Experimental setup

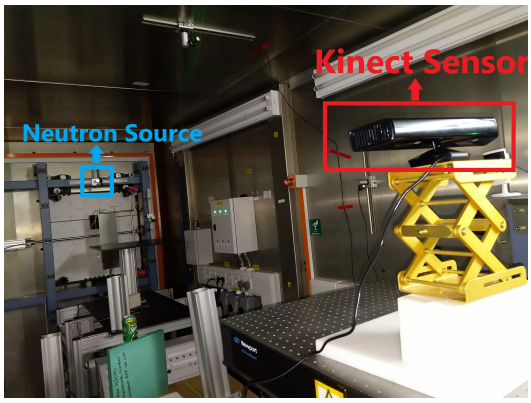
The exposure of the Kinect sensor to fast neutron was carried out by conducting experiments at ChipIr facility in ISIS, Rutherford Appleton Laboratory, Didcot, UK [154]. This facility provides a neutron spectrum which is suitable to emulate effects of terrestrial neutrons in any given electronic sensor. The ChipIr neutron flux (with $10\ MeV < E_n < 800\ MeV$) has been measured to be $5 \times 10^6\ cm^{-2}s^{-1}$. The Kinect sensor was irradiated for about 20 hours at ChipIr, that amounts to 3.5 million years of natural exposure. Fig. 4.3a depicts the experimental setup on the beam lines at ChipIr facility.

The Kinect sensor is capable of capturing RGB, Infra-Red (IR), depth, skeleton and audio streams simultaneously at a given time instance. The IR emitter projects a speckle pattern over the given scene. The reflected pattern is captured using IR camera. This allows the depth of objects in scene to be gauged by correlation [155]. Thus, neutron beam is focussed on IR emitter first for 10 hours. Then, an annealing process at room temperature for a time period of 10 hours is performed. This is followed by exposure of neutron beam focussed at IR camera for another 10 hours.

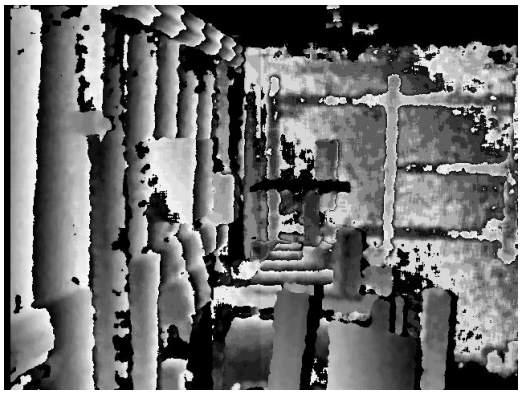
The main objective of this experimental study is to measure the degradation effect on depth images acquired using Kinect due to neutron exposure. The degradation manifest as noises in depth images which is characterized as a function of radiation exposure time. In order to realize this ambition, the noise is estimated first and then classified into two categories: *1) Displacement Damage* (permanent degradation) and *2) Single Event Effect* (transient degradation). The prime requirement is to obtain a reference image that is explained in the next section.

4.3.2 Reference Image

The reference image, I_{ref}, is depth image of static scene captured by the Kinect without the radiation exposure. It is worth noting that there are many noises induced in Kinect depth



(a) Experimental Setup



(b) Reference Image

(c) Radiation Noise

Fig. 4.3 Experimental setup and observed radiation noise after 10 hours of radiation exposure.

images due to multiple factors [155]. The prominent noises in depth images of any static scene are removed.

For a given static scene, the depth values are expected to be steady. However, it was observed that the depth value for object points are unstable in multiple frames varying in

time. This behaviour is temporal in nature and illustrated as salt and pepper noise in depth images [155]. Thus, median filtering was used to remove these noises.

Let I_{NR} be the set of depth images of static scenes collected without radiation. The reference image I_{ref} is obtained using:

$$I_{NR} = \{I_{NR}^0, I_{NR}^1, \ldots, I_{NR}^{\eta-1}\} \tag{4.1}$$

where η is total number of images acquired after time interval t. Any given pixel (i, j) in I_{ref} is obtained:

$$I_{ref}(i, j) = Median\ (I_{NR}^0(i, j), \ldots, I_{NR}^{\eta-1}(i, j)) \tag{4.2}$$

where $i \in \{1, \ldots, m\}, j \in \{1, \ldots, n\}$ are spatial indices and $m \times n$ is the dimension of reference image I_{ref}.

4.3.3 Radiation Induced Noise

After obtaining the reference image, Kinect sensor is irradiated with a neutron beam. Let I_R be a set of depth images captured after Kinect is exposed to the neutron radiation. Each depth map in I_R captures same static scene used in I_{NR}. Then I_R can be formulated as:

$$I_R = \{I_R^0, I_R^1, \ldots, I_R^{k-1}\} \tag{4.3}$$

where k is total number of images acquired after time interval t. The radiation induced noise, N, is calculated for any irradiated depth image I_R^l as follows:

$$N(i, j) = \begin{cases} 1 & |I_R^l(i, j) - I_{ref}(i, j)| > Th \\ 0 & otherwise \end{cases} \tag{4.4}$$

I_{ref} is the reference image calculated using Equation 4.21 with $m \times n$ as the resolution of depth images. Th is the threshold sampling value selected empirically in the experiment.

4.3.4 Classification Technique

Past studies have demonstrated that neutron irradiation of semiconductor materials can dislocate atoms from their normal lattice location. This allows creation of Frenkel pair and divacancy [156]. Due to the disalignment of atoms in crystalline structure, there are two effects observed in depth images which can be classified as:

- Displacement Damage (DD) are set of pixels which undergo permanent damage.

- Single Event Effect (SEE) are set of pixels which undergo transient damage.

To classify both the effects, Algorithm 4 was proposed. The algorithm compares spatial occurrence of noise pixels in depth images over a span of time.

Algorithm 4: Displacement Damage and Single Event Effect

Input:
1. I_{ref}: reference image.
2. I_R : a set of irradiated images with time stamp.

Output: DD, SEE : dispalcement damage and single event effect

Begin:
$L = NULL$;
for *each irradiated depth image* $I_R^l \in I_R$ **do**
    Calculate radiation noise N for I_R^l and I_{ref} using Equation 4.4;
    Store the noise pixel positions in a list L_t created at t;
    $L = L \cup L_t$;
for *each* $L_t \in L$ **do**
    Consider two consecutive lists L_t and L_{t+1}; ▷ where L_{t+1} denotes the list created from image with time stamp $t+1$
    if *a noise pixel appears in both the lists* **then**
        Noise is classified as Displacement Damage;
    else
        Noise is classified as Single Event Effect;

4.3.5 Observations

The evolution of DD and SEE in Kinect depth sensors over the exposure period is shown in Fig. 4.4. The displacement damage rises from being negligible in the initial hours to account for more than half of damaged pixels after four hours of irradiation. This saturates until there is a shift in focus of radiation from IR emitter to IR camera. The annealing process of 10 hour allows Kinect sensor to recover from this displacement damage. After the focus shifts to the IR camera, the progressive increment in displacement damage under neutrons exposure is observed. In contrast, there is an upsurge in Single Event Effect (SEE) observed at initial exposure of neutron irradiation which declines over span of time. This allows us to conclude that when Kinect experiences an initial exposure to neutron there is a higher ratio of noise is SEE in comparison to *DD*. However, with passage of time, SEE decreases and *DD* increases.

The conclusions that can be drawn from this study are that radiation induced degradation saturates after few hours of exposure and annealing process at room temperature triggers recovery of radiation induced degradation. Thus, annealing process is considered as a precedence provision when designing coverage path in upcoming sections.

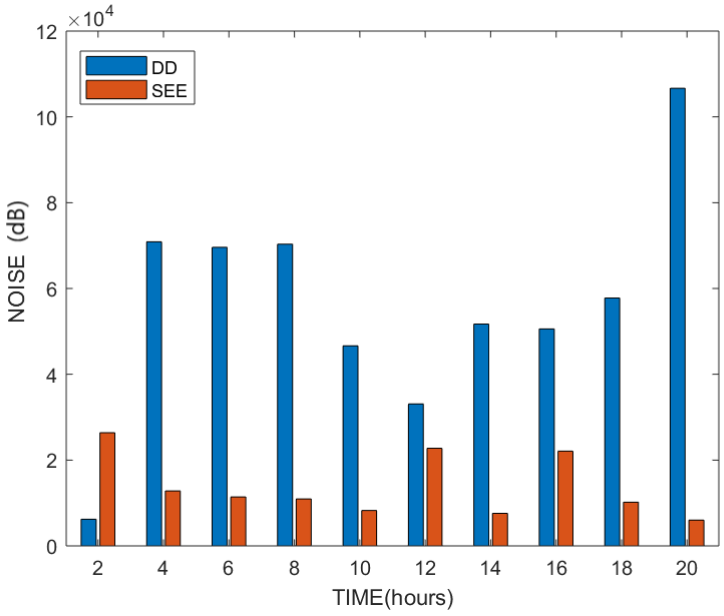


Fig. 4.4 Displacement Damage (DD) v/s Single Event Effect (SEE) observed over time.

4.4 Problem Formulation

In this section, CPP of disjoint regions with precedence provision is described. The optimisation problem is formally stated as a Constraint Satisfaction Problem via Mixed Integer Linear Programming formulation.

4.4.1 Problem Description and Assumption

An aerial robot is equipped with a depth sensor is initially stationed at depot v_0. Then, the robot is dispatched to cover a set of disjoint and spatially distributed rectangular regions, $\mathscr{M} = \{1, \ldots, m\}$. Each region $i \in \mathscr{M}$ is described by its four vertices $(v_{i1}, v_{i2}, v_{i3}, v_{i4}) = V_i$. Generally, in the aerial space, all the regions are connected. However, due to the presence of no-fly zones and exclusion, all the regions are not connected. After covering the entire site, the robot is required to return back to the depot. An assumption that the robot operates at fixed altitude and constant speed. If $\mathbb{R}^+ = \{a \in \mathbb{R} : a \geq 0\}$, the configuration space Ω for the robot $\mathscr{A}$ can be formulated as $\Omega \subseteq \mathbb{R}^2$. The sensing range of the sensor mounted on the robot is $r \times r$. It is assumed that the robot has sufficient power and energy to complete the task. The geographical area of the site has been modelled using a discrete 2D grid structure such that size of each grid cell is designed to be $r \times r$, so that if the robot visits the centre of grid cell, the entire grid cell can be covered. *The objective of the problem is to find optimal*

tour for the robot that starts and ends at the depot, such that robot's sensor footprint along the traversed path covers each target region completely and the total travel cost is minimized. Apart from covering each target location, the robot also has to achieve the goal of traversing the target regions in an ordered sequence.

As described in Section 2.7.1, the type of robot used for coverage also places an additional constraint on the path. For instance, if the robot is a fixed-wing UAV, its mechanical structure determines minimum safe turn radius restricting the robot to make abrupt changes in their direction. In this problem, the robot is a multirotor UAV that is allowed to make sharp turns with arbitrary turn radius, allowing significant increase in the flexibility, while designing the path to be a linear problem.

4.4.2 Notations

Addressing the area coverage problem more formally, the basic notations are introduced. The problem of CPP of disjoint regions can be subdivided into: 1) inter-region path planning 2) intra-region path planning. The intra-region coverage path for each region $i \in \mathscr{M}$ which can be described as P_i. $P_i = \{p_{ik}\}$ is a set of Cartesian coordinate of centre of each grid cells in region i, such that by visiting all the locations in P_i entire region i has been covered, where $k \in [g_i]$, g_i is the set of grid cells located in the region i. The visiting order of the locations in P_i are captured using a decision variable z^i_{kl}. The decision variable $z^i_{kl} = 1$, if the robot traverses from location p_{ik} to p_{il}, otherwise, $z^i_{kl} = 0$.

The entrance location in region i is described by decision variable en^i_k, such that $en^i_k = 1$, if the robot enters region from location p_{ik}, otherwise $en^i_k = 0$. Furthermore, ex^i_k as decision variable to describe the exit location in region i is introduced, such that $ex^i_k = 1$, if the robot exits from region from location p_{ik}, otherwise $ex^i_k = 0$. Finally, to capture the order of visit of the target regions, decision variable x_{ij} is introduced, where $i, j \in \mathscr{M}$, such that $x_{ij} = 1$, if the robot traverses from region i to region j, otherwise $x_{ij} = 0$. The specified order in which location must be traversed is given by $\Theta = \{(i, j) | i, j \in \mathscr{M}\}$. To capture this order, a decision variable y_{ij} is considered such that $y_{ij} = 1$, if the robot traverses from region i to j for specified order $(i, j) \in \Theta$, otherwise $y_{ij} = 0$.

Table 4.1 Variables used in MILP Formulation

Variables	Type of Variables	Definition
$\mathscr{M}$	Auxiliary	set of regions
P_i	Auxiliary	set of waypoints in region i
g_i	Auxiliary	grid cells of region i
Θ	Auxiliary	Specified Region Order
z^i_{kl}	Decision	Visiting order of intra-region waypoints
en^i_k	Decision	Entrance point of region i
ex^i_k	Decision	Exit point of region i
x_{ij}	Decision	inter-region Visiting order
y_{ij}	Decision	Specified Visiting order

4.4.3 Mixed Integer Programming Formulation

With the notations described in above section, the coverage path that covers the entire target site with precedence provision is described. The total travel cost C can be calculated as:

$$\begin{aligned} C = &\sum_{i=1}^{m}\sum_{j=1,j\neq i}^{m}\sum_{k=1}^{g_i}\sum_{l=1}^{g_j} x_{ij} ex^i_k en^j_l D(p_{ik}, p_{jl}) \\ &+ \sum_{i=1}^{m}\sum_{k=1}^{g_i}\sum_{l=1,l\neq k}^{g_i} z^i_{kl} D(p_{ik}, p_{il}) + \sum_{(i,j)\in\Theta} \hat{P}(1-y_{ij}) \end{aligned} \tag{4.5}$$

where function $D(x,y)$ calculates the travel cost from location x to y and $\hat{P}$ is the penalty imposed, if the robot does not traverse regions in the specified order Θ. To ensure the validity of the tour, the following constraints are imposed:

$$\sum_{j=1,j\neq i}^{m} x_{ij} = 1,\ \forall i \in \mathscr{M} \tag{4.6}$$

$$\sum_{i=1,i\neq j}^{m} x_{ij} = 1,\ \forall j \in \mathscr{M} \tag{4.7}$$

$$x_{ij} \geq y_{ij} \qquad \forall (i,j) \in \Theta \tag{4.8}$$

$$\sum_{l=1,l\neq k}^{g_i} z^i_{kl} = 1 - ex^i_k,\ \forall i \in \mathscr{M}, k \in g_i \tag{4.9}$$

$$\sum_{k=1,k\neq l}^{g_i} z^i_{kl} = 1 - en^i_l,\ \forall i \in \mathscr{M}, l \in g_i \tag{4.10}$$

$$\sum_{k=1}^{g_i} en_k^i = 1,\ \sum_{k=1}^{g_i} ex_k^i = 1,\ \forall i \in \mathscr{M} \tag{4.11}$$

$$x_{ij}, y_{ij} \in \{0,1\},\ \forall i,j \in \mathscr{M} \tag{4.12}$$

$$z_{kl}^i, en_k^i, ex_k^i \in \{0,1\},\ \forall i \in \mathscr{M}, k,l \in g_i \tag{4.13}$$

$$en_k^i + ex_k^i \leq 1,\ \forall i \in \mathscr{M}, k \in g_i \tag{4.14}$$

In the above mentioned constraints, Equation 4.6 - 4.7 ensure that each region is covered by the robot exactly once. The constraint in Equation 4.8 ensures that the specified order edge in Θ is visited. Equation 4.9 - 4.10 ensure that each location $p_{ik} \in P_i$ in region i is traversed only once. Equation 4.11 ensures that for each region there is only one entrance and exit. Equation 4.12 - 4.14 ensure that the decision variables take valid values.

The coverage problem can thus be formulated as:

$$\begin{aligned} &minimize \quad C \\ &subject\ to: \ Equation\ 4.6 - 4.14 \end{aligned} \tag{4.15}$$

4.4.4 Discussion

The problem of coverage of disjoint region with precedence provision can be modelled as an optimization problem of integrated Chinese Postman Problem-Coverage Path Problem. Analysis shows that both the problems are NP-hard in the strong sense [157, 71] and an optimal solution strategy to it will be prohibitively expensive in terms of solution generation times and required storage space. Further, it is also difficult to design a deterministic/greedy heuristic strategy which avoids solution enumeration, but can still deliver satisfactory outputs under all realistic scenarios. In past, such complex optimization problem were solved by decomposing into co-related problems [158]. This problem is solved by decomposing into two problems: 1) Inter-region path planning 2) Intra-region path planning.

1. *Inter-region* path planning: It aims to find the order of sequence $S(\mathscr{M}) = \{i, j, \ldots, k \mid \forall i, j, k \in \mathscr{M}\}$ in which the robot visits all the regions traversing all the specified edges denoted by $(i, j) \in \Theta$.

2. *Intra-region* path planning: It can be stated as path computation such that each and every point in any given region i is covered by the robot $\mathscr{A}$.

Note: If the size of each region $i \leq r \times r$ (sensor's range), the area coverage problem is reduced to only inter-region path planning, where each region $i \in \mathscr{M}$ can be described by its centre $\hat{v}_i$. If there is only one region i whose size is larger than sensor's range $r \times r$, the problem is reduced to intra-region path planning.

4.5 Inter-Region Path Planning

In this section, the problem of inter-region path planning is solved. To compute the inter-region path, the site is represented as a connectivity graph $\mathscr{G}$. The connectivity graph $\mathscr{G} = \{\mathscr{V}, \mathscr{E}\}$ is an Eulerian graph, where $\mathscr{V} = \{i, j, ..k | \ \forall i, j, k \in \mathscr{M}\}$ and $\mathscr{E} = \{e_i | \ e_i$ is cost of traversal between region $i-1$ and region $i\}$. The graph is computed in following subsection.

4.5.1 Connectivity Graph

Any region $i \in \mathscr{M}$ is defined by four vertices $\{v_{i1}, v_{i2}, v_{i3}, v_{i4}\}$. Using these four vertices, the coordinates of the centre of a region i, $\hat{v}_i$ can be derived using:

$$\hat{v}_i{}^x = v_{i4}^x + \frac{|v_{i4}^x - v_{i3}^x|}{2} \tag{4.16}$$

$$\hat{v}_i{}^y = v_{i4}^y + \frac{|v_{i4}^y - v_{i1}^y|}{2} \tag{4.17}$$

The centre $\hat{v}_i$ is used to describe the region i, such that $\mathscr{V} = \{\hat{v}_i \mid \ \hat{v}_i \ is \ the \ centre \ of \ the \ region \ i \ \in \ \mathscr{M}\}$. To address the problem of inter-region path traversal, the degenerated edges of the graph are eliminated first, which connect a vertex to itself also known as a self-loop. The judicious elimination of self-loops helps in realization of the goal trajectory minimization. After the elimination of self-loops, the cost which the robot $\mathscr{A}$ will incur in traversing between two regions is determined. The cost of traversal for the robot $\mathscr{A}$ for traversing from region i to region j is defined as $c_{i,j}$ and is computed as follows:

$$c_{i,j} = A^* \ distance \ between \ \hat{v}_i \ and \ \hat{v}_j \tag{4.18}$$

The robot $\mathscr{A}$ can traverse the next region j from current region i iff region i is the neighbour of region j. It is assumed that the generated graph $\mathscr{G}$ contains an Eulerian trail, i.e. there exists a cycle where each edge can be traversed only once. This is illustrated in Algorithm 5, where the cost of all the neighbours of region i is calculated. The calculated inter-region traversal cost (c) and specified possible set of traversal (Θ') will be used as the input to MILP-based inter-region traversal strategy discussed in the next section.

4.5.2 MILP Based Inter-Region Traversal Strategy

In this section, a *MILP* solution for inter-region path traversal problem is proposed. To address this problem, two binary decision variable x_{ij} and y_{ij} are considered. The decision variables are defined as:

Algorithm 5: CONNECTIVITY GRAPH

Input:
1. $\mathscr{M} = \{1,..,m\}$: set of m regions
2. $\Theta = \{(i,j) \mid i,j \in \mathscr{M}\}$: order of sites to be covered
3. N: a large positive number

Output: Θ', c : Set of specified possible region traversal, cost of inter-region traversal

```
begin
    INITIALIZATION:
        1) Θ' = Θ                          ▷ Θ' denotes modified ordered site traversal set
    %SELF-LOOP ELIMINATION%
    for each region i ∈ M do
        Calculate C(f_j) and O(f_j) using Equation 5.8 and 3.9, respectively.
        Obj[j] = O(f_j)
        Centre of region i calculated using Equation 4.16, 4.17
        c_{i,i} = N    /*Assign the cost of self-loop (i,i) as N to eliminate self loop traversal*/
        if exists a self loop (i,i) ∈ Θ' then
            Θ' = Θ' ∩ (i,i)        /*Remove the self loop (i,i) from Θ'.*/
    COST MATRIX CALCULATION:
    for each region i ∈ M do
        for each neighbour region j of i do
            Assign c_{i,j} as A* distance between centre v̂_i and v̂_j)
            /* Determine the cost of traversal from region i to region j*/
```

1. $x_{ij} = 1$ if the robot $\mathscr{A}$ traverses region i to region j, otherwise $x_{ij} = 0$.
2. $y_{ij} = 1$ if the robot $\mathscr{A}$ traverses region i to j for the specified edge $(i,j) \in \Theta'$, otherwise $y_{ij} = 0$.

Objective: The area coverage problem can be formulated as finding values of x_{ij} and y_{ij}, $\forall i,j \in \mathscr{M}$, such that the path starts and ends at the start region (also called depot)), each region is covered exactly once by the robot, and the total cost, *Cost*, given below is minimized.

$$Cost = \sum_{i=1}^{m}\sum_{j=1}^{m} c_{ij}x_{ij} + \sum_{(i,j)\in\Theta'} \hat{c}_{ij}(1-y_{ij}) \tag{4.19}$$

where c_{ij} is the travelling cost incurred by the robot $\mathscr{A}$, if it travels from region i to j, obtained using Algorithm 5 and $\hat{c}_{ij}$ is the penalty imposed on the robot $\mathscr{A}$ if it does not traverse region $(i,j) \in \Theta'$. The objective of the formulation can be written as:

$$minimize\ Cost \tag{4.20}$$

The first term in Equation 4.19 calculates the total travel costs for inter-regional movements. The last term calculates the penalty incurred if the order of sequence defined in Θ' is not followed by the robot $\mathscr{A}$. The validity of the required path traversal is ensured by imposing following constraints:

1. **Departure Constraint**: The constraint in Equation 4.21 ensures that the robot $\mathscr{A}$ leaves each region only once.

$$\sum_{i \in \mathscr{M} \setminus i \neq j} x_{ij} = 1, \quad \forall j \in \mathscr{M} \tag{4.21}$$

2. **Arrival Constraint**: The constraint in Equation 4.22 ensures that the robot $\mathscr{A}$ arrives only once in each region.

$$\sum_{j \in \mathscr{M} \setminus i \neq j} x_{ij} = 1, \quad \forall i \in \mathscr{M} \tag{4.22}$$

3. **Continuity Constraint**: The constraint in Equation 4.23 ensures that the inter-region path traversed by the robot $\mathscr{A}$ is continuous.

$$\sum_{j \in \mathscr{M}, i \neq j} x_{ij} = \sum_{j \in \mathscr{M}, i \neq j} x_{ji} \tag{4.23}$$

4. **Direct Path Constraint**: The constraint in Equation 4.24 ensures that the next region j to be traversed from region i is connected.

$$x_{ij} = 0, \quad \forall i, j \in \mathscr{M} \mid \exists !\{\text{direct path between i and j}\} \tag{4.24}$$

5. **Specified Order Constraint**: The constraint in Equation 4.25 forces variable y_{ij} to take value 0, if not traversed by the robot, i.e. $x_{ij} = 0$. Conversely, if the robot takes the route (i, j), i.e, $x_{ij} > 0$, the minimization of the objective function forces variable y_{ij} to take value 1.

$$x_{ij} \geq y_{ij} \qquad (i, j) \in \Theta \tag{4.25}$$

6. **SubTour Elimination:** Another major issue which path planning problems suffer from, is the ability of the robot $\mathscr{A}$ to get stuck in a subtour cycle in a graph $\mathscr{G}$, which has not been considered in previous coverage works [42],[149]. An additional decision variables u_i is introduced in this constraint to order all regions excluding the depot (starting region of robot) to prevent illegal cycle formations. This is ensured by

constraining $u_j \geq u_{i+1}$ when $x_{ij} = 1$

$$u_i + 1 \leq u_j + N * (1 - x_{ij}) \tag{4.26}$$

where, N is a large positive number.

4.5.3 Example 1: Illustrating MILP Strategy

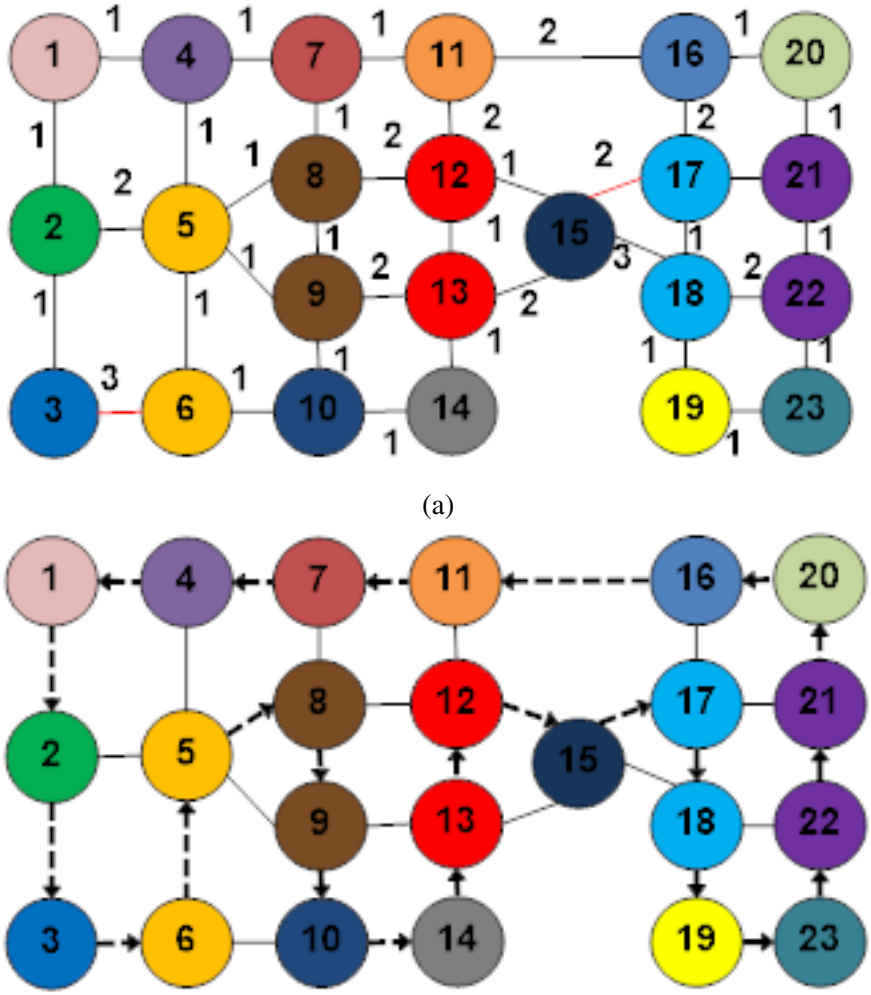
This section elucidates the working of the proposed MILP strategy using an example site of a spaceport. The given site resembles a site used in [159] comprising of 23 regions. Each unit region is a vertex v_i of the connectivity graph $\mathscr{G} = \{\mathscr{V}, \mathscr{E}\}$, shown in Fig. 4.5a.

Let us assume that robot $\mathscr{A}$ operates at a fixed altitude and each region is inter connected by a unit distance such that $\forall j,k\ c_{j,k} = 1$. The traversal from one region to another generates a cost which is shown on each edge (see Fig. 4.5b). This assumption will be relaxed in Section 4.7. As per MILP formulation discussed in Section 4.5.2, the CPLEX [160] solver generates the following optimal tour as illustrated in Fig. 4.5b. The specified order route for coverage is $\Theta = \{(3,6),(15,17)\}$. The generate path is: $\{1->2->3->6->5->8->9->10->14->13->12->15->17->18->19->23->22->21->20->16->11->7->4->1\}$. By traversing the generated path, the robot incurs a cost of 27 units.

From Fig. 4.5b, it is evident that path generated for the robot $\mathscr{A}$ has following salient features to ensure its validity:

1. Depot: The robot $\mathscr{A}$ starts and ends at the same region (depot, region = 1).
2. SubTour: Apart from the cycle which starts and ends at depot, there are no other cycle present in the path.
3. Specified Route: The robot traverses all the edges in the specified route $\Theta = \{(3,6),(15,17)\}$.
4. Trajectory Minimization: The constraint of time and energy is met, as no region is traversed more than once in the generated path.

Therefore, it is evident from the given example that the MILP strategy is able to generate a valid path which meets all the specified constraints of the area coverage problem in this chapter.



(a)

(b)

Fig. 4.5 Site depicted using (a) Inter-region connectivity graph $G(V,E)$ where red edge $\{(3,6),(15,17)\} \in \Theta$ denotes specified order (b) CPLEX generated inter-region path denoted using dotted lines

4.5.4 IRPP: Inter Region Path Planning Technique

The MILP-based inter-region path planning solution described in the previous section generates optimal solution. However, its exponential computational complexity makes it expensive, as the number of regions increases. Therefore, in this section, an efficient heuristic algorithm, *'IRPP'* is proposed, which generates acceptable solution to the problem even when the number of regions are fairly large. The pseudocode of IRPP is presented in Algorithm 6.

The algorithm is a heuristic approach which attempts to cover the specified order route $(i, j) \in \Theta'$ first. This in turn minimizes the traversal cost, as the penalties for not traversing arcs $(i, j) \in \Theta'$ are not imposed. It also takes into consideration that self loop and subtours are eliminated. An example at the end of this section is illustrated to elucidate the proposed approach.

IRPP Strategy

The main motivation of the inter-region path planning heuristic is to determine the order of region traversal such that:

1. Minimum cost of traversal is incurred.
2. All the regions are traversed once and robot $\mathscr{A}$ starts and ends at depot.
3. Specified order of traversal is the route under taken by the robot $\mathscr{A}$.

The algorithm comprises of four blocks: 1) Initialization, 2) Order Generation, 3) Feasible Solution, and 4) Cost Calculation. Each block is discussed in details as follows:

Initialization (Line 1-4): In this block, four variables: m, χ, n, Z are initialized. Variable m and n denote number of regions and number of specified edges, respectively. χ and Z are the arrays containing the order of region traversal which are initialized as *NULL*.

Order Generation (Line 5-7): To meet the three requirements, IRPP heuristic attempts to start the route by traversing the arcs $(i, j) \in \Theta'$ in parallel. As shown in line 5-7, in the order generation block from Algorithm 6, an order of region traversal Z_q for each arc $q \in \Theta'$ is generated by using function *NEXT_REGN ()*. Since, there is no correlation between paths generated for each arc $q \in \Theta'$, each order generation is computed in parallel.

Feasible Solution (Line 8-20): After obtaining a list of order of inter-region traversal Z from last block, the algorithm searches for all feasible solutions among the generated solutions of inter-region, Z. The solution is classified as feasible if, and only if, it meets the second motivation, i.e. each region is traversed once and starting and ending region are same. To determine the feasible solution, it is checked that the generated order contains all the m

distinct regions and the starting and ending region are identical. Those solutions which are unfeasible are discarded in line 18. This is helpful in saving the computation time in the next block, the *cost calculation* block.

Cost Calculation (Line 21-25): In the cost calculation block, the cost of traversal which the robot $\mathscr{A}$ will incur if it takes the feasible route, is determined. The cost of traversal is calculated in line 22, Algorithm 6 by summing the individual cost of arcs in the feasible order of traversal. In line 24, a feasible solution is selected from Z which has minimum cost. In the selected minimum cost feasible solution, the order of traversal begins with the region i for arc $(i, j) \in \Theta'$. However, for the scenario, order of traversal must begin from the depot. Thus, the computed order of traversal is re-ordered in a manner such that traversal of route begins from the depot.

Next Region Function: NEXT_REGN ()

IRPP calls function NEXT_REGN () to generate the order of traversal for all the arc $q = (i, j) \in \Theta'$. NEXT_REGN () returns the feasible solution Z_i. The function is composed of three components: *VSRF, NNV and TERMINATION*. The pseudocode for NEXT_REGN () is presented in Algorithm 7.

VSRF: Visit Specific Route First (Line 1-3): In the VSRF phase, the robot $\mathscr{A}$ visits the specified route $(i, j) \in \Theta'$ first. After reaching point j, it must take a decision to visit any one of the neighbours of j which is taken in the next phase, *NNV*. However, a decision point variable $next_{region}$ is initialized to j in this phase. Another variable list ρ_{cost} which assists in taking the decision of which next region to traverse after j is initialized to NULL. ρ_{cost} maintains the cost incurred for traversing each neighbour k after region j.

NNV: Next Neighbour Visit (Line 4-18): The phase next neighbour visit is responsible for taking the decision of traversing which neighbour of the current region j. First, all those neighbours of j, which have not been previously visited are computed. From the unvisited neighbours, the heuristic can either execute the **look ahead plan** or the **cost calculative plan**. The robot $\mathscr{A}$ finds out whether one of following two criteria is met:

1. **CRITERION I:** Neighbouring region k has an arc $(j,k) \in \Theta'$, i.e. by traversing route (j,k) it meets the specific coverage order in Θ'.

2. **CRITERION II:** Neighbour l of neighbouring region k has an arc $(k,l) \in \Theta'$, i.e. by traversing route (k,l) it meets the specific coverage order in Θ'.

Even if one of the criteria is met, a look-ahead plan is executed, where the robot traverses region k, irrespective of cost, to meet the stringent criteria of specified route travel. In case

Algorithm 6: IRPP

Input:
1. $\mathscr{M} = \{1,..,m\}$: set of m regions
2. $\Theta' = \{(i,j) \mid i,j \in \mathscr{M}\}$: order of sites to be covered
3. $c_{m,m}$: cost of $\mathscr{M}$ inter-region traversal

Output: $Cost, \chi$: cost and order of inter-region traversal

begin
- **INITIALIZATION**
  - 1)$m = |M|$
  - 2) $\chi = NULL$
  - 3) $n = |\Theta'|$
  - 4) $Z = NULL$ /*χ, Z denotes order of inter-region traversal and sequence starting from coverage edge, respectively.*/
- **ORDER GENERATION**
- **for** *each arc* $q \in \Theta'$ *in parallel* **do**
  - Z_q = NEXT_REGN(q); /*Order of traversal Z_q starting from arc q*/
- **FEASIBLE SOLUTION**
- **for** *each solution* $Z_q \in Z$ **do**
  - **if** *Order of traversal in* Z_q *contains all m distinct regions* **then**
    - **if** Z_q *finishes at the start region* **then**
      - Z_q solution is Feasible
    - **else**
      - Z_q is not feasible solution;
      - Discard Z_q;
  - **else**
    - Z_q is not feasible solution;
    - Discard Z_q;
- **COST CALCULATION**
- **for** *each feasible solution* $Z_q \in Z$ **do**
  - $cost_{Z_q} = \sum_{(j,k)\in Z_q} c_{j,k}$;
  - /*Calculate the cost $cost_{Z_q}$ as sum of the cost of all the arcs j in the feasible solution Z_q*/
- $Cost = min\{cost_{Z_1}, cost_{Z_2}, \ldots, cost_{Z_q}\}$;
- /*Assign $Cost$ with minimum cost solution in Z*/
- χ = re-order the minimum cost traversal route such that inter-region tour begins from depot;

none of criteria of route travel are met, cost calculative plan is executed where the next neighbour to traverse is selected based on the cost of traversal from region j to k, $c_{j,k}$. The cost for each neighbour k of current region j is stored in the previously defined list ρ_{cost} as follows:

$$\rho_{cost} = \rho_{cost} \cup c_{j,k} \tag{4.27}$$

The main challenge stems from the equal cost of traversal of all the neighbours. For instance, if the robot traverses the edge $(3,6)$. If the robot executes the policy of selecting vertex based on its occurrence first which is generally employed in many heuristics like next neighbour first [161], the path generated contains cycle or sub tours. To overcome this challenge, *'Away From Sink'* strategy is deployed. Since, the robot needs to traverse the entire tour and return to the starting region, the start region is labeled as the *'sink'*. The *'Away from the sink'* strategy allows the robot to traverse the neighbour which is farthest from the sink, so that all the regions are traversed once, without the presence of any subtours.

TERMINATION (Line 19-26): In this phase, it is determined whether in the last phase the robot $\mathscr{A}$ finds that all the neighbouring regions have been previously visited. This is indicative of the fact that the tour is complete. However, if the robot visits unvisited neighbouring regions and computes ρ_{cost} list, then the neighbour k is selected to be visited which has minimum cost $c_{j,k}$ in the ρ_{cost} list. Based on the selected region k, a new arc (j,k) is added to the solution Z_q. To further elucidate the working of the proposed heuristic, an example in the following subsection is presented.

4.5.5 Example 2: Illustrating working of IRPP heuristic

The construction heuristic *'IRPP'* is illustrated through a working example in this subsection. Let us consider the same extreme environment site as previously discussed in Section 4.5.3, i.e. number of regions $m = 23$ and specified order $\Theta = \{(3,6),(15,17)\}$. As per the *'VSRF: Visit Specific Route First'* block of Algorithm 7, the robot will first traverse edges in the specified order, i.e. $(3,6)$ and $(15,17)$.

The cost of traversal from one vertex to another is equal. For instance, if the robot traverses the edge $(3,6)$. As per Fig. 4.5a, by being stationed at vertex 6, the robot $\mathscr{A}$ has an option to traverse either vertex 5 or 10. Since the cost of traversal between neighbouring vertex is a unit cost, it takes the policy of *'away from sink'*. The path generated for robot $\mathscr{A}$ is $\{1->2->3->6->10->14->13->12->15->17->18->19->23->22->21->20->16->11->7->8->9->5->4->1\}$, as shown in Fig. 4.6.

The *'away from sink'* is executed when the robot traverses the path $(3->6->10)$ and is stationed at vertex *'10'*. As illustrated in Fig. 4.5a, vertex *'10'* has two equal cost neighbour vertices *'9' and '14'*. If the robot uses the traditional next neighbour methodology, it would have selected region *'9'*, which will result in generation of a cycle in the path. Since, the robot starts from the region *'3'* and the tour will end by traversing all the regions once and end at the region *'3'*. The region *'3'* is classified as sink vertex.

Algorithm 7: NEXT_REGN

Input:
1. $\mathscr{M} = \{1,..,m\}$: set of m regions
2. $\Theta' = \{(i,j) \mid i,j \in \mathscr{M}\}$: order of sites to be covered
3. $c_{m,m}$: cost of M inter-region traversal

Output: Z_q: order of inter-region traversal

Function *NEXT_REGN:*
        VSRF: Visit Specific Route First
  Visit the arc q using VISIT_ARC(q);
  Initialize decision point $next_{region}$ as j;
/* $next_{region}$ *denotes the next region to visited after j*/
  Initialize the neighbour cost list ρ_{cost} as $NULL$;
        NNV: Next Neighbour Visit
for *each neighbor k of j which are unvisited* **do**
  l : neighbours of k;
  if *CRITERION I OR CRITERION II satisfied* **then**
    /*****LOOK-AHEAD PLAN****/
    Assign k as $next_{region}$;
  else
    /*****COST-CONSTRUCTIVE PLAN*****/
    if *(traversal of* (j,k) *is the minimum cost among all the neighbours)* **then**
      Add $c_{j,k}$ to the ρ_{cost} list
      /****AWAY FROM SINK***/
      if *k is farthest from sink vertex* **then**
        Add $c_{j,k}$ to the ρ_{cost} list

            TERMINATION
if *no new neighbour visited* **then**
  /**robot HAS COMPLETED THE TOUR**/
  Z_q;
  /*Return generated solution Z_q*/
  STOP;
else
  if $\rho_{cost} \neq NULL$ **then**
    Assign $next_{region} = k$ such that k has minimum cost in ρ_{cost};
Visit the arc $q = (j, next_{region})$ using VISIT_ARC(q)
NEXT_REGN(q_{i+1};)
Function *VISIT_ARC:*
ADD arc $q = (i,j)$ to Z_q

The main aim of the inter-region path planning technique is to generate a path where the robot $\mathscr{A}$ traverses all the regions at least once without getting stuck in any subtours. If the robot moves from region *'10'* to region *'9'*, it moves closer to sink signifying that it is

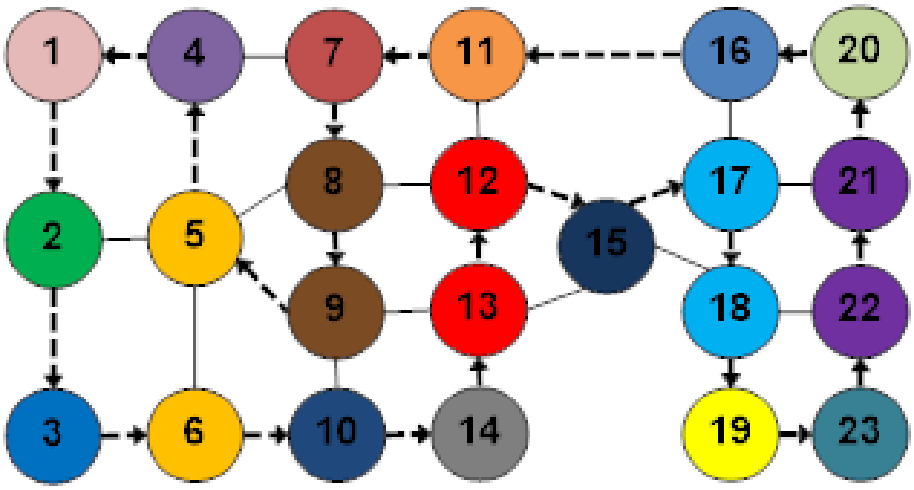


Fig. 4.6 Inter-region path planning generated using the proposed heuristic *'IRPP'* for the site depicted in Fig. 4.5a.

cutting a path short, i.e. not traversing all regions. Therefore, the next region that is farthest from the sink vertex *'3'* is selected, i.e. vertex *'14'*. While being stationed at vertex *'12'*, the robot executes the *'look-ahead'* plan and move to region *'15'*, which allows it to travel the specified edge $(15-17) \in \Theta$.

Using the 'IRPP' heuristic, the generated inter-region path in Fig. 4.6 is illustrated. It is evident from the figure that the generated path differs from the solution proposed by the MILP Strategy, however, it is able to achieve the same, $Cost = 27$.

4.6 Intra-Region Path Planning

After computing the inter-regional path for robot $\mathscr{A}$ in previous Section 4.5, the problem of computation of coverage path for each region $i \in \mathscr{M}$ in this section is tackled. The order of the inter-region path χ plays a pivotal role in determining the intra-region paths. The intra-regional path is computed first by decomposing the regions into grids and then computing Boustrophedon Motion for each region as described in the following subsections.

4.6.1 Grid Decomposition

For computation of any path, the starting and ending locations are the necessary inputs. Similarly, for computing a coverage path for any region i, the algorithm first and foremost determines the entrance and exit for all the regions i. Thus, to find an optimal intra-region

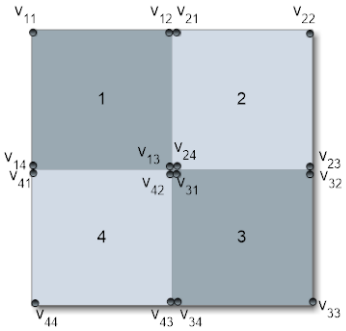


Fig. 4.7 An example site with four regions with their corner vertices.

path for any region $i \in \mathscr{M}$, the preceding and succeeding regions are used to determine the start and end location, respectively.

As known, a rectangular region i is characterized by four vertices: $\{v_{i1}, v_{i2}, v_{i3}, v_{i4}\}$. If $i-1$ is the preceding region for region i, the entrance for the region i is the vertex connecting both the regions i and $i-1$. For instance, there are four regions as shown in Fig. 4.7 characterized by their four vertices. The optimal inter-region path generated is $\{1->2->3->4->1\}$. For region 2, the preceding region is 1 so the entrance point can either be v_{24} or v_{21}. Similarly, if $i+1$ is the succeeding region, the exit for the region i is the vertex connecting the regions i and $i+1$. Now, in Fig. 4.7, the succeeding region is region 3. Therefore, the exit point from region 2 is either v_{24} or region v_{23}.

After determining the entrance and exit, the region is decomposed into a collection of grid cells. Let each region i be decomposed into a grid Ψ_i with a set of uniform grid cells such that

$$\Psi_i = \{\psi_{jk} | 1 \leq j \leq \frac{l_i}{\lceil \frac{l_i}{r} \rceil}, 1 \leq k \leq \frac{b_i}{\lceil \frac{b_i}{r} \rceil}\} \tag{4.28}$$

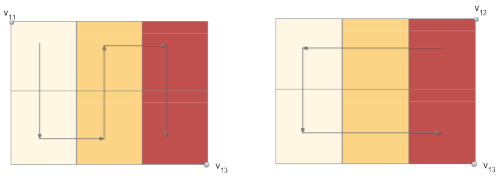
where, $l_i = ||v_{i4} - v_{i3}||$ and $b_i = ||v_{i2} - v_{i3}||$ is the length and breadth of each region i, respectively, $||.||$ is a L_2 norm operator and $r \times r$ is the sensing radius of robot $\mathscr{A}$. The number of grid cells, in region i is $|\Psi_i| = \lceil \frac{l_i}{r} \rceil \lceil \frac{b_i}{r} \rceil$. Since $\frac{l_i}{\lceil \frac{l_i}{r} \rceil} \leq r$ and $\frac{b_i}{\lceil \frac{b_i}{r} \rceil} \leq r$, i.e., size of each grid cell $\psi_{jk} \in \Psi$ is either smaller or equal to the sensing range of the robot. This implies that if a robot visits a grid cell, it is able to successfully map or cover the entire area of grid cell in a single visit. If each cell is considered as a region and the problem of intra-region traversal to tackle translates to how can a robot cover all the cells once, with different start and end locations.

4.6.2 Boustrophedon Motion

After decomposing each region into a grid and determining the start and end point for each region, the coverage path for the intra region traversal is computed. As described in Section 2.7, two important categories of CPP techniques in aerial context are *'Boustrphedon CPP'* and *'Grid based Travelling Salesman CPP'*. *Boustrophedon CPP* technique is an exact cellular decomposition technique which decomposes the site into smaller cells using a sweep line. After each cell is obtained, the orientation of each cell is analysed and back-forth motion (also known as *'Boustrophedon Motion'*) is setup.

In *'Grid based TSP'*, unlike *'Boustrphedon Path Planning'*, the site is decomposed into a regularly spaced grid with a uniform grid cell size. The CPP problem now translates into generating a TSP tour, i.e. traversing each cell once and starting and ending at same cell location. In the present case, the starting and ending location for the intra-region traversal are not same, therefore both the techniques are combined.

After decomposing each region into a grid and determining the start and end vertices as described in last subsection, a Boustrophedon Motion is computed between the start and end location based on the orientation using Algorithm 8. The orientation of start and end vertices is analysed first. If the start and end vertices are located diagonally opposite, the coverage path is a vertical Boustrophedon Motion as shown in Fig. 4.8a. The start and ending vertices v_{11} and v_{13}, respectively, are diagonally opposite, thus Algorithm 8 generates a horizontal back and forth motion. Similarly, if the start and end vertices are located on the same edge, the coverage path is a horizontal Boustrophedon Motion. As illustrated in Fig. 4.8b, the start and end vertices v_{12} and v_{13}, respectively, share a common edge, therefore a horizontal back and forth coverage path is generated. Now, Theorem 1 is used to prove that the grid-decomposition based Boustrophedon intra-regional path planning approach is able to provide full coverage.



(a) (b)

Fig. 4.8 Boustrphedon Motion for (a) diagonally opposite vertices (b) vertices with same edge.

Algorithm 8: BOUSTROPHEDON MOTION

Input:
1. $\Psi_i = \{\psi_{jk}\}$: Grid Ψ_i representing region i
2. $v_{ss'}$: Starting vertex
3. $v_{ee'}$: Ending vertex

Output: $\mathscr{P}$: coverage path

ORIENTATION ANALYSIS **if** $|s' - e'| > 1$ **then**
  /*Starting and Ending Vertices are Diagonally Opposite*/
  $\mathscr{P}$: Generate a Back and Forth Vertical Motion
else
  /*Starting and Ending vertices lie on same edge*/
  $\mathscr{P}$: Generate a Back and Forth Horizontal Motion

Theorem 1 (*Full Coverage of Grid based Decomposition with Inter-Region Path Planning*:)
Consider the intra-regional path planning problem described above. The intra-region path formulated by grid decomposition approach provides full coverage if $\frac{l_i}{\lceil \frac{l_i}{r} \rceil} = \frac{b_i}{\lceil \frac{b_i}{r} \rceil} = r, \forall i \in \mathscr{M}$ and each grid cell is regarded as a region.

Proof: If $\frac{l_i}{\lceil \frac{l_i}{r} \rceil} = \frac{b_i}{\lceil \frac{b_i}{r} \rceil} = r$, the size of each grid cell equals the sensing rage of the robot $\mathscr{A}$, this implies that the shortest paths to plan through the centres of all the cells in any region $i \in \mathscr{M}$ is equivalent to the shortest path to cover the entire region i. Therefore, when applying grid decomposition along with the inter-region path planning approach, the grid decomposition based path planning allows full coverage of the target site.

4.7 Experimental Evaluation

In this section, a set of simulation studies is performed to evaluate the performance of the proposed approach in terms of *'full coverage', 'efficiency' and 'comparison with existing work'*. All the algorithms are implemented using MATLAB and IBM-CPLEX, and the experiments are performed on a DELL XPS IDV8QVO with Intel Core i5, 8GB memory and 225GB storage[1].

4.7.1 Experimental Setup

The input data for the experiments have been generated using the following framework.

[1]The code for all algorithms can be found at: https://github.com/ZebaKhanam91/CoveragePathPlanning

Target site generation: The data corresponding to the regions for a site which needs to be covered has been generated randomly. The number of regions in the sites is given by the user with an upper bound of 50 regions. This assumption is based on the limit of an aerial robot to perform coverage with given energy budget [42]. The regions in the site are assumed to be distributed around an origin marked by the coordinates (0,0). A new region is created by first tentatively selecting a centroid for it. The coordinates of this centroid are obtained by selecting its distance from the origin through a random distribution U1 $[0,200]$ and also determining the angle that it subtends on the x axis, from another uniform random distribution U2 $[0,359]$.

A new region is created using function $rectangle()$ [162]. This function takes as input the coordinates of the centroid (obtained as above), width and height of each region to be generated. $rectangle()$ returns a rectangle based on the region parameters provided as input. The width and height of a region are generated from normal distribution having standard deviation as $\sigma_{side} = 2.5$ and the mean as $\mu_{side} = 7.5$. The new region is actually accepted if it does not overlap with any other region already generated; otherwise, it is discarded. These steps are repeated unless and until desired number of regions are created. In case of a rejection, the steps to create a new region are repeated. The above steps are continued until the required number of regions have been generated. The size of a region is measured in terms of number of grid cells that it contains. The size of each grid cell is assumed to be $1 \times 1\ m^2$.

Specific Order generation: As discussed above, the overall operation has an associated order which provides precedence provision for coverage. The specified order Θ is obtained where the number of specified edges are obtained which the robot must traverse using normal distribution having standard deviation as $\sigma = \frac{m}{8}$ and the mean as $\mu = \frac{m}{4}$. The specified pair of edge in Θ is selected by uniform distribution U3 $[1,m]$. If the edge pair already exists in Θ, this edge pair is discarded.

Robot Characteristics: The robot is assumed to have square footprints whose areas are expressed as an integral number of grid cells. The sensing range of robot r can take different values ranging from $\{1,2,3,4\}$.

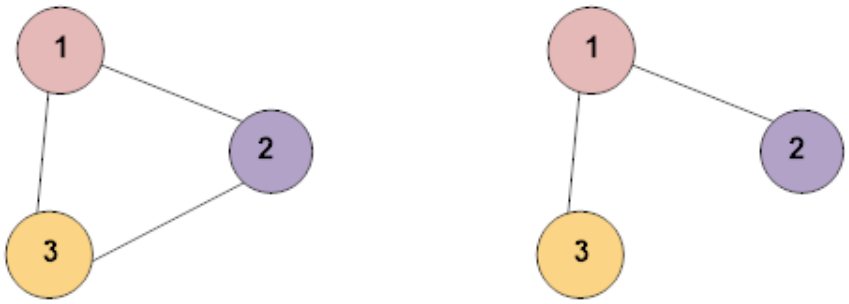
4.7.2 Full Coverage

Theorem 1 in Section 4.6.2 shows that the proposed technique can compute full coverage path for each region, if grid-based decomposition is coupled with inter-region traversal. The inter-region traversal generates a path under the minor assumption that the connectivity graph generated in Section 4.5.1 contains at least one Eulerian trail, i.e. there exits a cycle where each edge is traversed only once. The minor assumption for intra-region traversal is

that $\frac{l_i}{\lceil \frac{l_i}{r} \rceil} = \frac{b_i}{\lceil \frac{b_i}{r} \rceil} = r$. The optimality of the proposed coverage approach is studied using the following experiment designed to elucidate the ability of the proposed approach to find the solution when the above mentioned assumption is either satisfied or violated.

Experiment I

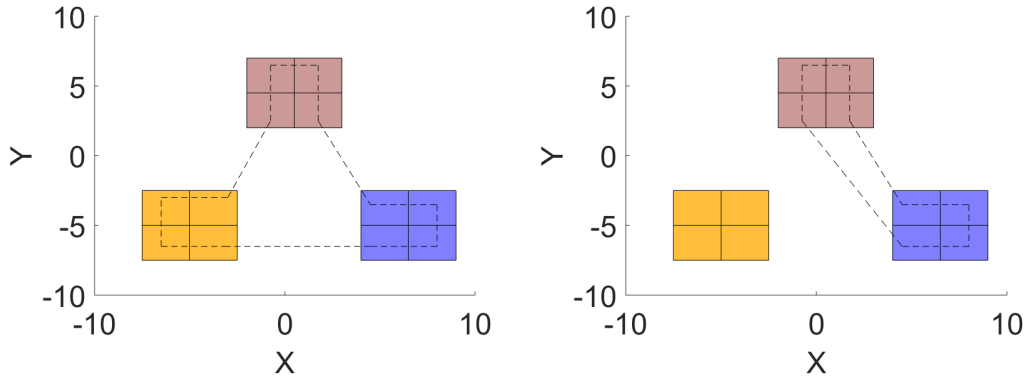
The Experiment I is designed to investigate the assumption of the inter-region traversal, i.e. '*there exists at least one Eulerian trail*'. The first experiment is carried out under a two site setting depicted using the connectivity graphs in Fig. 4.9. In the first and second setting, the path is determined when the assumption is satisfied and relaxed, respectively. The first experiment contains three regions ($\mathscr{M} = 1,2,3$), which have equal length and width, such that $l_i = b_i = 4, \forall i \in \mathscr{M}$. The sensor range of the robot $\mathscr{A}$ is $r = 2$. Thus, $\frac{l_i}{\lceil \frac{l_i}{r} \rceil} = \frac{b_i}{\lceil \frac{b_i}{r} \rceil} = r = 2$, holds for both cases.



(a) (b)

Fig. 4.9 Experiment I site (a) containing eulerian trail (b) without eulerian trail.

The order of coverage for the site is $\Theta = (1,2)$. The distance between each region is unit cost. CPLEX and IRPP heuristic both generate the path $\{1->2->3->1\}$ for the first setting with Eulerian trail with $cost = 3\ units$. The optimal coverage path is depicted in Fig. 4.10a. To illustrate the performance of CPP algorithm when the underlying assumption of inter-region traversal is violated, the region *'2'* and *'3'* are not connected. The remaining settings like coverage order and cost traversal remain same. The path generated by the CPLEX and IRPP heuristics $\{1->2->1\}$, which is a suboptimal solution as region *'3'* is not traversed. The provision of correlation between work (precedence provision) forces the robot to traverse edge $1->2$. Since, the region *'2'* is only connected to region *'1'*, the robot traverses back to depot 1. Now, it can traverse to region 3, however it is unable to minimise coverage trajectory. Thus, it can be concluded that if the underlying assumption of inter-region path planning, connectivity graph $\mathscr{G}$ derived using Algorithm 5 contains Eulerian trail, is violated then the computed coverage path does not cover the entire site.



(a) (b)

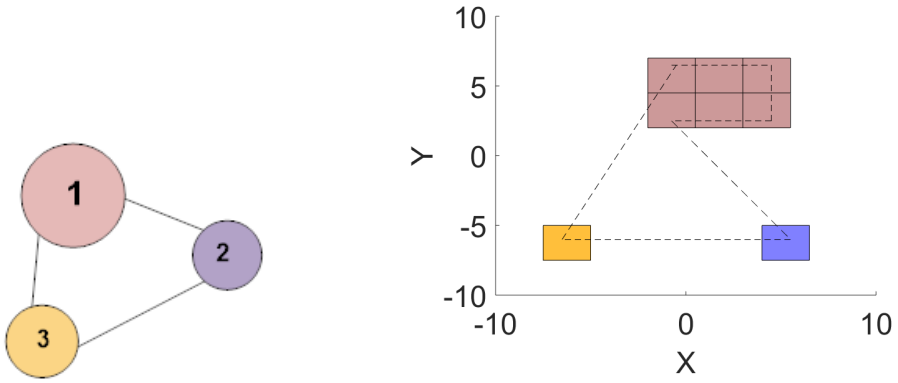
Fig. 4.10 Full coverage path found when site (a) containing Eulerian trail (b) without Eulerian trail.

Experiment II

The Experiment II is designed to investigate the assumption of the intra-region traversal, i.e. $\frac{l_i}{\lceil \frac{l_i}{r} \rceil} = \frac{b_i}{\lceil \frac{b_i}{r} \rceil} = r$. The second experiment is carried out under two site settings depicted using the connectivity graphs in Fig. 4.10 which was considered in Experiment I. However, the sensor range is increased $r = 3$, thus violating the constraint $\frac{l_i}{\lceil \frac{l_i}{r} \rceil} \neq \frac{b_i}{\lceil \frac{b_i}{r} \rceil} \neq r$. Under this setting, the optimal path generated is similar to the path generated in Experiment I illustrated in Fig. 4.9. Thus, it can be concluded that even if the minor assumption is violated, intra-region path traversal generates a full coverage path solution. This can be attributed to the combination of grid decomposition with Boustrophedon Motion.

Experiment III

The experiment III is designed to investigate the performance of area coverage algorithm, when unlike previous experiments, the regions of traversal are not similar. The connectivity graph of the site is shown in Fig. 4.11a. The region *'1'* is the largest region of size 6×4 and the other two regions are of size 2×2. The range of sensor is set to $r = 2$. The inter-region path generated using CPLEX and IRPP is $\{1 -> 2 -> 3 -> 1\}$. In intra-region traversal, the region *'2'* and region *'3'* have a single grid cell and since $\frac{l_i}{\lceil \frac{l_i}{r} \rceil} = \frac{b_i}{\lceil \frac{b_i}{r} \rceil} = r$, the assumption holds true, thus visiting the centre of the grid cell is sufficient to cover the entire grid cell. Similarly, region *'1'* is decomposed into 3×2 grid cells using Algorithm 8 , the robot $\mathscr{A}$ executes a vertical Boustrophedon Motion.



(a) (b)

Fig. 4.11 Full coverage path found when site (a) containing eulerian trail (b) without eulerian trail.

4.7.3 Efficiency Study

In this subsection, the efficiency of the proposed approach is investigated. As the approach consists of two phases: inter region and intra-region path planning. The following experiments are designed to study the properties of each phase with respect to region size and number of regions.

Table 4.2 Execution Time for IRPP and MILP with respect to varying number of regions and specified edges.

Number of Regions	Number of Specified edges ($\|\Theta\|$)	Execution Time (s)	
		MILP	IRPP
3	1	59.76	39.23
	2	57.61	43.13
4	1	61.27	41.23
	2	63.11	45.13
	3	62.9	46.7
5	1	81.6	56.6
	2	80.17	61.2
	3	82.9	64.44
6	2	90.3	55.12
	3	85.21	60.33
	4	87.78	67.74

Experiment IV

In Experiment IV, the properties of two proposed techniques for inter-region traversal: the MILP based technique (proposed in Section 4.5.2) and IRPP heuristic (proposed in Section 4.5.4) are studied under varying number of regions. A set of uniform regions of size $l_i = 2,\ b_i = 4$ are considered. The sensor range is set to $r = 2$. The order of execution $\Theta = (2,1)$. As shown in Fig. 4.12, the execution time of both the MILP based technique

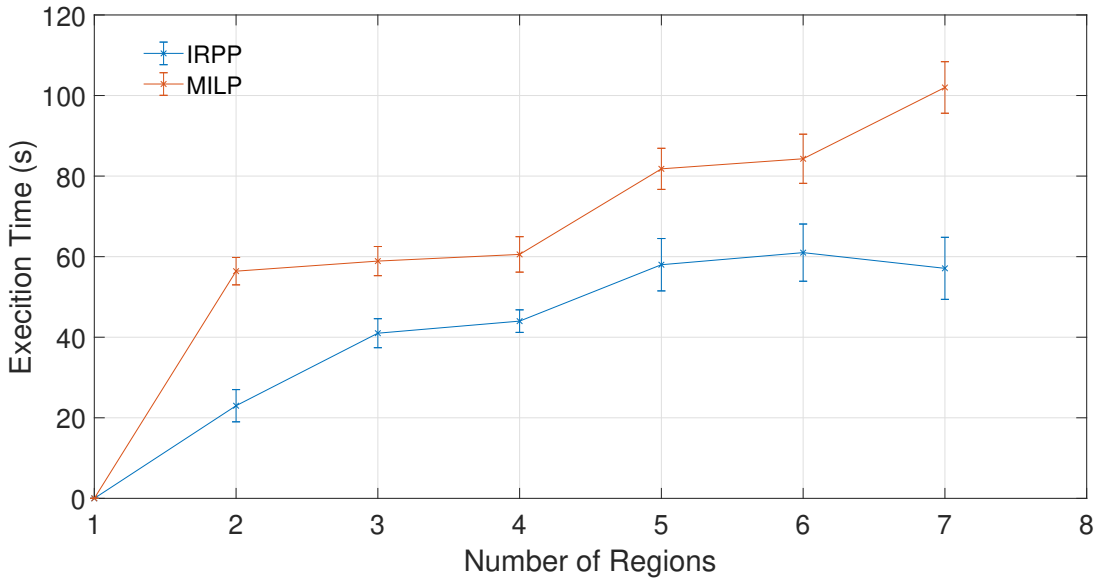


Fig. 4.12 Analysis of execution time for Inter-region traversal with respect to number of regions

and the *'IRPP'* heuristic based approach increases with the increment in number of regions indicating that both the proposed techniques suffer from the curse of dimensionality with respect to number of regions. However, the performance of both the techniques are compared in Fig. 4.13. It can be concluded that the MILP based technique yields a path with less cost as compared to 'IRPP' based heuristic with increase in the number of regions. In order to achieve this, more computation time is required as the number of regions increase (see Fig. 4.13).

This experiment is expanded further to investigate the role precedence provision plays in the inter-region path planning. To investigate this role, the execution time of both the inter-region traversal techniques is recorded for a site with number of regions varying between 3 and 6. Further, for a fixed number of regions, cardinality of specified order ($|\Theta|$) is varied and computational time is tabulated in Table 4.2. It is noted that the number of specified edges

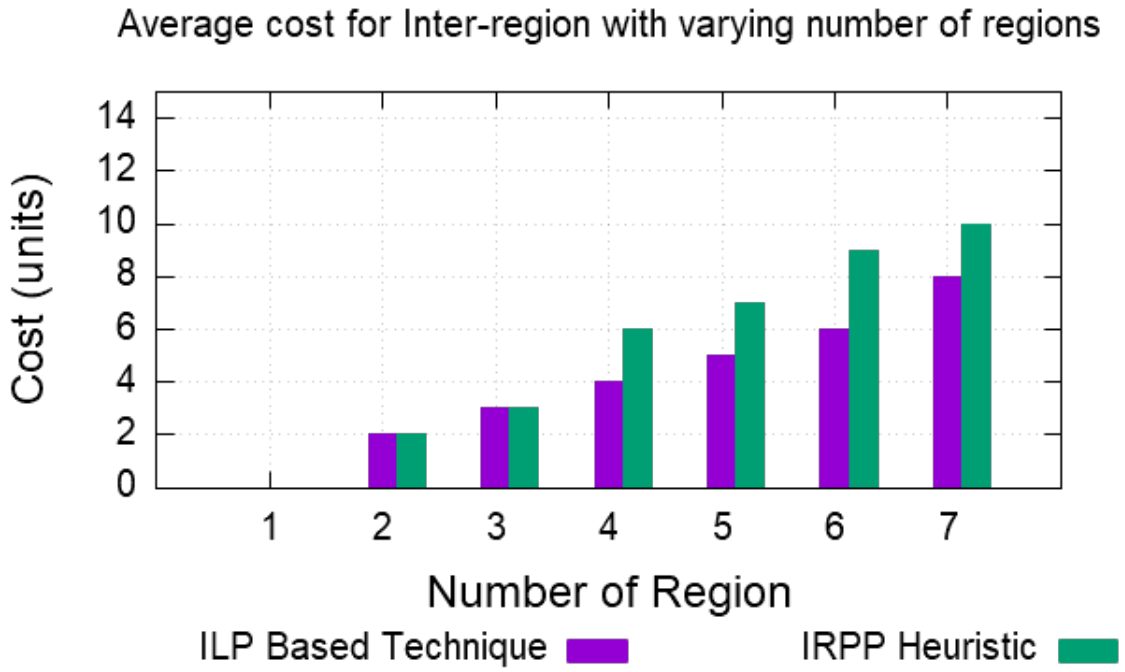


Fig. 4.13 Analysis of cost for Inter-region traversal with respect to number of regions.

don't have any effect on execution time for MILP based algorithm. But, the execution time increases as $|\Theta|$ increases for IRPP. This can be attributed to the fact that MILP considers Θ as hardlined constraints whereas IRPP takes Θ as a provision.

Experiment V

Experiment V is designed to investigate the efficiency of the proposed intra-region traversal technique under varying region sizes. In this experiment, two regions are considered with length of each region set to be $l_i = 2$, $i \in \{1,2\}$. The sensor range is set to $r = 2$. The breadth of each region is then increased simultaneously. The inter-region path generated using both the inter-region traversal techniques is $\{1->2->1\}$ with $cost = 2\ units$.

It is observed from Fig. 4.14 that the execution time varies linearly with the increment in the breadth of region. This trend can be attributed to the fact that as the breadth of each region increases, the number of grid cells increase. Therefore, this leads to overall increase in execution time required to traverse each region indicating scalability to region size.

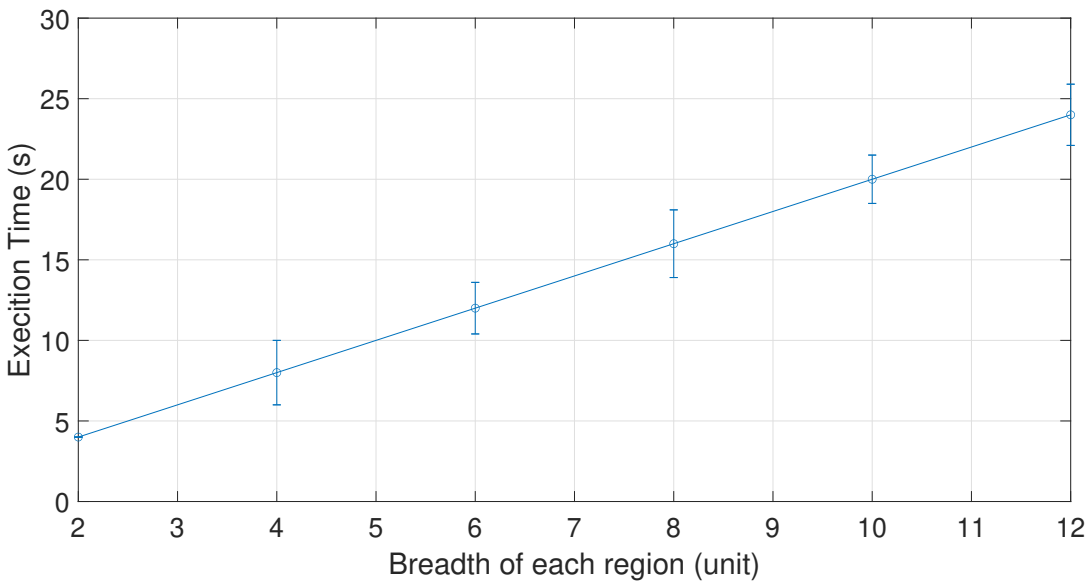


Fig. 4.14 Analysis of execution time for Intra-region traversal with respect to breadth of regions.

Experiment VI

Experiment VI is designed to show the capability of the proposed approach in carrying out coverage of relatively large scale sites. It was suggested in [42] that robots moving at a fixed altitude can traverse up to 17 regions in a single trip. In Fig. 4.14, the execution time for intra-region traversal is directly proportional to number of grid cells. Fig. 4.15 observes a trend that execution time for the inter-region traversal using IRPP heuristics increases with the increment in the number of regions. Any large scale site can have large number of regions. However, in a recent survey [11], it was stated that coverage problem is computed on a part of a map rather than a complete map. The upper limit of number of regions in a partial map for computation of coverage is assumed to be 50 such that $M = \{1, 2, \ldots, 50\}$. The execution time for the MILP technique is calculated to be 889.58s. This illustrates that MILP is capable of generating a feasible solution for even the large scale sites. The IRPP heuristic based technique takes up to 140.7s to generate a feasible solution.

4.7.4 Comparison with Existing Works

In this section the performance of the proposed approach is evaluated with respect to following approaches of CPP of disjoint regions:

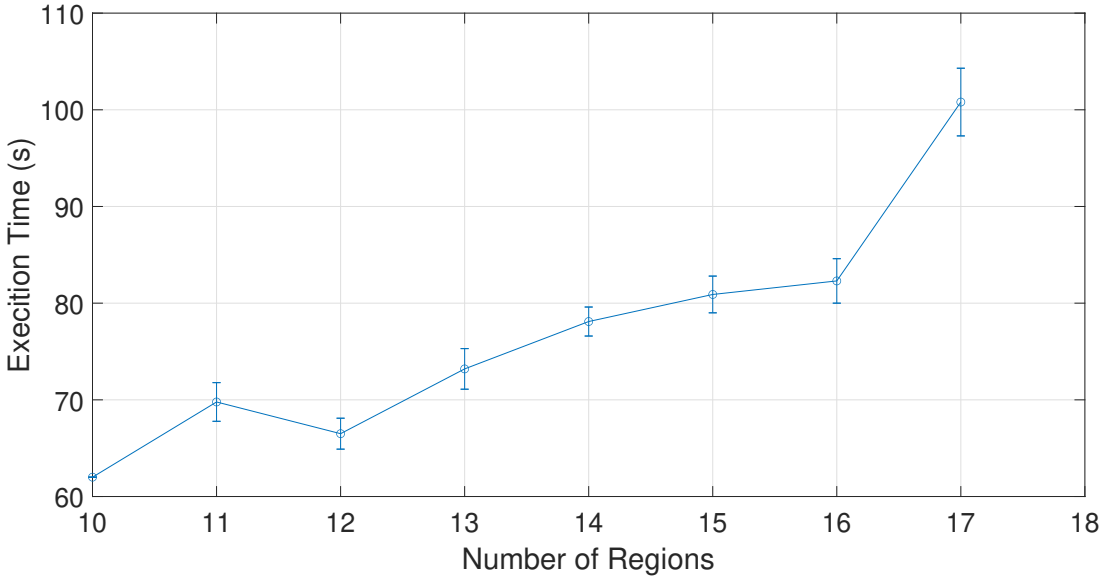


Fig. 4.15 Analysis of execution time for inter-region traversal with respect to number of regions for large scale sites.

1. TSPP [109]: This method first derives inter-region path planning using genetic algorithm [163] and then determines intra-regional path.

2. Dynamic Programming [42]: The coverage problem is solved using integrated approach of dynamic programming.

3. Extended Genetic Algorithm [43]: The coverage problem is used as an integrated problem using genetic algorithm [163].

In this comparison study, genetic algorithm is implemented based on the code in [164] where the parameter setting were tuned to be: 1) Number of iterations: 10^4, 2) Population size: 100, and 3) termination when fitness score does not change for 10 iterations. The dynamic algorithm was implemented based on code in [165].

The performance of the proposed work is compared with the above mentioned works in terms of coverage cost. Fig. 4.16 shows a trend that when the number of regions are small (less than or equal to 3) the performance of all the approaches are similar. However, as the number of regions increase, the coverage cost of existing works increases. The main reason that they incur a large cost is because they fail to follow the order as observed in Fig. 4.16. Keeping up with the trend in respect to coverage cost, it can be concluded that the proposed approach with precedence provision outperforms all the above considered works [42, 43, 109].

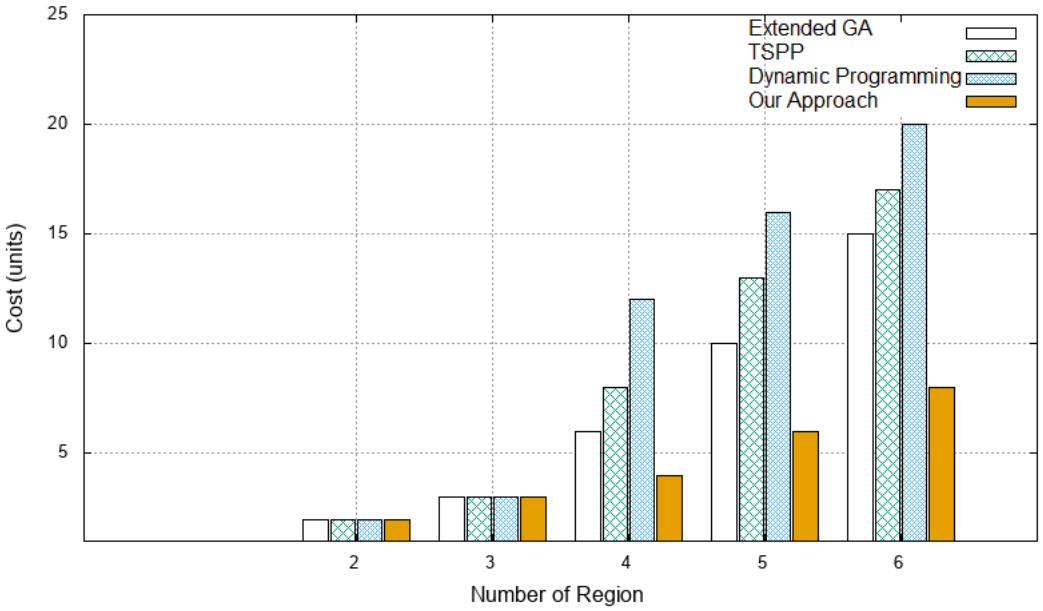


Fig. 4.16 Cost incurred with respect to number of regions.

4.8 Summary

In this chapter, new CPP techniques for coverage of sites with disjoint regions and precedence provision were mathematically formulated. The problem was first decoupled as inter-region path planning and intra-region path planning. MILP and heuristic based technique are proposed to solve the inter-region path planning. For intra-region path planning, each region is decomposed into a grid and Boustrophedon Motion is planned over each region. The combination of both the approaches generates a full coverage path which is proved under minor assumption. The simulation studies are carried out to study the full coverage and efficiency of the proposed approach in different scenarios. A comparison study with the state-of-the-art techniques revealed that the proposed approach out-performed them in terms of coverage cost incurred.

Chapter 5

Coverage Path Planning of Distributed Regions with Energy Constraint

5.1 Introduction

Previously, in Chapter 4, coverage path planning for distributed regions using an aerial robot was discussed. However, another challenge which an aerial robot or Unmanned Aerial Vehicle (UAV) faces, while covering distributed multiple regions is an energy constraint where complete area coverage is not possible. In this chapter, this research gap is addressed by proposing a novel algorithm which solves a variant of area coverage problem where the UAV aims to achieve near-optimal area coverage due to path length limitation caused by the energy constraint. The problem is approached by first formulating the problem and later on presenting a solution. The solution has been partitioned into two inter-dependent subproblems : i) inter-region coverage, ii) intra-region coverage. The performance of the algorithm is evaluated by analysing its properties over an exhaustive set of test case scenarios and comparing it against two state-of-the-art area coverage approaches.

The remainder of the chapter is organised as follows. The overview of the problem is presented in Section 5.2. This is followed by detailed description of proposed algorithm in Section 5.3. The experimental evaluation of the proposed method is performed in Section 5.4. Finally, the conclusions are summarized in Section 5.5.

5.2 Problem Overview

The existing research in CPP has been focused on use of Unmanned Aerial Vehicles (UAVs) in many application domains for surveying and covering large areas. One of the key challenges

for UAV based area coverage is limited on-board energy [166, 167]. This problem intensifies when UAV is entrusted with coverage of multiple distributed regions.

Energy-efficient coverage path planning using an aerial robot for a single region or multiple connected regions is well researched [168–170]. However, the research works which explore the problem of computation of coverage path for multiple disjoint regions assume that an aerial robot has sufficient on-board energy [42, 43, 109]. There exist only one research work that considers energy limitation and constraint [44]. This work allowed the UAV to return to the depot to change its battery. However, this might not be a realistic scenario in many cases such as extreme environmental inspection and limited energy may lead to partial area coverage. Nevertheless, appropriate execution of partial coverage can provide useful information of the site. Recently, a few works have explored optimization of coverage path for partial coverage of a single region [110–112]. To the best of the author's knowledge, there does not exist a single work addressing the problem of partial area coverage over geographically distributed regions by energy constrained aerial robot.

The research presented in this chapter considers solving a variation of CPP problem where an aerial robot cannot achieve full coverage of multiple distributed regions due to the energy constraints on the UAV which impose limitation on the total path length. The algorithm first computes the inter-region path which also decides the start and exit point for each region. Using these points, the algorithm then distributes waypoints to achieve near-optimal coverage taking into consideration energy constraints. The optimal path is defined as a set of waypoints such that after traversing the path, entire area is covered by robot's sensor. In this context, the near-optimal path is defined as a set of waypoints, if visited by the aerial robot, robot's sensor is able to maximize the coverage area within the given limitation [110]. Fig. 5.1 shows an example of a near optimal coverage path for a single region. The robot maximizes the area coverage of gas power plant, coverage path is adrift and unconventional due to environmental obstructions like uneven nitrogen gas distribution.

5.3 Proposed Algorithm

The algorithm for near optimal coverage of disjoint regions is described next which is applicable to an energy constrained UAV. Considering the kinematics, the following assumptions are made. The path is defined as a sequence of waypoints which the UAV traverses. The velocity and acceleration when moving between waypoints is constant such that the UAV requires same amount of energy to complete the travel between two pairs of waypoints which are at same distance.

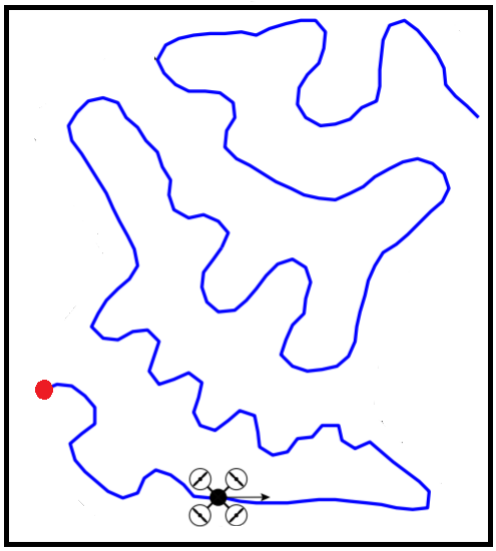


Fig. 5.1 A near-optimal coverage path considering a path-length constraint. Start point is denoted with a red circle [110]

5.3.1 Algorithm Overview

In this section, a higher level overview of the algorithm is illustrated. Fig. 5.2 illustrates a flow chart of the path generation process.

1. Calculate the total path length and the number of waypoints for each region.
2. Calculate the inter-region path and first and last waypoint for each region.
3. Generate a random sequence of path waypoints for each region with desired spacing ensuring that the path does not intersect each other.
4. Optimize the intra-region path to maximize the coverage area subjected to energy constraints.

5.3.2 Calculating Total Path Length

The proposed algorithm requires the number of waypoints in each region to be pre-computed. For each region, the waypoints are all spaced at an equal distance d_i away from the adjacent waypoint. If there are n regions which the UAV has to traverse and for each region r_i, there

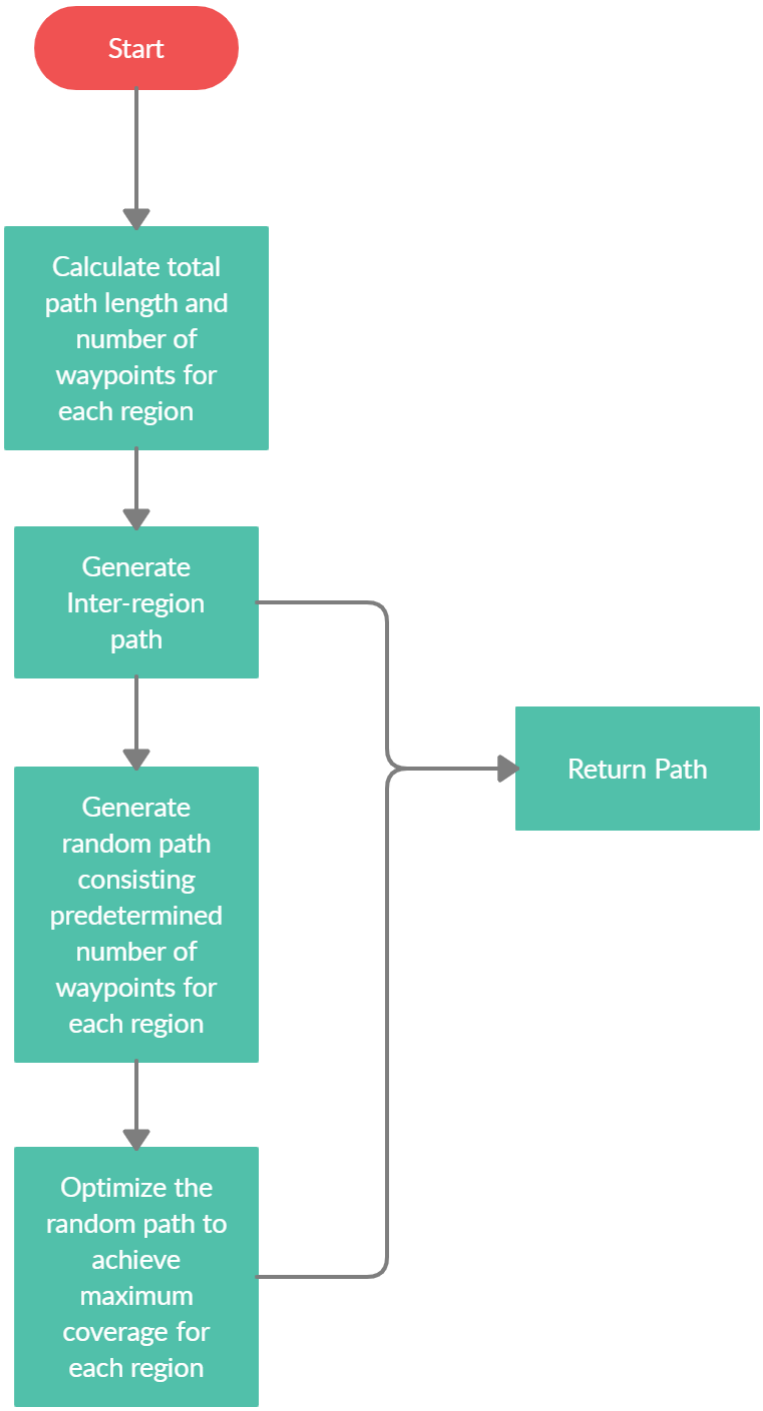


Fig. 5.2 Flowchart elucidating the proposed algorithm for partial area coverage.

are m_i ($W_i = \{w_1^i, \ldots, w_{m_i}^i\}$) total waypoints, the path length can be computed as:

$$l_{total} = \sum_{i=1}^{n} (m_i - 1)d_i + \sum_{i=1}^{n-1} Dist(w_{m_i}^i, w_1^{i+1}) \tag{5.1}$$

where, $Dist(a,b)$ is a function which calculates euclidean distance between two input waypoints a and b. The maximum path length is dependent on the energy consumption of the UAV used for the mission. To compute the energy required to traverse the path length, two functions $F_m(d,v)$ and $F_r(t)$ are required. $F_m(d,v)$ outputs the energy required to travel distance d with a velocity v and $F_r(t)$ outputs the energy consumed when resting at a waypoint for sensing for an amount of time t. Two assumptions are that the UAV travels at a consistent velocity v_t and the waypoints in a particular region i are spaced equally with distance d. Thus, the energy $E_m = F_m(d,v_t)$ is constant for each region $r_i \in R$. The aerial robot is assumed to stop at each waypoint for sensing for a constant amount of time t_r, so $E_r = F_r(t_r)$. Another assumption which is made is that, for each region, the first waypoint w_1^i and last waypoint $w_{m_i}^i$ are the vertices of the region i. The energy $E_m^{i,j} = F_m(Dist(w_{m_i}^i, w_1^j), v_t)$ is consumed while traveling from region r_i to region r_j. If there are m_i total waypoints in each region r_i, the energy E_{path} consumed by the UAV while covering all the regions is:

$$E_{path} = \sum_{i=1}^{n} \{m_i E_r + (m_i - 1)E_m\} + \sum_{i=1}^{n-1} E_m^{i,i+1} \tag{5.2}$$

As the total energy capacity E_{total} for a particular UAV is already known, the number of waypoints can be computed by imposing the following condition:

$$E_{path} \leq E_{total} \tag{5.3}$$

In Equation 5.2, the first and second terms indicate the intra-region energy consumption, E_{iar}, described in Section 5.3.4 and the last term indicates the inter-region energy consumption, E_{ier}, described in Section 5.3.3. Solving for m_i, resulting in:

$$\sum_{i=1}^{n} m_i(E_r + E_m) - E_m \leq E_{total} - E_{ier} \tag{5.4}$$

Using Equation 5.4, the waypoints m_i for a region r_i are determined by:

$$\sum_{i=1}^{n} m_i = \frac{E_{iar} + n \times E_m}{E_r + E_m} \tag{5.5}$$

The number of waypoints m_i for each region is assumed to be:

$$m_i = \lfloor \frac{A_i}{A_{max}} \rfloor \times \hat{D} \tag{5.6}$$

Using Equation 5.5 and 5.6, $\hat{D}$ can be formulated as:

$$\hat{D} = \frac{E_{iar} + n \times E_m}{E_r + E_m} \times \frac{A_{max}}{A_{total}} \tag{5.7}$$

where, A_i, A_{max}, A_{total} denote area of region r_i, area of region with maximum area and total area of all the regions, respectively.

In the following subsections, the computation of inter-region energy E_{ier} and intra-region E_{iar} consumption are described.

5.3.3 Inter-Region Energy Consumption (E_{ier})

The first step in the proposed algorithm is to compute energy consumed by the aerial robot while traveling between different regions. As mentioned in last subsection, it is assumed that the first and the last waypoints w_1^i and $w_{m_i}^i$, respectively are the vertices of the regions. To determine the energy consumed inter-region traversal E_{ier}, it is needed to compute:

1. First and last waypoints (w_1^i and $w_{m_i}^i$) $\forall i \in n$.
2. Inter-region traversal order.

However, to minimize the energy consumption, the computation of waypoints should be dependent on inter-region traversal orders. The full algorithm is described in Algorithm 9.

Putting together all the elements mathematically stated in Algorithm 9, the inter-region energy E_{ier} computation algorithm works as follows. To begin, a list $\hat{V}$ listing number of regions visited by aerial robot is initialized $NULL$. In addition, the start depot region r_q is initialized as the current region C_r of traversal by the aerial robot and inter-region energy E_{ier} is initialized as zero. The algorithm then loops through several actions. First, it determines the nearest region to the current region which has not been visited by the aerial robot. The nearest region (r_i) is determined by the distance between the vertices of different regions (D^{C_r,r_i}). Next, the vertex of the current region is designated as the last waypoint of current region ($w_{m_i}^{C_r}$) and the vertex of the nearest region is designated as the start waypoint of next region ($w_1^{r_i}$). Finally, the current region C_r is added to the visited list $\hat{V}$ and the nearest region r_i is selected as the current region. Using the distance between two waypoints, energy consumption is computed and added to the total inter-region energy E_{ier}. This process iterates until all the regions are visited.

Algorithm 9: INTER-REGION ENERGY CALCULATION

Input:
1. $R = \{r_1, \ldots, r_n\}$: Set of regions;
2. $< Centre, Vertices, Side > CVS = \{< c_1, \{v_1^1, \ldots, v_m^1\}, S_1 >, \ldots, < c_n, \{v_1^n, \ldots, v_m^n\}, S_n >\}$
3. r_q : Starting depot region;
4. S_{r_q} : Starting waypoint of depot region r_q;
5. v_t : Constant velocity;

Output: Energy Consumption E_{ier}

begin
  INITIALIZATION
    1) $\hat{V} = NULL$: List of regions visited;
    2) $C_r = r_q$: Current region;
    3) $w_1^{r_i}$: Starting waypoint of region r_i;
    4) $w_{m_i}^{r_i}$: Ending waypoint of region r_i;
    5) $D = \{D^{1,2}, \ldots, D^{1,n}, D^{2,3}, \ldots, D^{2,n}, \ldots, D^{n-1,n}\}$: Set of distance matrices where $D_{k,l}^{i,j}$ represents distance between vertex k of region r_i and vertex l of region r_j;
    6) $E_{ier} = 0$; ▷ Initializing Inter-Region Energy
  while $|\hat{V}| \neq |R| - 1$ **do**
    for *each region* $r_i \in R \ni (r_i \notin \hat{V}$ *and* $r_i \neq C_r)$ **do**
      min_{dist} = minimum of D^{C_r, r_i};
      row, col = indices of minimum of D^{C_r, r_i};
      $w_{m_i}^{C_r} = row$;
      $w_1^{r_i} = col$;
      $\hat{V} = \hat{V} \cup C_r$;
      $C_r = r_i$;
      $E_{ier} = E_{ier} + F_m(min_{dist}, v_t)$;

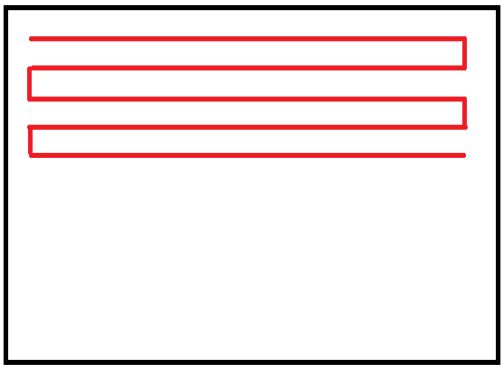
5.3.4 Intra-Region Energy Consumption (E_{iar})

After calculating the number of waypoints (m_i), desired spacing (d_i) and inter-region order of traversal with entrance and exit points for each region, intra-region energy consumption E_{iar} can be calculated. The first step is to calculate the intra-region path, randomly. The random intra-region path needs to be optimized to reduce the overall cost which is described in detail in the next section.

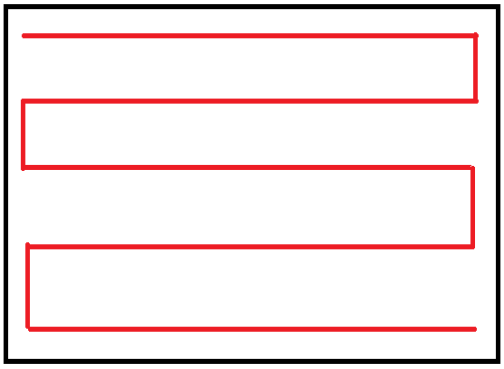
Optimality

Due to the energy-constraint, the aerial robot can only achieve partial coverage. Thus, the proposed algorithm aims to generate a sequence of waypoints to cover the regions in best possible manner, attempting to maximize the coverage area. When generating a random path, it is ensured that the coverage path between waypoints do not overlap. However, there is a

possibility that there can exist two coverage paths which don't overlap and achieve same amount of area coverage but provide different qualities of information. For instance, in Fig. 5.3, the aerial robot is able to cover same amount of area in both the cases. However, in the first case in Fig. 5.3a, the robot is able to provide detailed coverage information about the upper portion of region without spending time acquiring information about the bottom region. In the second case as elucidated in Fig. 5.3b, the area coverage by the aerial robot provides information which presents an overall picture of the region. However, it is assumed that having moderate quality information of the entire region is better than having detailed information of a part of the region and no information about the remaining region. To ensure that the aerial robot takes a coverage path similar to Fig. 5.3b, following cost function $J(.)$ is imposed on coverage path P_i.



(a)



(b)

Fig. 5.3 Example of coverage path with same path length where robot is able to collect (a) detailed information and (b) moderate information

$$J(P_i) = \iint_{r_i} \min_{w_j^i} ||x - w_j^i|| dA \tag{5.8}$$

For region r_i, $W_i = \{w_1^i, \ldots, w_{m_i}^i\}$ is a set of m_i waypoints, $\mathbf{x}$ is a point lying inside the region r_i represented by a vector $[xy]^T \in r_i$ and dA is the differential area $dydx$. It is assumed that the region is a set of grid cells where the resolution of each grid cell is of the size of the footprint of the robot's sensor. When the robot is stationed at the center of the grid cell it can cover the entire grid cell using its sensor. In connection with the previous assumption, another assumption is made that the possible set of waypoints which robot can traverse while covering any region is the centre of the grid cell. If g_{n_i} is the total number of grid cells in the region r_i, this assumption allows the reformulation of the cost function in Equation 5.8 as:

$$J(P_i) = \sum_{x=1}^{g_{n_i}} \min_{w_j^i} ||x - w_j^i|| \tag{5.9}$$

Applying the above formulated cost function $J(P_i)$, the problem of finding location of waypoints for inter-region coverage path P_i^* for region r_i translates to:

$$P_i^* = \arg\min_{P_i} J(P_i) \tag{5.10}$$

subject to constraints,

$$||w_j^i - w_{j-1}^i|| = d,\ \forall i = 1, \ldots, n\ ,\ \forall j = 2, \ldots, m_i \tag{5.11}$$

$$||w_{j+1}^i - w_j^i|| = d,\ \forall i = 1, \ldots, n,\ \forall j = 1, \ldots, m_i - 1 \tag{5.12}$$

$$w_1^i, w_{m_i}^i \in \text{vertex of region } r_i \tag{5.13}$$

5.4 Simulations and Results

In this section, simulations are presented to assess the performance of the proposed algorithm, beginning with several examples of inter-region and intra-region path. A series of experiments are performed to characterize the algorithm with respect to various parameters and compare the algorithm to other path generation algorithms. During the simulation, an arbitrary energy model is used to determine the total path length, which is further utilised to determine number of waypoints for each region. The number of waypoints for each region are input parameters for intra-region path generation. All the algorithms are implemented using MATLAB R2020a,

and the experiments are performed on a DELL XPS IDV8QVO with Intel Core i5, 8GB memory and 225GB storage. The built-in MATLAB function *fmincon* is used for direct optimization.

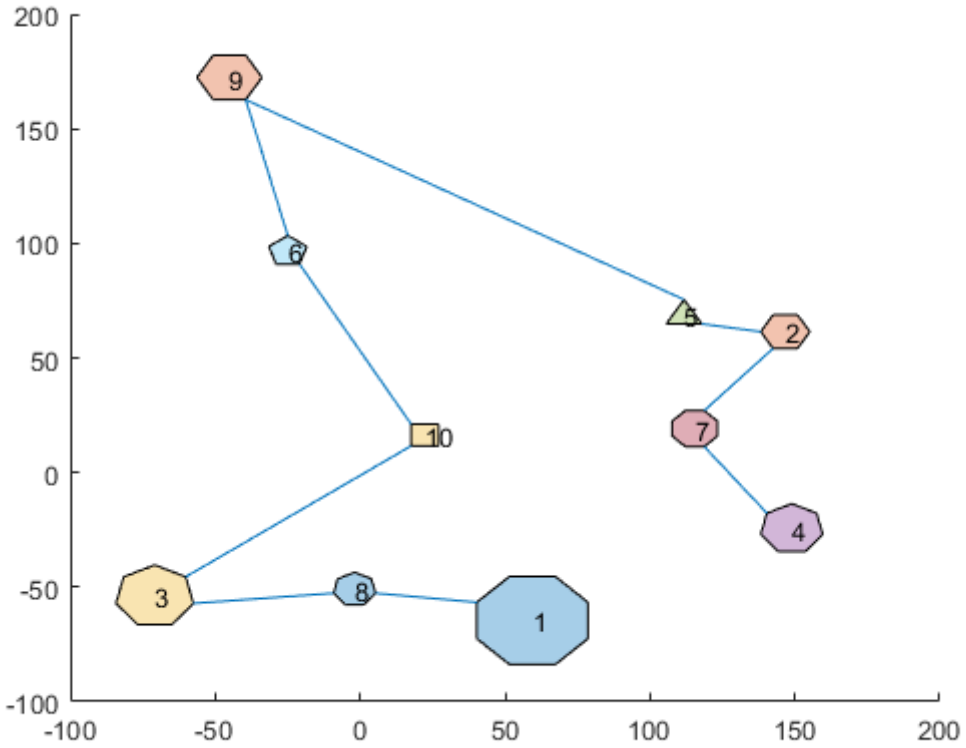
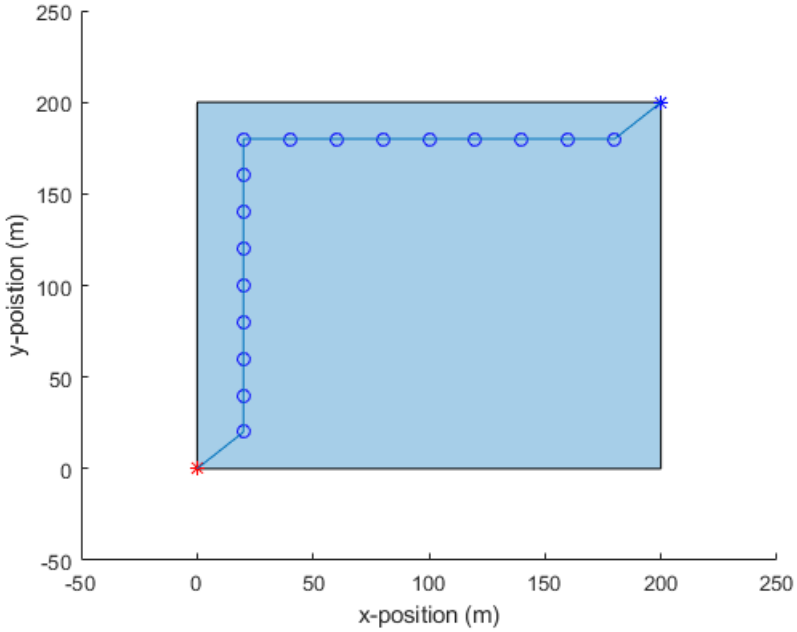


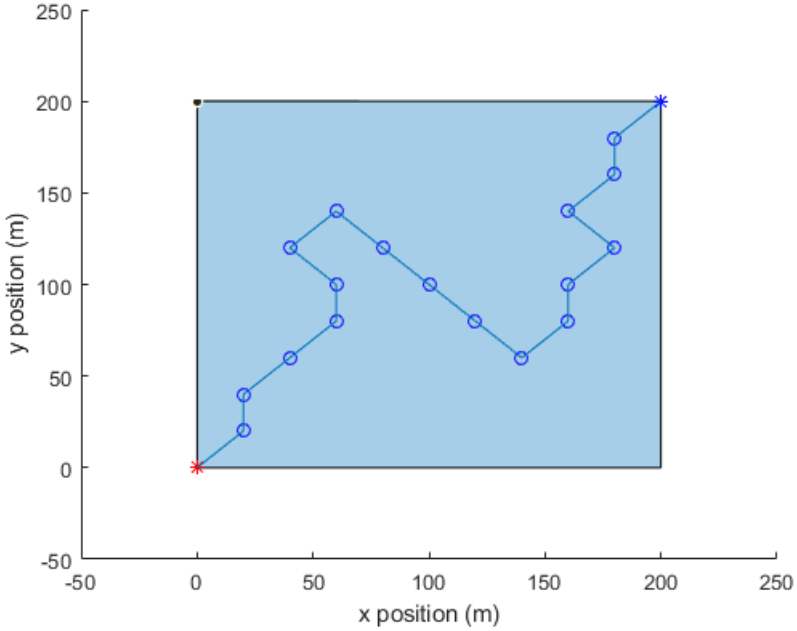
Fig. 5.4 Example of inter-region path generated to cover 10 regions. Depot is denoted by region 1

5.4.1 Target Site Generation

The parameters corresponding to the target site which needs to be covered are generated randomly. The number of regions in the site is given by the user with an upper bound of 100 regions. The above-mentioned assumption is based on the energy budget an aerial robot needs to perform area coverage [42]. The regions in any given site are distributed around an origin denoted by the coordinates (0,0). Any new region is created by generating the coordinates of its centroid. These coordinates are generated by first selecting the distance from the origin through a uniform random distribution U1 $[0, 200]$ and then by determining the angle that it subtends on the x axis, using another uniform random distribution U2 $[0, 360]$.



(a)



(b)

Fig. 5.5 Example of coverage path generated by proposed algorithm for intra-region coverage of a 200 m x 200 m square region with starting waypoint (vertex of region) marked as red asterisk and ending waypoint (vertex of region) marked as blue asterisk (a) initial random path for (b) optimized final path.

The new region is created using built-in MATLAB function $nsidedpoly()$ [162]. This function takes as input the number of vertices, the coordinates of the centroid, and length of each side of the region. $nsidedpoly()$ generates a regular polygon based on the input parameters. The number of vertices is generated through a uniform random distribution U3 $[3, 10]$. The length of the side of a region is generated using normal distribution with standard deviation as $\sigma_{side} = 2.5$ and the mean as $\mu_{side} = 7.5m$. The new region is accepted if and only if it does not overlap with already created region; otherwise, the region is discarded. These steps are repeated until the desired number of regions are generated. The size of each grid cell is assumed to be $1 \times 1\ m^2$.

5.4.2 Path Generation Examples

In the first example, the inter-region algorithm is run in a target site with 10 regions. The region 1 is selected as the starting region (depot). The generated inter-region path is shown in Fig. 5.4. In the second example, an intra-region path is illustrated for a square region with dimension 200 m x 200 m, which is discretized into a 200×200 cell grid. The initial random path is depicted in Fig. 5.5a. The algorithm optimizes to the path shown in Fig. 5.5b.

5.4.3 Algorithm Characterization

In order to characterize the proposed algorithm, several experiments were run in simulation. First experiment is conducted to determine the runtime of the algorithm with respect to the number of regions in the target site. The second set of experiments is conducted to compare the algorithm optimality and runtime of the proposed algorithm against two other algorithms focusing on area coverage of distributed regions. The first state-of-the-art algorithm is the method described by Xie et al [44]. This algorithm is chosen because it is the only work in literature which determines the path for area coverage of multiple distributed regions by the energy constrained UAV. However, due to the fact that the Xie's method performs multiple tours to provide complete area coverage, the proposed work computes coverage path which performs partial coverage in a single tour. Therefore, a slight modification is made to Xie method where the area covered in a single tour is only considered to do sensible comparison. The second state-of-the-art algorithm is a path generation method proposed in Chapter 4 which covers the area by Boustrophedon path for intra-region and uses nearest neighbour algorithm for inter-region traversal. This algorithm is chosen because of its ability to generate computationally inexpensive area coverage solutions.

Average Runtime

To determine the time complexity of the proposed algorithm, an experimental setup is designed by fixing the waypoint spacing d to be 1 grid cell. The number of regions are varied across the range of 25 to 100. For each different number of regions, 100 trials of experiment are performed and average runtime per iteration is computed. The results are illustrated in Fig. 5.6, depicting that as the number of regions increase, the average run time increases. This increment can be attributed to the fact that as the number of regions would increase, the algorithm would require more time to compute inter-region path and optimise the intra-region path to achieve near-optimal area coverage.

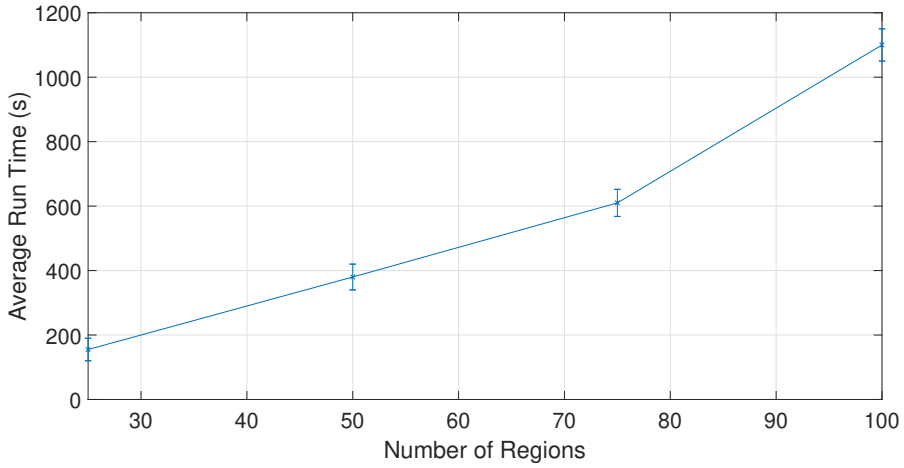


Fig. 5.6 Average runtime for different number of regions

Normalized Area Coverage Comparison

The optimality of the proposed algorithm is assessed by measuring normalized area coverage. Further, this parameter is used to compare the proposed algorithm with the two previously described state-of-the-art algorithms. Five different target sites are generated for the experiments by varying the number of regions from the range of $\{10, 20, 30, 40, 50\}$. All the algorithms are executed 10 times for each number of regions.

The normalized area coverage is computed as the ratio of the partial area covered by the UAV and total area of the target site. The results are shown in Fig. 5.7 . The proposed algorithm generated path is able to achieve normalized area coverage between 55% to 73%, on the other hand, the method [44] achieves partial area coverage between 34% to 47%. The method proposed in Chapter 4 is able to cover 26% to 39% area.

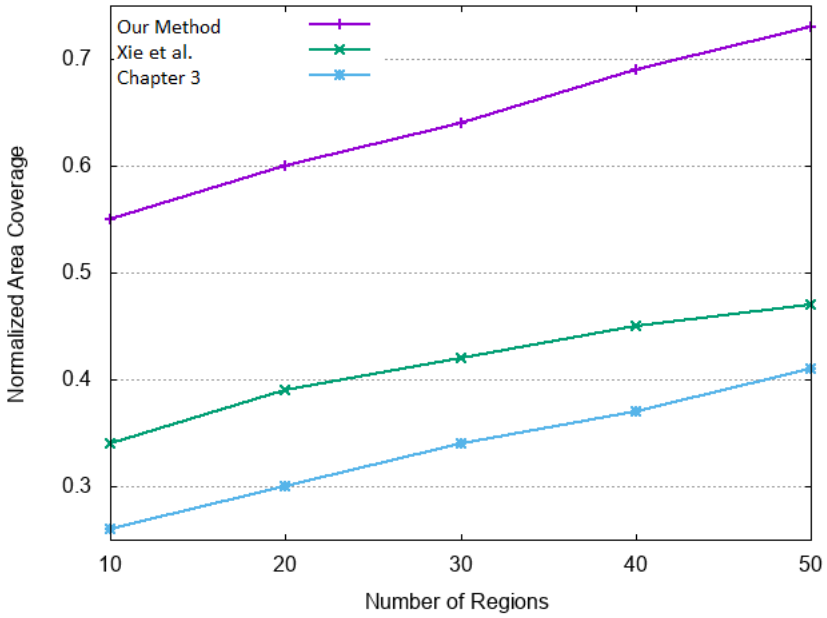


Fig. 5.7 Comparison of Normalized Area Coverage for different number of regions

5.5 Summary

This chapter has presented a partial coverage method for planning near-optimal area coverage paths for multiple distributed regions using an aerial robot given energy limitations. The coverage path generation algorithm first computes an inter-region order traversal and then distributes path waypoints for intra-region coverage. A handful of works have been done previously exploring generation of area coverage path for multiple disconnected regions due to energy constraints, and the path-planning algorithm proposed in this chapter is a novel approach to solving that problem by generating path which aims to achieve near-optimal partial area coverage with given energy budget.

Simulations are conducted to compare the proposed method with other methods that can generate coverage paths for distributed multiple regions. The algorithm presented in this chapter provides paths much closer coverage to the full coverage than the state-of-the-art area coverage methods. The reason behind better performance in terms of area coverage is due to the algorithm design which focuses on optimizing intra-region area coverage by traversing a set of waypoints.

The proposed algorithm has two limitations. The first limitation is that the intra-region path is devised by optimizing the cost function using a built-in MATLAB function. This optimal solution strategy will be prohibitively expensive in terms of solution generation times and required storage space. Further, it will also be difficult to deliver satisfactory outputs as

the number of regions will increase. Hence, probabilistic mission planning can be used as a decision making module [171]. The path planning for any UAV is generally modelled as partially observable Markov decision process (POMDP) [172]. In the case of energy constrained coverage path planning, the problem is mapped as constrained POMDP(CPOMDP) [173]. Recent linear programming solution for approximation of CPOMDP can be considered to generate optimal solution within reasonable time. The second limitation is that this algorithm requires an accurate energy model, assuming that the UAV is able to rotate in place.

Chapter 6

Coverage Path Planning of Distributed Regions using a Heterogeneous Fleet

6.1 Introduction

The previous chapters have indicated that generation of solutions for automated coverage path planning over disjoint regions within constrained time bounds is an important problem especially in domains requiring emergency coverage operations over possibly extreme terrains. In this chapter, the proposed work focusses towards the coverage of disjoint regions in real time, using a fleet of Unmanned Aerial Vehicles (UAVs) having heterogeneous capabilities in terms of sensor footprint sizes, speed and power rating. In this chapter, a lower overhead simulated annealing based statistical solution approach is proposed that is able to deliver good solutions within acceptable time limits. To manage the complexity of the overall problem, while producing fast but satisfactory solutions, the problem has been partitioned into two inter-dependent subproblems : i) Inter-region coverage, ii) Intra-region coverage. The performance of the algorithm has been evaluated by analysing its properties over an exhaustive set of test case scenarios and comparing it against a state-of-the-art genetic algorithm approach.

The remainder of the chapter is organised as follows. Section 6.2 describes the problem overview. This is followed by stating the problem statement in Section 6.3. Section 6.4 proposes a simulated annealing based heuristic approach. The experimental evaluation of the proposed method is shown in Section 6.5. Finally, the conclusions are presented in Section 6.6.

6.2 Problem Overview

Coverage Path Planning (CPP) problem over disjoint regions is relevant in practical scenario including environmental mapping, and extreme environmental inspection [174]. The area coverage operation in such scenario are carried out in critical emergency situations with stringent constraints on the available time within which the coverage must be accomplished. *This work deals with the problem of real time CPP over a set of disjoint regions using a fleet of UAVs which are heterogeneous UAVs in terms of their sensor footprint sizes, power rating and manoeuvring speeds.*

There exist only few works that have explored the problem of CPP for multiple disjoint regions. In these works, both terrestrial robots [175] and UAVs [72] have alternatively been used as coverage robots. However in this chapter, only mutually isolated regions separated by extreme terrain/environment are considered, where use of terrestrial robots are not possible and UAVs must be deployed to serve the purpose. Exploring the literature in Chapter 2, it was observed that the works dealing with the problem of CPP using UAVs may be broadly classified into two major categories based on: a) whether the solution addresses the problem of covering a single region, multiple connected regions, or a set of disjoint isolated regions, b) whether single or multiple UAVs have been used in the solution approach.

However, only a handful of works explore the CPP problem dealing with multiple disjoint regions. In addition to the need to systematically cover the internal area of multiple regions, instead of a single one, the problem with disjoint regions introduces the additional challenge of generating least cost tours covering a set of regions. Each such tour is intended for a particular robot so that it may start from its depot, conduct the tour through the regions under its purview and come back to its depot while incurring minimal overhead. While the intra-region traversal requires a solution to CPP, inter-region traversals are typically mapped to a variant of the Travelling Salesman Problem (TSP). It may be noted that both CPP and TSP are NP-hard in the strong sense and hence optimal solutions become extremely expensive in terms of both time and space even with moderately sized problems. Thus, researches have primarily focused towards design of heuristic solutions to this problem.

In a recent work, Xie et al. [42] proposed a dynamic programming based framework to solve the integrated TSP-CPP problem for rectangular distributed regions to be covered using a single UAV. A further extension of this work considered polygonal regions with arbitrarily many vertices and solved using a genetic algorithm approach [43]. To best of the author's knowledge, there exists only a single work addressing the problem of area coverage over disjoint geographically distributed regions by multiple UAVs [145]. An optimal solution was deduced using a Mixed Integer Linear Programming (MILP) framework. The problem being very complex, use of MILP formulation makes the solution approach highly compute

intensive, and restricts its scalability to problems containing only small number of regions. However, as discussed before, important applications of disjoint region coverage often include emergency coverage mission over vast geographical areas where both the number of regions and their average sizes may be pretty large. This demands the design of efficient and scalable heuristic strategies. In addition, the solution strategy proposed in [145] does not consider real time constraints and restricts the set of UAVs to be homogeneous in nature. In practice though, such mission critical operations may be marked by strict time bounds with the utility provider deploying a set of possibly heterogeneous UAVs at its disposal in an attempt to successfully accomplish the mission within the stipulated deadline.

In this chapter, the problem of CPP computation for area coverage of disjoint regions using a heterogeneous fleet of UAVs is considered. The deployed fleet of UAVs have different footprint sizes, speed, power rating and predefined time frames within which to complete a mission. Given the structure that the problem consists of, it is difficult to design efficient heuristic solution which performs well under all scenarios which may occur in practice. Hence, it was resorted to a statistical solution strategy based on simulated annealing. With its ability to progressively move towards local optima, while intermittently perturbing the search in an attempt to obtain better solutions, simulated annealing has allowed us to design a flexible scheme which can deliver good solutions if given reasonable time. Our experimental evaluations show that the proposed scheme is able to outperform a state-of-the-art genetic algorithm approach in all scenarios.

6.3 Problem Statement

The research presented in this chapter aims to solve the real time area coverage problem corresponding to a target site consisting of multiple disjoint regions using a fleet of heterogeneous UAVs. Each aerial robot or UAV in the fleet is referred to as an agent in this chapter. Let us consider a scenario where a set of agents $\mathscr{A} = \{A_1, \ldots, A_n\}$ need to inspect $\mathscr{M} = \{M_1, \ldots, M_m\}$ spatially distributed and non-overlapping convex polygonal regions. Each agent in $\mathscr{A}$ contains a sensor which inspects the surface beneath with a square footprint whose length can take any alternative value from the set $Range = \{r_1 \times r_1, r_2 \times r_2, \ldots, r_k \times r_k\}$, where $r_i \in Range$ is a positive non-zero integer and k is a constant. Each agent, A_i, is also categorized by a possibly distinct power rating P_i and manoeuvring speed S_i. The power rating P_i and speed S_i can take an integer value from a uniform range. All the agents in the fleet operate at a fixed altitude. The geographical area of the site has been modelled using a discrete 2D grid structure so that all distances in both x and y directions assume integer values, with the origin being at left bottom corner of the site. Each region will be covered by

unique designated agent such that each agent will start from the depot covering all regions under its purview and return back to its depot after finishing its tour. A region is completely defined by the grid cells which lie fully or partially within its area. A region M_i is completely defined by the set of grid cells $G_i = \{g_{ij}\}_{j=1}^{n_i}$, where g_{ij} denotes any grid cell lying fully or partially within the area of M_i, and n_i denotes the cardinality of G_i. Entire mission must be accomplished within an end to end deadline τ.

Problem Definition: Given a set of disjoint regions and a set of agents, the objective is to construct a tour for each agent which starts from the depot, covers a set of disjoint regions and comes back to the depot such that the total energy dissipated by all the aerial agents is minimized. Also, the time elapsed in completing the mission which is given by maximum time consumed by any touring agent, is upper bounded by the end to end deadline τ.

6.4 Heuristic Approach

In this section, a heuristic framework is proposed for solving the coverage problem over geographically distributed regions. The structure of the given problem immediately reveals that it can be considered as a composition of two constituent mutually dependent subproblems: 1) Inter-region path planning 2) Intra-region path planning. Inter-region path planning assigns regions to the agents, computes the traversal order for each agent and determines the entry and exit cell for each region. On the other hand, intra-region path planning determines the coverage path for each region.

Analysis shows that the problem is NP-hard in the strong sense and an optimal solution strategy to it will be prohibitively expensive in terms of solution generation times and required storage space. Further, it is also difficult to design a deterministic/greedy heuristic strategy which avoids solution enumeration, but can deliver satisfactory outputs under all realistic scenarios. Hence, it was resorted to a simulated annealing based statistical solution approach, which can generate considerably good solutions within reasonable time. Additionally, simulated annealing allows the flexibility of generating the best solution within a given upper bound on solution generation times as may be needed in possibly emergency situations where the proposed solution is applied.

Simulated annealing is a meta-heuristic search strategy which takes inspiration from the annealing process in the field of metallurgy. In the annealing process, the material is heated to a certain temperature and cooled in a controlled fashion to increase the size of crystals and reduce their defects. Similarly, simulated annealing is used to approximate the global minimum of a given function whose search space is discrete. It takes an initial solution and temperature as input. With each iteration, a random neighbouring solution is generated. The

algorithm accepts not only new solutions that lower the objective, but also the solutions which raise the objective, with a certain probability. By accepting the solutions which raise the objective, the algorithm is able to avoid getting trapped in local minima, so that it is able to explore globally a more exhaustive set of possible solutions. The temperature is systematically decreased according to a cooling schedule over the iterations of the algorithm. With the decrease in temperature, simulated annealing lowers the probability of accepting solutions which raise the objective. As a consequence, the extent of search gets reduced and the algorithm ultimately converges to a minimum.

To solve the coverage problem of disjoint regions using heterogeneous agents, a simulated annealing framework is deployed to find a global minimum which meets the objective of minimizing inter-region traversal and maximize intra-region coverage area. The proposed strategy is quite sensitive to the quality of the initial solution, and hence it becomes important to design an efficient initial solution generation scheme. We now discuss in detail the different components of our simulated annealing framework starting with the initial solution generation mechanism. The notations used in the approach are tabulated in Table 6.1

Table 6.1 Notations across Heuristic Approach

Variables	Definition
$\mathscr{A}$	set of n agents
$\mathscr{M}$	set of m regions
G_i	set of grid cells in region i
r_i	footprint of agent i
P_i	Power Rating of agent i
S_i	Speed of agent i
τ	Time Deadline
Ag_Cl	Tensor mapping agents to regions and cluster
$E\_C : \langle Cluster\ c, Agent\ a, Energy\_Cost\ e \rangle$;	Properties of Region Cluster

6.4.1 Initial Solution Generation

The initial solution strategy attempts to minimize the total energy cost of coverage and is designed as a composite solution to two subproblems, inter-region and intra-region path planning.

The total energy cost (Z) can be mathematically stated as:

$$Z = \sum_{A_i \in \mathscr{A}} \Big(\sum_{j \in Region(A_i)} C_{A_i}(j, j+1) + \sum_{j \in Region(A_i)} \hat{C}_{A_i}(j) \Big) \tag{6.1}$$

Algorithm 10: INITIAL SOLUTION GENERATION

Input:
1. $\mathscr{A} = \{A_1, \ldots, A_n\}$: set of agents
2. $\mathscr{M} = \{M_1, \ldots, M_m\}$: set of regions
3. <Footprint, Power Rating, Speed> $FPS = \{< r_1, P_1, S_1 >, \ldots, < r_n, P_n, S_n >\}$
4. Time Deadline : τ

Output: Initial Solution Sol, Ag_Cl, $Time_{Initial}$, $Z_{Initial}$, E_C

1 **Function** *Initial_Solution:*
2 Generate a set $C = \{C_1, ...C_n\}$ of n clusters;
3 Create an empty list E_C
4 $l = 0$; ▷ Index for list E_C
5 **for** *each cluster* $C_i \in C$ **do**
6 $TSP_i = NN(C_i)$; ▷ Generate ordered list of regions TSP_i, the tour over all regions in C_i.
7 $DIER = Dist(TSP_i)$; ▷ Calculate the total tour distance for TSP_i
8 $DIAR = \sum_{M_k \in C_i} G_{M_k}$; ▷ G_{M_k} is the number of grid cells in region M_k
9 **for** *each agent* $A_j \in A$ **do**
10 $IARD = \frac{DIAR}{r_j}$; ▷ Estimated intra-region distance for each agent A_j
11 $Time_j = \frac{IARD+DIER}{S_j}$; ▷ Time required by agent A_j to cover cluster C_i
12 **if** $Time_j \leq \tau$ **then**
13 %Feasible Solution %
14 $Energy\_Cost = Time_j \times P_j$;
15 **else**
16 %Infeasible Solution %
17 $Energy\_Cost = \infty$;
18 $E\_C(l) = \langle C_i, A_j, Energy\_Cost \rangle$;
19 $l = l + 1$;

where, $Region(A_i)$ is the set of regions allocated to the agent A_i for coverage, $C_{A_i}(j, j+1)$ is the cost incurred by agent A_i when traversing from region j to region $j+1$ and $\hat{C}_{A_i}(j))$ is the cost incurred by agent A_i when covering region j. Equation 6.1 contains two parameters where the first parameter considers the cost due to inter-region traversal and the second parameter considers the cost due to intra-region coverage.

Algorithm 10 presents the pseudo code for the initial solution generation scheme.

Line (2) **Generating Clusters** : Given a set of n heterogeneous aerial agents, the first step is to generate n disjoint clusters, each of which will be dedicated to a distinct agent. For a given set of spatially distributed regions, geographical proximities between them may be considered to be an important feature towards deciding the subset of regions which should be grouped together under the purview of a single agent. For the initial solution, n region clusters are generated using the K-means algorithm, which uses the separation between region centroids as a measure of distance in the clustering algorithm.

Algorithm 11: INITIAL SOLUTION GENERATION (CONTD.)

```
18 Function Initial_Solution:
20 Sort list E_C in ascending order based on E_C.e;
21 num_assign = 0;                          ▷ List of cluster assigned to agent
22 Create empty boolean lists Agent_Assigned, Cluster_Assigned (Initialize to False)
     and another list Ag_Cl, each of size n;
23 i = 1;                                   ▷ Index to read nodes in list E_C
24 Time_Initial = Z_Intial = 0              ▷ Initialize total energy cost and coverage time to zero
25 while num_assign ≠ n do
26     if agent_Assigned(E_C(i).a) == FALSE AND
        Cluster_Assigned(E_C(i).c) == FALSE then
27         Cluster_Assigned(E_C(i).c) = TRUE;
28         Agent_Assigned(E_C(i).a) = TRUE;
29         num_assign = num_assign + 1;
30         Ag_Cl(A_j) = C_i;
31         Z_Initial = Z_Initial + E_C(i).e;
32         if Time_E_C(i).a > Time_Initial then
33             Time_Initial = Time_E_C(i).a;
34     i = i + 1;
```

Lines (3-4) E_C **list creation**: After clustering the regions, the next important step is to assign to each cluster, the agent which will perform coverage of the cluster. It may be noted that the agents are heterogeneous being characterized by distinct speed, footprint and power rating values, and hence each agent may possibly incur distinct energy costs for covering any given cluster. Since, the objective is to minimize the *total energy cost Z* as calculated in Equation 6.1, the estimated energy cost corresponding to all agent cluster pairs needs to be computed in order to decide which mapping among them should actually be chosen. Energy cost information for each agent cluster pair is stored in a list E_C, each element of which is a three tuple $< Cluster\ c, Agent\ a, Energy\_Cost\ e >$.

Lines (6-8) **Inter-region Distance Computation:** Although the agents are heterogeneous, the total inter-region distance within a cluster remains the same irrespective of which agent the cluster is allocated to. This is because distance that must be traversed by an aerial agent to move from one region to another region is independent of its footprint size. The distance between a pair of regions is calculated by first determining the grid cells through which a line connecting two region centroids, pass. Then, the grid cells on this line are determined which intersect with the two region boundaries. Finally, eucledian distance between these two grid cells is used as a measure of the distance between two regions. Given the inter-region distances of a cluster, a TSP tour for the cluster, is generated using Nearest Neighbour (NN)

function. Algorithm 12 presents the pseudo code of nearest neighbour function. Initializing the depot as the first region in the tour, this algorithm progress in an iterative fashion including one more region in the tour in each iteration. In any given iteration, the region (say M_k) having the shortest distance from the region last added in the tour is selected and included into the current partial tour. The tour is completed when all the regions have been covered and the agent returns back to the depot. For each cluster, enumeration of the generated tour is returned back to the main function in the ordered list TSP_i (Lines 5-6). Line 7 calculates the total inter-region distance ($DIER$) and Line 8 determines the total number of grid cells over all regions associated with the current cluster C_i ($DIAR$). After calculating these cluster parameters, energy costs are determined with respect to all agents for this cluster.

Lines (9-19) **Energy Cost Computation of all Agent-Cluster pair**: Footprint r_j denotes the number of grid cells that a UAV can cover in parallel. If $DIAR$ denotes the total number of grid cells over all regions in cluster C_i, then $IARD(IARD = DIAR/r_j)$ denotes the minimum intra-region distance that the agent A_j must traverse to cover the entire cluster (Line 10). Subsequently, the time required by A_j for traversing cluster C_i is obtained as a ratio of total intra-region and inter-region distance and its speed S_j. In line 12, Algorithm 10 verifies whether the required traversal time for agent is lower than the given deadline (τ). If the required time is less than the deadline, Algorithm 10 considers it as a feasible solution and computes the energy cost for that particular agent. Given the power rating (P_j), the energy cost is calculated in line 14. For each agent, the computed energy cost is appended in list (E_C) with cluster id and agent id and corresponding energy cost value.

Line (20-34) **Assigning agents to clusters:** After computing the list E_C, Algorithm 11 sorts the list in ascending order of energy cost. The agent to cluster assignment occurs by sequentially exploring the E_C from the beginning. The agent id associated with the i^{th} element of E_C is assigned to the corresponding cluster id, if neither the agent nor the cluster id corresponding to $E\_C(i)$ has not already been allocated. Here, the boolean lists $Agent\_Assigned$ and $Cluster\_Assigned$ are used to determine whether an aerial robot/ cluster has already been assigned and is used to determine in O(1) time. To complete the assignment process in minimum time, two boolean lists $Agent\_Assigned$ and $Cluster\_Assigned$ are used. They are initially assigned to $False$. When each node in sorted list E_C is visited and based on the value of tuple $Energy\_Cost\ e$, the agents are assigned to clusters. The complexity of assignment of agents to clusters is O(n) where n is the number of clusters/ agents.

Intra-region path planning:

In this section, the CPP methodology is introduced that computes the coverage path for a single polygonal region, a pertinent step towards CPP computation for the entire target site

Algorithm 12: FUNCTION NN ( )

Input:
1. $C_i = \{M_1, \ldots, M_l\}$: a set region
Output: TSP_i: TSP Tour of all region in cluster C_i

Function NN**:**
  for *each region* M_j *in cluster* C_i **do**
    Calculate centroid $centroid_{ij}$ for region M_j
  $j = 1$;
  $TSP_i = \{j\}$; ▷ Tour starts at depot
  $W = C_i \backslash j$; ▷ Unvisited regions in Cluster C_i
  while $W \neq NULL$ **do**
    for *all regions* $M_k \in W$ **do**
      $Dist(k)$ = Distance between $centroid_{ik}$ and $centroid_{ij}$.
    Determine region M_k for which *Dist* is minimum. ▷ Select region M_k which is nearest to region j
    TSP_i = Append (TSP_i, M_k)
    $j = k$ ▷ The region k becomes the current region j
    $W = W \backslash M_j$
Return TSP_i

composed of disjoint regions. Any region i is modelled as a polygon which can be described by an ordered set of p_i vertices $\{v_1, \ldots, v_{p_i}\}$ with γ_h denoting the inner angle associated with the vertex h.

The spatial resolution of each grid cell is equivalent to the minimum sensing range among the fleet of agents $\mathscr{A}$. This implies that there exist agents, which, if placed at the centre of a grid cell, can sense more than one grid cell. Let us consider three UAVs hovering over a single grid cell as elucidated in Fig. 6.1. UAV_1 with the smallest sensing range has a footprint of a single green cell. UAV_2 can cover 9 grid cells (yellow grid cells along with green cell). UAV_3 can cover 25 grid cells (red, yellow and green grid cells). The Boustrophedon motion simplifies the path design and has been adopted widely by many CPP methodologies [43]. For a given convex polygon region, Boustrophedon motion that covers the region can be computed using a set of parallel lines called as support lines (SL) drawn horizontally through the vertices defining the span of the region as shown in Fig. 6.2. An intra-region coverage path for a given region is computed using function *IRP* illustrated in Algorithm 13. An intra-region coverage path for a given region is impacted by many parameters like region visiting order, entrance and exit locations. To address this issue, all the parameters are considered while computing the CPP for a single region. Let us suppose an aerial agent is traversing from region $j-1$ to region j and coverage path for region j needs to be computed.

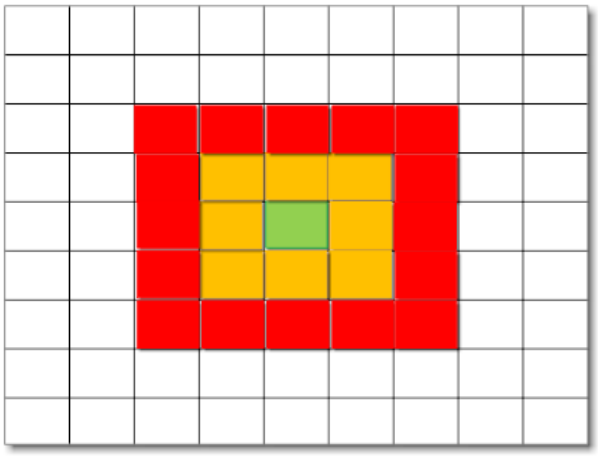


Fig. 6.1 Sensing Range of three UAVs are hovering at the top of the green cell.

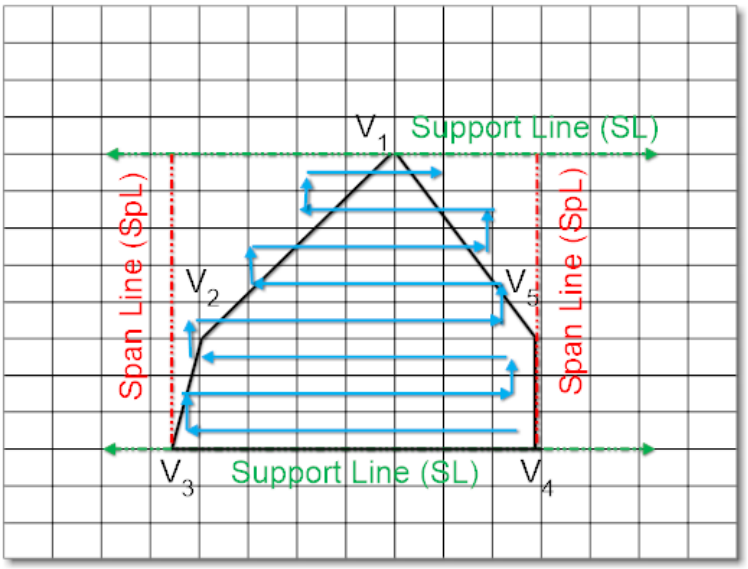


Fig. 6.2 A Boustrophedon motion (green lines) that covers a rectangular region with support lines represented as dashed grey lines.

Algorithm 13: FUNCTION IRP ( )

Input:
1. Region j for intra-region visit
2. Previous region $j-1$ and next region to be visited
3. Footprint size r_j for agent A_j

Output: Coverage Path for region j

Function `IRP`**:**
- %Entrance point%
- *Line_V*: a set of vertices on support line **for** *each vertices* $v_i \in V$ **do**
  - **if** *vertices* v_i *lies on support line* **then**
    - *Line_V* = *Line_V* $\cup v_i$;
- Entrance Point : Select vertex from *Line_V* which is closest to region $j-1$;
- %Back Forth Motion%
- **while** *Entire region j is covered* **do**
  - **while** r_j *rows are not covered* **do**
    - $Path_j =< G_1, \ldots, G_k >$: Move horizontally with r_j cells;
  - $Path_j =< G_{k+1}, \ldots, G_l >$: Move r_j cells vertically after covering entire rows under purview r_j;
- Return $Path_j$;

The entrance of region j as the nearest vertex to region $j-1$ lying on SL is determined. After computing the entrance, the Boustrophedon motion is computed, which is a back and forth motion from one level grid cell to another level moving towards the vertex lying on the opposite SL.

For the coverage by the agents having a footprint size larger than the spatial resolution of a single grid cell, instead of moving to a neighbouring cell using usual Boustrophedon motion, r_k cells in the horizontal direction is determined, where $r_k \times r_k$ is the sensing range of the agent k. Once a row is covered horizontally, the agent jump r_k cells vertically. Thus, the usual Boustrophedon motion with a step size of $r_k \times r_k$ is employed. This approach will work for cases where the regions do not have a rectangular boundary. For instance, let us consider a pentagon where V_4 is the entrance point as shown in the Fig. 6.3.

In this case, given grid based region sampling, the aerial agent ensures the complete coverage of entire second row first and then move r_k cells downwards.

Lemma 1: Given a convex polygon region and footprint size of an aerial agent k as $r_k \times r_k$*, the Boustrophedon motion generated by our proposed intra-region path computation technique guarantees full coverage of the region.*

Proof: Any given region can be decomposed as a collection of grid cells. If a UAV with its footprint size $r_k \geq w$, covers the region by employing Boustrophedon motion with a step

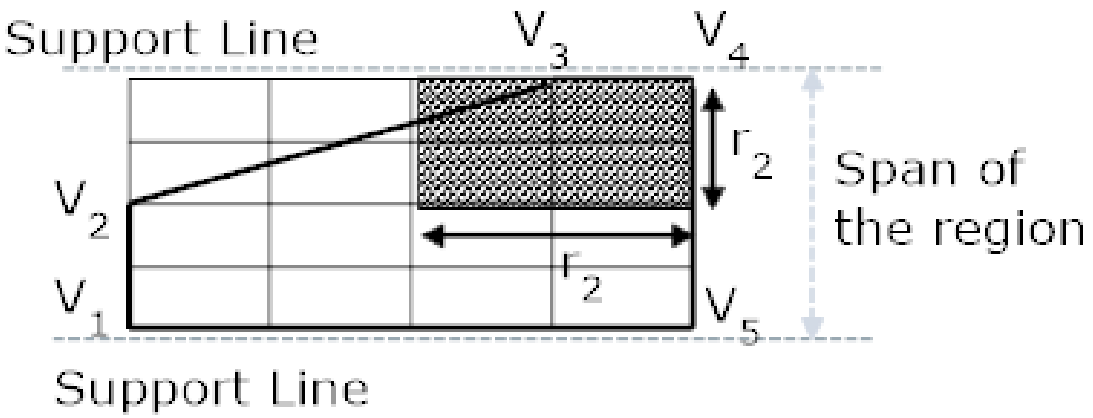


Fig. 6.3 Coverage path planning of a pentagon by an aerial agent with $r_2 \times r_2$ coverage area.

size r_k, it ensures that each grid cell within this convex polygon region is covered at least once, that is full coverage of the region.

6.4.2 Simulated Annealing

After discussing the initial solution generation procedure and inter-region and intra-region coverage mechanisms, a step-wise description of the overall simulated annealing scheme presented in Algorithm 14.

Lines (2-6) **Initialization**: The procedure starts by assigning the initial solution as the current solution, while η ($\eta = 500$) and $Temp$ ($Temp = 10^4$) store maximum number of allowed iterations and initial temperature, respectively. The best energy cost Z_{Best} is initialized as the current energy cost Z_{Cur}.

Lines (8-10) **New Solution Generation**: Each iteration of the while loop in lines 7-32 generates a new solution. The configuration of a solution is given by ordered set of regions and the path traversed by each agent within its assigned cluster. Before generating the next solution, the current solution Sol_{Cur} and its energy cost Z_{Cur} is stored in Sol_{Prev} and Z_{Prev}. Now, a new solution using Function NEW_SOL is determined which generates a new neighbouring solution using current solution Sol_{Cur} as discussed in Algorithm 15. For generating a new solution, the algorithm randomly selects two distinct clusters C_i and C_k in current solution Sol_{Cur} and then randomly selects a region M_j in the selected cluster in Line 2-4. The selected region M_j is moved from cluster C_i to C_k in the new solution. Finally, new energy cost Z_{Cur} and coverage time $Time_{Cur}$ is calculated.

Lines (12-24) **Accepting the Solution**: The new solution is directly accepted, if it meets deadline $(T_{Cur} < \tau)$ and its energy cost is better than that of the previous solution. Further, in this case, if the energy cost of the current solution is better than the energy cost of the

Algorithm 14: FUNCTION S_ANNEALING ( )

Input: 1. $\mathscr{A}, \mathscr{M}$: set of agents and regions
3. $< Footprint, PowerRating, Speed > FPS = \{< r_1, P_1, S_1, \ldots, < r_n, P_n, S_n >\}$
4. τ: Deadline within which the site must be covered.
5. $Initial\_Sol$,$Z_{Initial}$,$Time_{Initial}$: Initial solution, Initial energy and time and Energy list.
Output: Sol_{Cur}, Z_{Cur} and $Time_{Cur}$

Function *S_Annealing:*
  1) $Sol_{Cur} = Initial\_Sol$; ▷ Initialize current solution Sol_{Cur} as Initial Solution
  2) $\eta = i = 500$; ▷ η is maximum number of iterations and i is the iteration index
  3) Temp = 10^4; ▷ Initially temperature is set to a high value.
  4) $Z_{Best} = Z_{Cur} = Z_{Initial}$;
  5) $Sol_{Best} = Sol_{Cur}$ ▷ Initialize initial solution along with its energy cost as best solution and its energy.
while $i > 0$ **do**
  $Sol_{Prev} = Sol_{Cur}$;
  $Sol_{Cur}, Z_{Cur}, Time_{Cur}$ = NEW_SOL (Sol_{Cur}, Ag_Cl, FPS, E_C, Z_{Cur}, $Time_{Cur}$) ▷ Generate new solution along with its associated energy cost and coverage time.
  $diff = Z_{Cur}$ - Z_{Prev}; ▷ Difference in energy cost between current and previous solutions.
  if $diff < 0$ **then**
    if $Time_{Cur} \leq \tau$ **then**
      % Accept current Solution Sol_{Cur} %;
      if $Z_{Cur} < Z_{Best}$ **then**
        $Z_{Best} = Z_{Cur}$;
        $Sol_{Best} = Sol_{Cur}$; ▷ Accept current solution as best solution.
    else
      $\delta = Time_{Cur} - \tau$; ▷ Difference by which time deadline is exceeded.
      $\lambda = rand(0.5, 1)$; ▷ Generate a random number between 0.5 and 1.
      $Metropolis = e^{\frac{-\delta}{Temp}}$;
      if $Metropolis > \lambda$ **then**
        % Accept current solution Sol_{Cur} %
      else
        %Discard current solution Sol_{Cur} %
  else
    $Metropolis = e^{-diff/Temp}$;
    if $Metropolis > \lambda$ **then**
      %Accept current solution Sol_{Cur}%
    else
      % Discard current solution Sol_{Cur}%
  $i = i - 1$;
  $Temp = \frac{i}{\eta} \times Temp$;

best solution obtained so far, then the current solution is updated as the current best solution. Otherwise, if the new solution violates deadline then it is probabilistically accepted only if the value of metropolis ($Metroplois = e^{-(Time_{Cur}-\tau)/Temp}$) is greater than λ , a random number between 0.5 and 1. It may be noted that the metropolis is a variable whose value is bounded between 0 and 1 and it reduces as the temperature cools over time. Hence, the probability of accepting a deadline violating solution also reduces over time.

Lines (26-32) **Accepting Solution with higher energy cost:** If the energy cost of the current solution is worse than that of the previous solution, the current solution is probabilistically accepted if the value of metropolis variable ($Metroplois = e^{-(Z_{Cur}-Z_{Prev})/Temp}$) is greater than λ, similar to the mechanism for accepting a deadline violating solution as discussed above. Finally, the current iteration ends by updating the iteration count and the value of temperature $temp$.

Algorithm 15: FUNCTION NEW_SOL ( )

Input: 1. Sol_{Cur}: Input solution
2. Ag_Cl, E_C: Agent cluster pair, Energy cost list
3. $< Footprint, PowerRating, Speed > FPS = \{< r_1, P_1, S_1, \ldots, < r_n, P_n, S_n >\}$
4. $Z_{Cur}, Time_{Cur}$: Energy cost and coverage time for input solution.
5. w : width of grid cell

Output: New Solution Sol_{Cur}, Energy Cost Z_{Cur} and Coverage Time $Time_{Cur}$

1 **Function** NEW_SOL**:**
2 Select a cluster C_i randomly in Sol_{Cur};
3 Select a region M_j randomly in cluster C_i;
4 Select another cluster C_k randomly in Sol_{Cur} such that $C_k \neq C_i$;
5 Remove region M_j from cluster C_i and add to cluster C_k;
6 **for** *each* $H \in \{i,k\}$ **do**
7 $TSP_H = NN(C_H)$ ▷ Generate inter-region path
8 $IER = Dist(TSP_H)$; ▷ Calculate total inter-region distance
9 $v = Ag\_Cl(H)$ ▷ Agent-id assigned to cluster-id H
10 **for** *each region* $M_k \in C_H$ **do**
11 $Path_k$: intra-region path using IRP(M_k,r_v);
12 $IAR = \sum_{M_j \in C_H} Path_j \times w$; ▷ Total intra-region distance
13 $total\_dist = IER + IAR$;
14 $Time_v = \frac{total\_dist}{S_v}$; ▷ Time required by agent A_v to cover site
15 $Z_v = Time_v \times P_v$; ▷ Energy cost incurred by agent A_V
16 $Z_{Cur} = Z_{Cur} - E\_C(v).e + Z_v$ ▷ Update energy cost Z_{Cur}
17 **if** $Time_v > Time_{Cur}$ **then**
18 $Time_{Cur} = Time_v$ ▷ Update coverage time T_{Cur}
19 Return $Sol_{Cur}, Z_{Cur}, Time_{Cur}$;

6.5 Experimentation and Results

In this section, a simulation study is performed to evaluate the performance of our proposed simulated annealing approach and compare the same against an existing genetic algorithm scheme [43]. The common parameters, number of iterations and population size, are set 500 and 100, respectively. These values were selected from [43]. All the algorithms are implemented using MATLAB, and the experiments are performed on a DELL XPS IDV8QVO with Intel Core i5, 8GB memory and 225GB storage.

6.5.1 Experimental Setup

The input data for the experiments have been generated using the following framework:

1. Region Characteristics: The data corresponding to the regions for a site which needs to be covered has been generated randomly. Sites with four different values for the number of regions viz. $\{25, 50, 75, 100\}$, are generated. The regions in the site are assumed to be distributed around an origin marked by the coordinates (0,0). A new region is created by first tentatively selecting a centroid for it. The coordinates of this centroid are obtained by selecting its distance from the origin through a random distribution U1 $[0, 200]$ and also determining the angle that it subtends on the x axis, from another uniform random distribution U2 $[0, 359]$.

   A new region is created using function $nsidedpoly()$ [162]. This function takes as input the coordinates of the centroid (obtained as discussed above), the number of vertices and length of each sides of the region to be generated. $nsidedpoly()$ returns a regular polygon based on the region parameters provided as input. The number of vertices is generated from a uniform random distribution U3 $[3, 10]$. The length of the side of a region is generated from normal distribution having standard deviation as $\sigma_{side} = 2.5$ and with four different values for the mean $\mu_{side} = \{5, 10, 15, 20\}$. The new region is actually accepted if it does not overlap with any other region already generated; otherwise, it is discarded. These steps are repeated unless and until desired number of regions are created. In case of a rejection, the steps to create a new region are repeated. The above steps are continued until the required number of regions have been generated. The size of a region is measured in terms of number of grid cells that it contains. The size of each grid cell is assumed to be $1 \times 1\ m^2$.

   As discussed before, the overall operation has an associated deadline by which coverage of the target site must be completed. For each dataset, the time ($T_{Initial}$) required for

the coverage in the initial solution generated by Algorithm 10 is used as a deadline ($\tau = T_{Initial}$).

2. Agent Characteristics: The data corresponding to the agents or agents which are deployed to cover a site is generated in the following fashion. Four different values for the number of agents has been considered $\{4,6,8,10\}$. All the agents(or aerial agent) are assumed to have square footprints whose areas are expressed as an integral number of grid cells. Agents can have different footprint sizes. This size, in terms of grid cells, can assume anyone of three values $\{1,4,9\}$. The speed of each type of agent is generated from a normal distribution with mean $\mu_{speed} = 5$ and standard deviation $\sigma_{speed} = 1$. The power consumption for each UAV has also been generated using a normal distribution with mean $\mu_{power} = 100$ and standard deviation $\sigma_{power} = 25$.

3. Evaluation Metrics: The principal metrics used for evaluation of the proposed simulated annealing framework are *'Normalized Energy Cost'* and *'Energy Efficiency'*. *Normalized Energy Cost (Z_{norm})* can be mathematically formulated as:

$$Z_{norm} = \frac{Z_{Cur}}{Z_{max}} \tag{6.2}$$

where, Z_{Cur} denotes the total actual cost incurred by a solution derived using the simulated annealing framework. On the other hand, Z_{max} is a measure of the maximum energy that will be consumed in the traversal for the target site by the agent with highest energy consumption. *Energy Efficiency (EE)* can be mathematically formulated as:

$$EE = 1 - \frac{Z_{Cur}}{Z_{Initial}} \tag{6.3}$$

where, Z_{Cur} and $Z_{Initial}$ denote the total cost incurred by a solution derived using simulated annealing and total cost incurred by the initial solution, respectively.

6.5.2 Results

Varying the number of regions

In the first experiment, a target site is considered with varying number of regions as $M = \{25,50,75,100\}$. The number of agents and mean length of each side has been fixed to 6 agents and 7.5. To illustrate the role of heterogeneity in the fleet of agents, following two cases are considered:

1. Case 1: A homogeneous fleet of agents such that all the UAVs are of similar type.

2. Case 2: A heterogeneous fleet of agents consisting of all three types of UAVs.

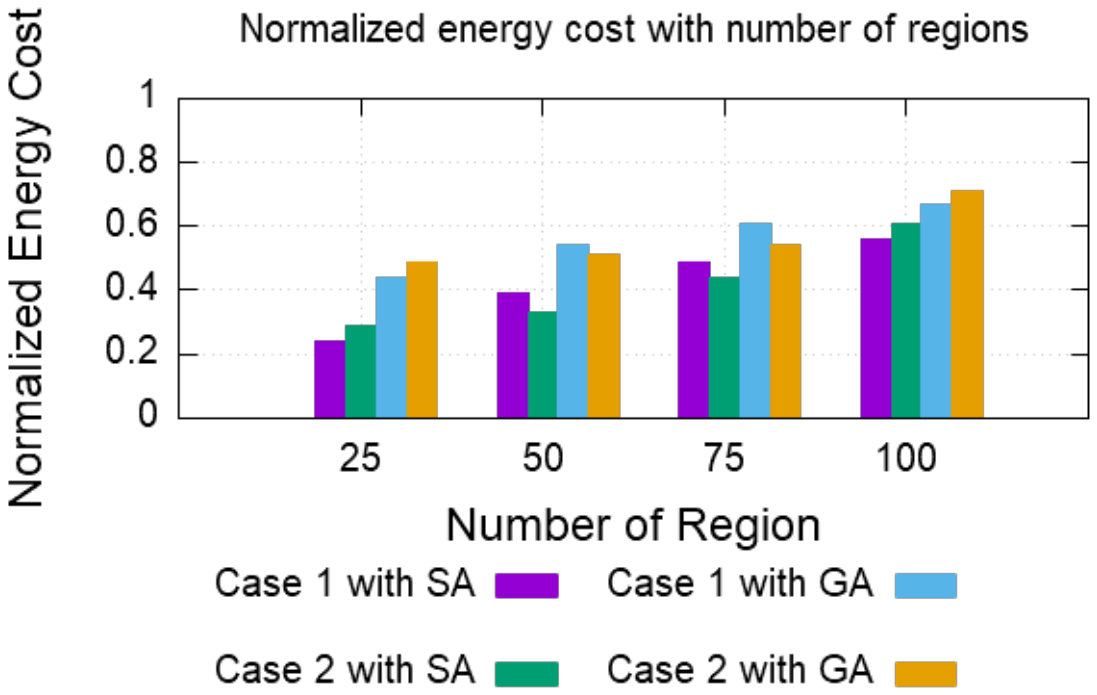


Fig. 6.4 Analysis of normalized energy cost with respect to number of regions.

Fig. 6.4 shows the normalized energy cost with respect to different number of regions. As the number of regions increases, the normalized energy cost increases. Since, the maximum energy cost is fixed, the increase in normalized energy cost indicates increase in total energy cost Z_{Cur}. The total energy cost Z_{Cur} is the summation of inter-region traversal cost and intra-region traversal cost as illustrated in Algorithm 14. With increase in the number of regions, the total intra-region and inter-region energy cost also increases for each agent. The results also show that solutions derived using the simulated annealing framework is able to reduce normalized energy cost Z_{norm} by 20-40%, with respect to energy cost associated with genetic algorithm.

Fig. 6.5 shows the variation of 'Energy Efficiency (EE)' with respect to number of regions. The trends reveal that the simulated annealing framework is able to find a solution with $50-60\%$ lower total energy cost with respect to the initial solution. The performance of our proposed approach using simulated annealing is better than genetic algorithm both in terms of Z_{norm} and EE. The better performance of simulated annealing can be attributed to its ability to search for better solutions in the search space by judiciously accepting bad

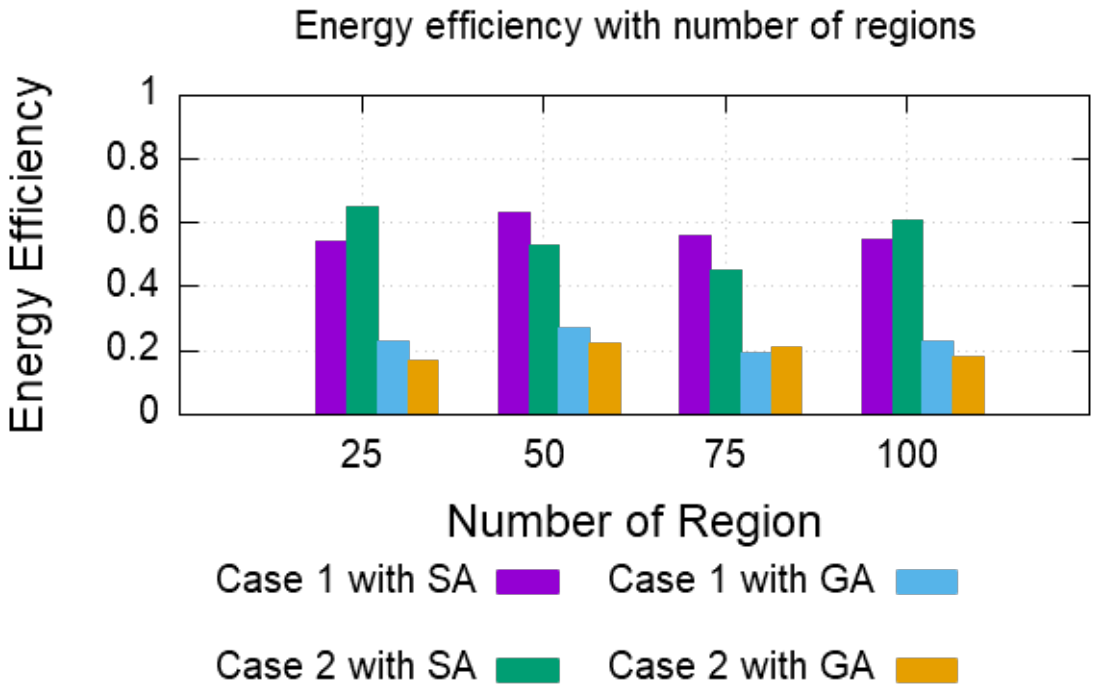


Fig. 6.5 Analysis of energy efficiency with respect to number of regions.

solutions at the initial stages which is not the case for genetic algorithm that always attempts to satisfy a certain fitness function.

Varying the number of agents

In the second experiment, a target site of 100 disjoint regions is considered. The number of agents are varied from 4 to 10. Fig. 6.6 shows the variation of normalized energy cost with respect to change in the number of agents. As the number of agents increases, the average Z_{norm} decreases.

This may be attributed to the fact that with the increase in the number of agents, the cluster became more compact reducing the total tour length that must be covered by the set of agents. A lower tour length in turn, brings down the energy cost. It is also observed that a 20-50% reduction is achieved by SA against GA.

Fig. 6.7 analyses the energy efficiency with respect to number of agents. The trends reveal that the total energy cost Z_{Cur} decreases by 40-60% with respect to the initial solutions. As for the previous cases, SA is seen to perform better than GA, primarily due to its capability to more comprehensively explore the solution space by judiciously accepting bad solutions.

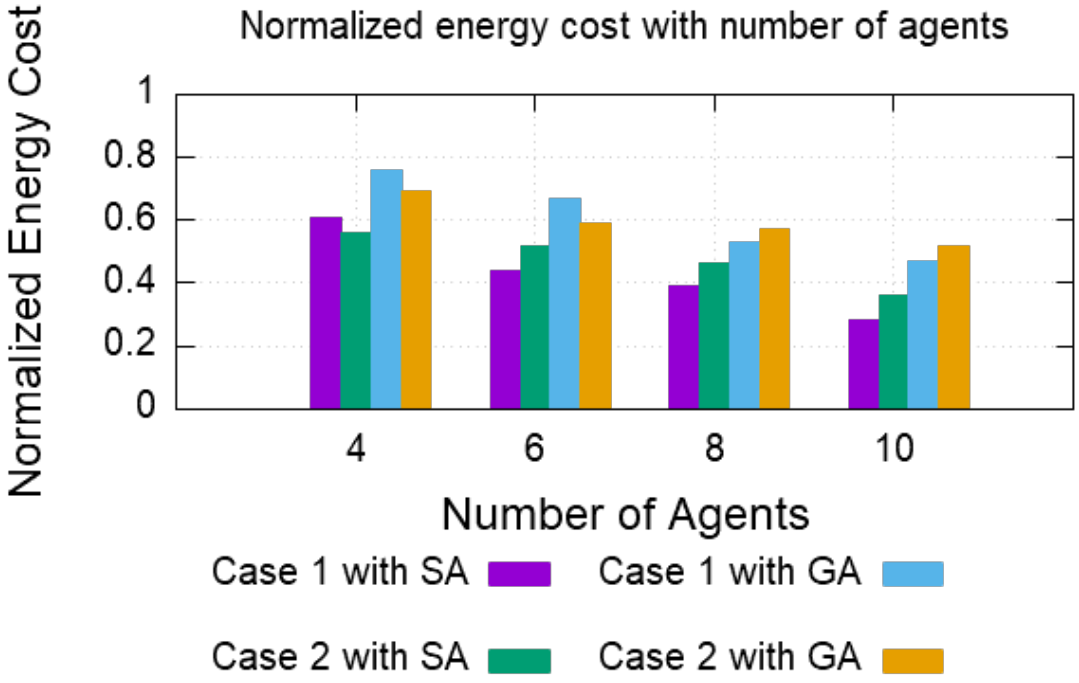


Fig. 6.6 Analysis of normalised energy cost with respect to number of agents.

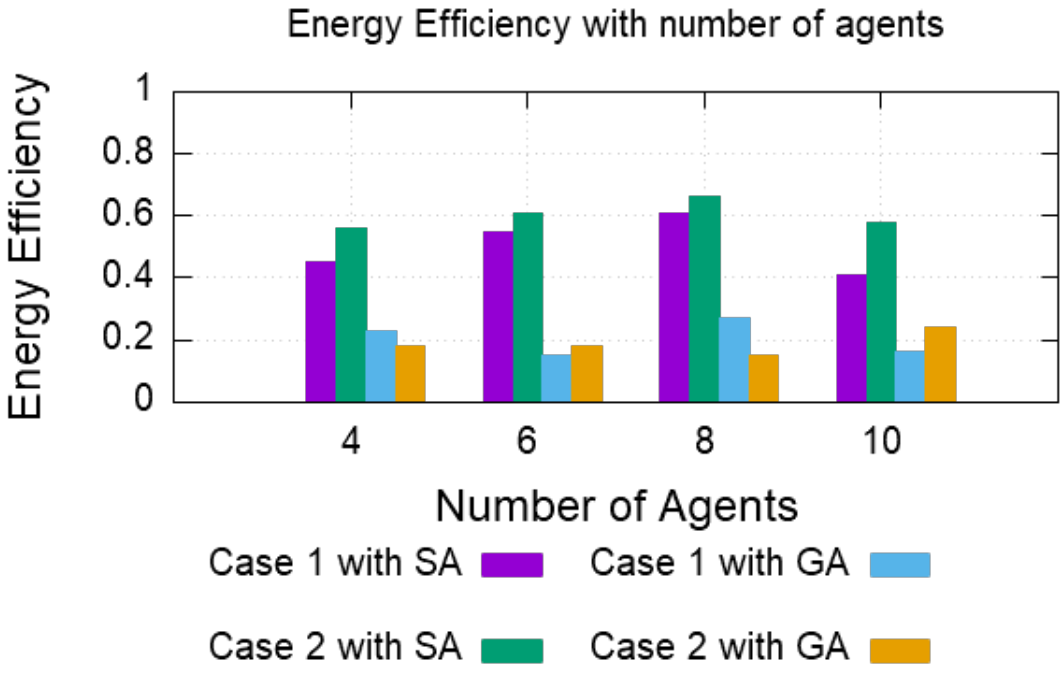


Fig. 6.7 Analysis of energy efficiency with respect to number of agents.

Varying the area of regions

In the third experiment, a target site of 100 disjoint regions is considered. The number of agents are fixed at 6. The main aim of this experiment is to analyze the normalized energy cost (Z_{norm}) with respect to varying average area of regions. Regular polygonal regions with specific average area is generated for any dataset in our experiments by fixing the average number of vertices (All vertices are generated from a single uniform distribution U3) and the mean length corresponding to the region sides (region sides are generated from normal distributions). For instance, a regular pentagon with sides of length 7.5 m will have an area of 96.78 m^2. In this experiment, the average area of regions is varied over different datasets by varying the mean length of sides ($\mu_{side} = \{5, 10, 15, 20\}$) while keeping the standard deviation fixed ($\sigma_{side} = 2.5$) for the normal distributions from which the sides are generated. The average number of vertices remain unaltered for all datasets.

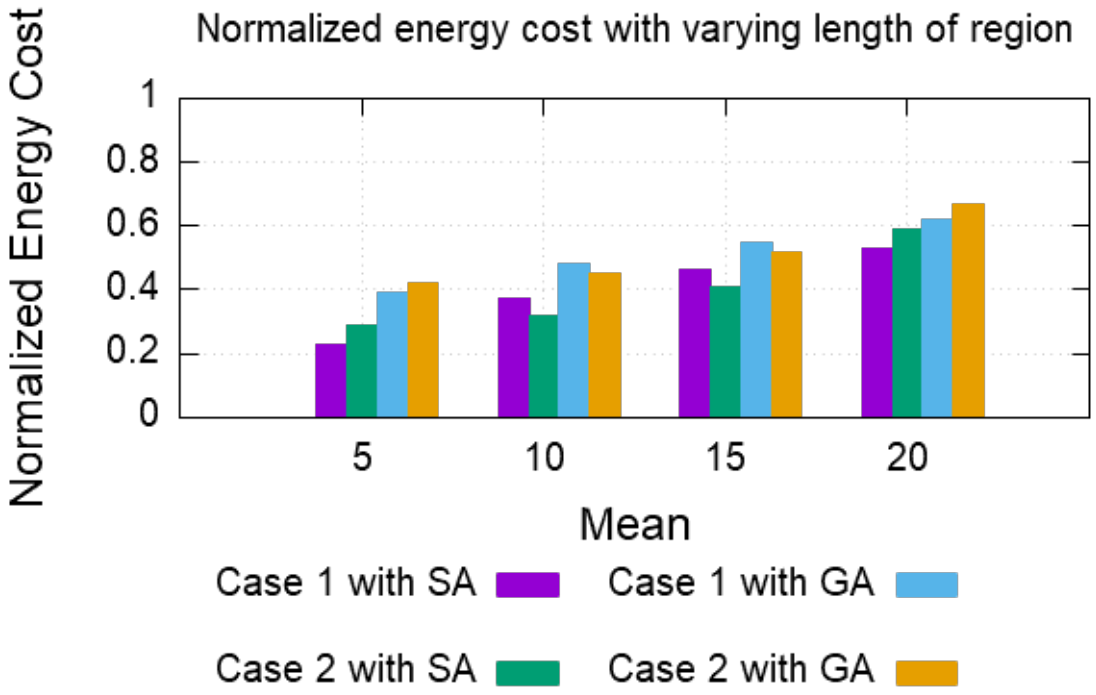


Fig. 6.8 Analysis of normalized energy cost with varying the length of regions.

Fig. 6.8 shows the variation of normalized energy cost with varying sizes of regions. The results reveal that as the average length of sides increase (effecting a corresponding increase in region area), the normalized energy cost increases. The reason behind this trend can be attributed to the fact that total energy cost Z_{Cur} is dependent upon inter-region and intra-region traversal costs. As the lengths of the sides increase, intra-region traversal costs

also increase. However, the normalized energy cost for solutions derived using simulated annealing reduces by 15-40% in comparison to the solutions derived using genetic algorithm.

As illustrated in previous experiments, the performance of the simulated annealing framework is better in comparison to the genetic algorithm, as it is able to often find close to optimal minima in terms of total energy costs Z_{Cur}, incurred in the traversal of the target site.

6.6 Summary

In this chapter, a heuristic approach for CPP of multiple disjoint regions using a heterogeneous fleet of aerial agents is presented. An initial solution is first generated where clusters of regions are constructed using the K-Means clustering algorithm. For each cluster, an aerial agent is assigned for coverage of regions within the clusters. The initial solution is an approximation solution which is improved using a simulated annealing framework. The properties of the heuristic approach is investigated using simulation studies. The experiments conducted reveal the efficacy of the algorithm in comparison to genetic algorithm.

Chapter 7

Conclusion

Coverage Path Planning (CPP) is a well-established, challenging and multi-dimensional research problem in the domain of motion planning. It is the process of determining a path that allows the robot to map/scan/cover the given area or volume while avoiding obstacles. The intrinsic parameters of robot like on-board energy, localisation and sensing capabilities, and extrinsic parameters like environmental conditions pose significant challenges making CPP a NP-hard problem.

CPP originated in 1994 from the application of autonomous floor cleaning [176]. Although the time-span of three decades has passed, this problem is still relevant. This can be attributed to the fact that CPP is a dynamic and an application-oriented problem. The application of CPP is wide-spread across multiple domains like agriculture [177–179], environmental inspection [180, 181], floor cleaning [12], terrain reconstruction [107, 182], demining [183] and lawn mowing [23]. This wide-spread application of CPP allows researchers from diverse fields like control theory, mechanics, computational and differential geometry, and computer science to collaborate. To be part of this research community and observe it *grow by leaps and bounds* in the past four years, the author can conclude that CPP will become more exigent in the near future. The technological developments in hardware, newly proposed optimisation techniques, and inter-disciplinary research, will pose a new set of challenges for all the communities like research, business and engineering associated with the domain of CPP.

This thesis is a fusion of identified research challenges within CPP. The research can be categorized in two classes: (a) Mobile Robots based CPP (Chapter 3), and (b) Aerial Robot based CPP (Chapter 4-6). In the latter category, the upcoming problem of area coverage of multiple distributed regions is tackled. Three research tracks that originate from this category are : (a) Precedence Provision (Chapter 4), (b) Energy Constraint (Chapter 5), and (c) Heterogeneous Fleet (Chapter 6). These chapters are mutually-independent and can serve

as a building block for future research related to CPP. This chapter aims to summarize the research contribution in this thesis and at the same time provide the readers with the direction of future research.

7.1 Contribution Summary

This thesis encapsulates the author's research that was performed during the doctoral studies. The major contributions of this thesis can be enlisted as follows:

1. Firstly, the thesis traces the origin of RAS and its evolution after industrial revolution. The current state of autonomous robots are discussed and challenges going forward in INDUSTRY 4.0 are highlighted. The major challenges in long term autonomy of autonomous robots specifically Coverage Path Planning (CPP) are explored. The fundamental concept of CPP and its positioning with respect to long term autonomy research are described.The literature review in this thesis not only covers CPP techniques, but also presents a mathematical background of optimisation framework and classical NP-hard problems. The gap in literature is clustered into four research challenges pertaining to CPP.

2. Chapter 3 proposes a novel offline-online strategy that equips mobile robots with the ability to autonomously plan a coverage path and reach the static target effectively and efficiently. The extreme environment in which autonomous robots operates imposes a strict time constraint. The offline stage of the strategy geometrically identifies information enriched regions to build a road map. This road map is effectively used to minimise time in online stage where sensor-based coverage is performed.

3. In Chapter 4, the author explores the problem of area coverage of extreme environments using an aerial robot. Extreme environment poses two main challenges for aerial robots. The first challenge is that target site comprises of distributed regions. Thus, apart from computing coverage path of each region, the decision makers have to also compute inter-region path as well. The second challenge encompasses sensor degradation due to presence of radioactive elements. The sensor characterisation conducted at Rutherford Appleton Laboratory reveals that annealing is a prime requirement for the covering robot. Therefore, the aerial robot is equipped with precedence provision while traversing the extreme environment site.

4. In Chapter 5, another interesting challenge of limited on-broad energy of aerial robot during area coverage operations is addressed. This chapter brings to the table a

strategical and practical framework where aerial robot aims to achieve near-optimal area coverage due to path length limitation caused by the energy constraint. This framework differs from existing strategies where multiple trips to battery stations are preferred over critical factors like time and practical application.

5. Chapter 6 moves a step forward by solving area coverage over disjoint regions within constrained time bounds, an important problem especially for emergency coverage tasks such as post-disaster relief, military surveillance, search and rescue missions. The proposed technique takes into consideration that the critical mission forces decision makers to deploy all the resources which results in area coverage by a heterogeneous fleet.

7.2 Where Do We Go Next?

After carrying out intensive research spanning 4 years, a poignant question that comes to the mind of the author, *'Where do we go next?'*. The prime objective of asking this question is to point out the research gaps and ideas. So that, a dialogue in the research community of CPP can be initiated.

1. The offline-online strategy is a novel attempt at combining two problems like CPP and OSP with diverse objectives. The trade off between search time and area coverage is task dependent. Thus, a major challenge exists as a novel strategy needs to designed each time a new application combining CPP and OSP comes up. It will be worth exploring development of a generic framework where optimisation parameters and degree of trade-off can be learned given any coverage tasks.

2. As a follow up to the above listed point, an offline-online strategy provides a hypothesis where offline stage and online coverage share a symbiotic relationship. However, there is significant room for improvement in identification of information which needs to be shared among both the stages for improving overall performance.

3. Moreover, the offline stage is dependent on the architectural layout. The focus of research community on information extraction using floor plan has led to development of datasets for indoor environments like universities and offices [135]. However, the extreme environmental sites such as nuclear plants have different construction and architectural norms. Therefore, the feature extraction from existing datasets cannot be used in the offline stage of GO-CPP in extreme environment.

4. Chapter 4 conducted sensor characterisation of Kinect in the presence of fast neutrons. This allowed for identification of precedence provision for area coverage. Extensive experiments need to be performed for characterisation of different types of sensors. This is a challenging task as access to such facility is limited and irritation process is hazardous in nature. However, these experiments will allow additional requirements to be identified for robots operating in extreme environmental conditions.

5. Furthermore, repeated experiments of sensor characterisation will also allow the researchers to propose a generic sensor degradation model that can later be simulated in frameworks like Gazebo for accurate representation of extreme environment.

6. The proposed strategy of the near optimal area coverage in Chapter 5 suffers from the curse of dimensionality. This proves to be a major research gap. Thus, a key extension to the work would be to propose a computationally inexpensive solution.

7. Recently, deep learning techniques have been used to solve TSP and CPP individually and also research works focusing on TSP-CPP as an area coverage problem explore it through heuristic methods. However, the research community can use deep learning techniques, more specifically deep reinforcement learning, for solving TSP-CPP problem.

8. The author would also like to recommend the future direction that this research can possibly take. Associated fields have seen advancements such as Conventional Neural Networks which serves as a generic framework for computer vision applications. CPP research needs to converge towards a generic framework and approach that encompasses major constraints and provisions.

9. Economically, deployment of robots in extreme environment has proven to be an expensive solution. Therefore, the research strategy needs to focus on developing an intelligent environment which will allow collaboration between conventional robot and humans. The upcoming technological development in the area of mixed reality can be extreme useful in training the robot.

10. Another important finding of this research is unreliable nature of visual sensor data specifically for the purpose of robot localisation. In order to utilise this sensor data, a Cyber Physical System (CPS) can be a plausible solution. The CPS based approach has demonstrated considerably good results for long term autonomy due to the ability of effective collaboration.

11. Lastly, the need of the hour is to bring all the stakeholders of various CPP applications on the same page. Even though, the research community has published ample work on CPP, the commercial viability of CPP applications still raises a lot of questions in the business world. The failure of some of the past companies to make profits is the main reason that venture capitalists are hesitant in backing robotics startups when compared to other AI powered technological applications [184]. The author is of strong opinion that the lack of emphasis on user research while designing autonomous operations of robots is one of the reason behind the demise of many robotic startups.

References

[1] N. Tesla, "A machine to end war," *Liberty Magazine*, vol. 9, pp. 5–7, 1935.

[2] H.-J. Warnecke, H. Gzik, and W. Utner, "From the industrial robot welding cell to the industrial robot welding system," *The International Journal of Advanced Manufacturing Technology*, vol. 1, no. 4, pp. 25–38, 1986.

[3] A. Gasparetto and L. Scalera, "From the unimate to the delta robot: the early decades of industrial robotics," in *Explorations in the History and Heritage of Machines and Mechanisms*. Springer, 2019, pp. 284–295.

[4] E. Appleton and D. J. Williams, *Industrial robot applications*. Springer Science & Business Media, 2012.

[5] M. A. K. Bahrin, M. F. Othman, N. H. N. Azli, and M. F. Talib, "Industry 4.0: A review on industrial automation and robotic," *Jurnal teknologi*, vol. 78, no. 6-13, 2016.

[6] N. K. Lee, "Total automation: The possibility of lights-out manufacturing in the near future," *Missouri S&T's Peer to Peer*, vol. 2, no. 1, p. 4, 2018.

[7] J. Wirtz, P. G. Patterson, W. H. Kunz, T. Gruber, V. N. Lu, S. Paluch, and A. Martins, "Brave new world: service robots in the frontline," *Journal of Service Management*, 2018.

[8] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, and W. Burgard, *Principles of robot motion: theory, algorithms, and implementations*. MIT press, 2005.

[9] H. Choset, "Coverage for robotics–a survey of recent results," *Annals of Mathematics and Artificial Intelligence*, vol. 31, no. 1, pp. 113–126, 2001.

[10] F. Yasutomi, M. Yamada, and K. Tsukamoto, "Cleaning robot control," in *1988 IEEE International Conference on Robotics and Automation*, vol. 3, 1988, pp. 1839–1841.

[11] R. Bormann, F. Jordan, J. Hampp, and M. Hägele, "Indoor coverage path planning: Survey, implementation, analysis," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 1718–1725.

[12] A. K. Lakshmanan, R. E. Mohan, B. Ramalingam, A. V. Le, P. Veerajagadeshwar, K. Tiwari, and M. Ilyas, "Complete coverage path planning using reinforcement learning for tetromino based cleaning and maintenance robot," *Automation in Construction*, vol. 112, p. 103078, 2020.

[13] B. Zhang, J. Wu, L. Wang, and Z. Yu, "Accurate dynamic modeling and control parameters design of an industrial hybrid spray-painting robot," *Robotics and Computer-Integrated Manufacturing*, vol. 63, p. 101923, 2020.

[14] H. V. Pham, P. Moore, and D. X. Truong, "Proposed smooth-stc algorithm for enhanced coverage path planning performance in mobile robot applications," *Robotics*, vol. 8, no. 2, p. 44, 2019.

[15] W. Chen and D. Zhao, "Path planning for spray painting robot of workpiece surfaces," *Mathematical Problems in Engineering*, vol. 2013, pp. 1–6, 2013.

[16] R. K. Katzschmann, J. DelPreto, R. MacCurdy, and D. Rus, "Exploration of underwater life with an acoustically controlled soft robotic fish," *Science Robotics*, vol. 3, no. 16, 2018.

[17] D. Zhu, C. Tian, B. Sun, and C. Luo, "Complete coverage path planning of autonomous underwater vehicle based on gbnn algorithm," *Journal of Intelligent & Robotic Systems*, vol. 94, no. 1, pp. 237–249, 2019.

[18] C. Fang and S. Anstee, "Coverage path planning for harbour seabed surveys using an autonomous underwater vehicle," in *OCEANS'10 IEEE SYDNEY*, 2010, pp. 1–8.

[19] M. Y. Rachkov, L. Marques, and A. T. de Almeida, "Multisensor demining robot," *Autonomous Robots*, vol. 18, no. 3, pp. 275–291, 2005.

[20] E. U. Acar, H. Choset, Y. Zhang, and M. Schervish, "Path planning for robotic demining: Robust sensor-based coverage of unstructured environments and probabilistic methods," *The International Journal of Robotics Research*, vol. 22, no. 7-8, pp. 441–466, 2003.

[21] E. U. Acar, Y. Zhang, H. Choset, M. Schervish, A. G. Costa, R. Melamud, D. C. Lean, and A. Graveline, "Path planning for robotic demining and development of a test platform," in *International Conference on Field and Service Robotics*, 2001, pp. 161–168.

[22] R. W. Hicks II and E. L. Hall, "Survey of robot lawn mowers," in *Intelligent Robots and Computer Vision XIX: Algorithms, Techniques, and Active Vision*, vol. 4197, 2000, pp. 262–269.

[23] I. A. Hameed, "Coverage path planning software for autonomous robotic lawn mower using dubins' curve," in *2017 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, 2017, pp. 517–522.

[24] N. S. Naik, V. V. Shete, and S. R. Danve, "Precision agriculture robot for seeding function," in *2016 International Conference on Inventive Computation Technologies (ICICT)*, vol. 2, 2016, pp. 1–3.

[25] T. H. Pham, Y. Bestaoui, and S. Mammar, "Aerial robot coverage path planning approach with concave obstacles in precision agriculture," in *2017 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*, 2017, pp. 43–48.

[26] P. Tokekar, J. Vander Hook, D. Mulla, and V. Isler, "Sensor planning for a symbiotic uav and ugv system for precision agriculture," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1498–1511, 2016.

[27] M. Farsi, K. Ratcliff, J. Johnson, C. Allen, K. Karam, and R. Pawson, "Robot control system for window cleaning," in *Proceedings of 1994 American Control Conference-ACC'94*, vol. 1, 1994, pp. 994–995.

[28] S. Hopwood, "Robots are changing the world: Should you fear them?" 2019. [Online]. Available: https://www.robotshop.com/community/blog/show/robots-are-changing-the-world-should-you-fear-them

[29] Z. L. Cao, Y. Huang, and E. L. Hall, "Region filling operations with random obstacle avoidance for mobile robots," *Journal of Robotic systems*, vol. 5, no. 2, pp. 87–102, 1988.

[30] E. M. Arkin and R. Hassin, "Approximation algorithms for the geometric covering salesman problem," *Discrete Applied Mathematics*, vol. 55, no. 3, pp. 197–218, 1994.

[31] D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook, *The traveling salesman problem*. Princeton university press, 2011.

[32] E. M. Arkin, S. P. Fekete, and J. S. Mitchell, "Approximation algorithms for lawn mowing and milling," *Computational Geometry*, vol. 17, no. 1-2, pp. 25–50, 2000.

[33] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[34] J. Reif and Z. Sun, "An efficient approximation algorithm for weighted region shortest path problem," in *Proceedings of the 4th Workshop on Algorithmic Foundations of Robotics*, 2000, pp. 191–203.

[35] M. Abrahamsen, A. Adamaszek, and T. Miltzow, "The art gallery problem is -complete," in *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, 2018, pp. 65–73.

[36] S. Carlsson, H. Jonsson, and B. J. Nilsson, "Finding the shortest watchman route in a simple polygon," *Discrete & Computational Geometry*, vol. 22, no. 3, pp. 377–402, 1999.

[37] C. Das, A. Becker, and T. Bretl, "Probably approximately correct coverage for robots with uncertainty," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 1160–1166.

[38] L. Paull, M. Seto, and H. Li, "Area coverage planning that accounts for pose uncertainty with an auv seabed surveying application," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 6592–6599.

[39] M. A. A. El-Hadidy and A. A.-A. H. El-Bagoury, "Optimal search strategy for a three-dimensional randomly located target," *International Journal of Operational Research*, vol. 29, no. 1, pp. 115–126, 2017.

[40] R. Ravichandran, D. Ghose, and K. Das, "Uav based survivor search during floods," in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2019, pp. 1407–1415.

[41] H. Guo, Z. Mao, W. Ding, and P. Liu, "Optimal search path planning for unmanned surface vehicle based on an improved genetic algorithm," *Computers & Electrical Engineering*, vol. 79, p. 106467, 2019.

[42] J. Xie, L. R. G. Carrillo, and L. Jin, "An integrated traveling salesman and coverage path planning problem for unmanned aircraft systems," *IEEE Control Systems Letters*, vol. 3, no. 1, pp. 67–72, 2018.

[43] J. Xie, L. R. G. Carrillo, and L. Jin, "Path planning for uav to cover multiple separated convex polygonal regions," *IEEE Access*, vol. 8, pp. 51 770–51 785, 2020.

[44] J. Xie and J. Chen, "Multi-regional coverage path planning for robots with energy constraint," in *2020 IEEE 16th International Conference on Control & Automation (ICCA)*, 2020, pp. 1372–1377.

[45] M. R. Irving and Y.-H. Song, "Optimisation techniques for electrical power systems. part 1: Mathematical optimisation methods," *Power Engineering Journal*, vol. 14, no. 5, pp. 245–254, 2000.

[46] J. McKeown, D. Meegan, and D. Sprevak, *An introduction to unconstrained optimisation*. CRC Press, 1990.

[47] S. Heipcke, "Comparing constraint programming and mathematical programming approaches to discrete optimisation—the change problem," *Journal of the Operational Research Society*, vol. 50, no. 6, pp. 581–595, 1999.

[48] Y. Fu and P. W. Anderson, "Application of statistical mechanics to np-complete problems in combinatorial optimisation," *Journal of Physics A: Mathematical and General*, vol. 19, no. 9, p. 1605, 1986.

[49] A. Beck, *Introduction to nonlinear optimization: Theory, algorithms, and applications with MATLAB*. SIAM, 2014.

[50] L. Meng, C. Lu, B. Zhang, Y. Ren, C. Lv, H. Sang, J. Li, and C. Zhang, "Constraint programing for solving four complex flexible shop scheduling problems," *IET Collaborative Intelligent Manufacturing*, vol. 3, no. 2, pp. 147–160, 2021.

[51] N. Biggs, E. K. Lloyd, and R. J. Wilson, *Graph Theory, 1736-1936*. Oxford University Press, 1986.

[52] C. H. Papadimitriou and K. Steiglitz, *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.

[53] G. Dantzig, *Linear programming and extensions*. Princeton university press, 2016.

[54] T. Bektaş and L. Gouveia, "Requiem for the miller–tucker–zemlin subtour elimination constraints?" *European Journal of Operational Research*, vol. 236, no. 3, pp. 820–832, 2014.

[55] G. Laporte, "The traveling salesman problem: An overview of exact and approximate algorithms," *European Journal of Operational Research*, vol. 59, no. 2, pp. 231–247, 1992.

[56] S. Poikonen, B. Golden, and E. A. Wasil, "A branch-and-bound approach to the traveling salesman problem with a drone," *INFORMS Journal on Computing*, vol. 31, no. 2, pp. 335–346, 2019.

[57] F. Duchoň, A. Babinec, M. Kajan, P. Beňo, M. Florek, T. Fico, and L. Jurišica, "Path planning with modified a star algorithm for a mobile robot," *Procedia Engineering*, vol. 96, pp. 59–69, 2014.

[58] S. Rani, Y. A. Kurnia, S. N. Huda, and S. A. S. Ekamas, "Smart travel itinerary planning application using held-karp algorithm and balanced clustering approach," in *Proceedings of the 2019 2nd International Conference on E-Business, Information Management and Computer Science*, 2019, pp. 1–5.

[59] E.-G. Talbi, *Metaheuristics: from design to implementation*. John Wiley & Sons, 2009, vol. 74.

[60] G. Kizilateş and F. Nuriyeva, "On the nearest neighbor algorithms for the traveling salesman problem," in *Advances in Computational Science, Engineering and Information Technology*. Springer, 2013, pp. 111–118.

[61] D. S. Johnson and L. A. McGeoch, "8. the traveling salesman problem: a case study," in *Local search in combinatorial optimization*. Princeton University Press, 2018, pp. 215–310.

[62] H. H. Hoos and T. Stützle, *Stochastic local search: Foundations and applications*. Elsevier, 2004.

[63] J. Yang, X. Shi, M. Marchese, and Y. Liang, "An ant colony optimization method for generalized tsp problem," *Progress in Natural Science*, vol. 18, no. 11, pp. 1417–1422, 2008.

[64] H. H. Mukhairez and A. Y. Maghari, "Performance comparison of simulated annealing, ga and aco applied to tsp," *International Journal of Intelligent Computing Research (IJICR)*, vol. 6, no. 4, 2015.

[65] Y. Deng, Y. Liu, and D. Zhou, "An improved genetic algorithm with initial population strategy for symmetric tsp," *Mathematical Problems in Engineering*, vol. 2015, 2015.

[66] R. Parween, M. V. Heredia, M. M. Rayguru, R. E. Abdulkader, and M. R. Elara, "Autonomous self-reconfigurable floor cleaning robot," *IEEE Access*, vol. 8, pp. 114 433–114 442, 2020.

[67] R. Abbaspour, "Design and implementation of multi-sensor based autonomous minesweeping robot," in *International Congress on Ultra Modern Telecommunications and Control Systems*, 2010, pp. 443–449.

[68] S. Kalaivanan and R. Kalpana, "Coverage path planning for an autonomous robot specific to agricultural operations," in *2017 International Conference on Intelligent Computing and Control (I2C2)*, 2017, pp. 1–5.

[69] L. Zacchini, A. Ridolfi, and B. Allotta, "Receding-horizon sampling-based sensor-driven coverage planning strategy for auv seabed inspections," in *2020 IEEE/OES Autonomous Underwater Vehicles Symposium (AUV)(50043)*, pp. 1–6.

[70] W. Lin, A. Anwar, Z. Li, M. Tong, J. Qiu, and H. Gao, "Recognition and pose estimation of auto parts for an autonomous spray painting robot," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 3, pp. 1709–1719, 2018.

[71] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous systems*, vol. 61, no. 12, pp. 1258–1276, 2013.

[72] T. M. Cabreira, L. B. Brisolara, and P. R. Ferreira Jr, "Survey on coverage path planning with unmanned aerial vehicles," *Drones*, vol. 3, no. 1, p. 4, 2019.

[73] R. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal on Robotics and Automation*, vol. 2, no. 1, pp. 14–23, 1986.

[74] T. Balch and R. C. Arkin, "Communication in reactive multiagent robotic systems," *Autonomous Robots*, vol. 1, no. 1, pp. 27–52, 1994.

[75] D. Mackenzie, T. Balch, *et al.*, "Making a clean sweep: Behavior based vacuuming," in *Aaai Fall Symposium, Instationating Real-World Agents*, 1996, pp. 1–6.

[76] J. Fink, V. Bauwens, F. Kaplan, and P. Dillenbourg, "Living with a vacuum cleaning robot," *International Journal of Social Robotics*, vol. 5, no. 3, pp. 389–408, 2013.

[77] R. Stuart and N. Peter, *Artificial intelligence-a modern approach 3rd ed*. Berkeley, 2016, vol. 3.

[78] L. Blasi, E. D'Amato, M. Mattei, and I. Notaro, "Path planning and real-time collision avoidance based on the essential visibility graph," *Applied Sciences*, vol. 10, no. 16, p. 5613, 2020.

[79] S. M. LaValle, J. J. Kuffner, B. Donald, *et al.*, "Rapidly-exploring random trees: Progress and prospects," *Algorithmic and Computational Robotics: New Directions*, vol. 5, pp. 293–308, 2001.

[80] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, 1989.

[81] H. Choset and P. Pignon, "Coverage path planning: The boustrophedon cellular decomposition," in *Field and Service Robotics*, 1998, pp. 203–209.

[82] V. An, Z. Qu, F. Crosby, R. Roberts, and V. An, "A triangulation-based coverage path planning," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 6, pp. 2157–2169, 2018.

[83] H. Bast and S. Hert, "The area partitioning problem," *12th Canadian Conference on Computational Geometry*, vol. 2, 2000.

[84] W. H. Huang, "The minimal sum of altitudes decomposition for coverage algorithms," *Rensselaer Polytechnic Institute Computer Science Technical Report 00-3*, 2000.

[85] P. A. Jimenez, B. Shirinzadeh, A. Nicholson, and G. Alici, "Optimal area covering using genetic algorithms," in *2007 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 2007, pp. 1–5.

[86] L. C. Santos, F. N. Santos, E. S. Pires, A. Valente, P. Costa, and S. Magalhães, "Path planning for ground robots in agriculture: A short review," in *2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 2020, pp. 61–66.

[87] A. L. Alfeo, M. G. Cimino, N. De Francesco, A. Lazzeri, M. Lega, and G. Vaglini, "Swarm coordination of mini-uavs for target search using imperfect sensors," *Intelligent Decision Technologies*, vol. 12, no. 2, pp. 149–162, 2018.

[88] Z. Khanam, B. Aslam, S. Saha, X. Zhai, S. Ehsan, R. Stolkin, and K. McDonald-Maier, "Gamma-induced image degradation analysis of robot vision sensor for autonomous inspection of nuclear sites," *IEEE Sensors Journal*, pp. 1–1, 2021.

[89] M. Laraia, *Nuclear decommissioning: Planning, execution and international experience*. Elsevier, 2012.

[90] N. Basilico and S. Carpin, "Deploying teams of heterogeneous uavs in cooperative two-level surveillance missions," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 610–615.

[91] A. Saif, K. Dimyati, K. A. Noordin, S. H. Alsamhi, and A. Hawbani, "Multi-uav and sar collaboration model for disaster management in b5g networks," *Internet Technology Letters*, p. e310, 2021.

[92] P. Lottes, R. Khanna, J. Pfeifer, R. Siegwart, and C. Stachniss, "Uav-based crop and weed classification for smart farming," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3024–3031.

[93] J.-L. Roch, "Uav classification and associated mission planning," in *Multi-Rotor Platform-based UAV Systems*. Elsevier, 2020, pp. 27–44.

[94] S. Rathinam, Z. W. Kim, and R. Sengupta, "Vision-based monitoring of locally linear structures using an unmanned aerial vehicle," *Journal of Infrastructure Systems*, vol. 14, no. 1, pp. 52–63, 2008.

[95] E. Ferrer-González, F. Agüera-Vega, F. Carvajal-Ramírez, and P. Martínez-Carricondo, "Uav photogrammetry accuracy assessment for corridor mapping based on the number and distribution of ground control points," *Remote Sensing*, vol. 12, no. 15, p. 2447, 2020.

[96] P. Fust and J. Loos, "Development perspectives for the application of autonomous, unmanned aerial systems (uass) in wildlife conservation," *Biological Conservation*, vol. 241, p. 108380, 2020.

[97] M. Niculiță, M. C. Margarint, and P. Tarolli, "Using uav and lidar data for gully geomorphic changes monitoring," in *Developments in Earth Surface Processes*. Elsevier, 2020, vol. 23, pp. 271–315.

[98] A. Barrientos, J. Colorado, J. d. Cerro, A. Martinez, C. Rossi, D. Sanz, and J. Valente, "Aerial remote sensing in agriculture: A practical approach to area coverage and path planning for fleets of mini aerial robots," *Journal of Field Robotics*, vol. 28, no. 5, pp. 667–689, 2011.

[99] J. Alvarenga, N. I. Vitzilaios, K. P. Valavanis, and M. J. Rutherford, "Survey of unmanned helicopter model-based navigation and control techniques," *Journal of Intelligent & Robotic Systems*, vol. 80, no. 1, pp. 87–138, 2015.

[100] C. Kanellakis and G. Nikolakopoulos, "Survey on computer vision for uavs: Current developments and trends," *Journal of Intelligent & Robotic Systems*, vol. 87, no. 1, pp. 141–168, 2017.

[101] A. S. Saeed, A. B. Younes, S. Islam, J. Dias, L. Seneviratne, and G. Cai, "A review on the platform design, dynamic modeling and control of hybrid uavs," in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2015, pp. 806–815.

[102] A. C. Watts, V. G. Ambrosia, and E. A. Hinkley, "Unmanned aircraft systems in remote sensing and scientific research: Classification and considerations of use," *Remote Sensing*, vol. 4, no. 6, pp. 1671–1692, 2012.

[103] M. Theile, H. Bayerlein, R. Nai, D. Gesbert, and M. Caccamo, "Uav coverage path planning under varying power constraints using deep reinforcement learning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 1444–1449.

[104] D. C. Guastella, L. Cantelli, G. Giammello, C. D. Melita, G. Spatino, and G. Muscato, "Complete coverage path planning for aerial vehicle flocks deployed in outdoor environments," *Computers & Electrical Engineering*, vol. 75, pp. 189–201, 2019.

[105] M. Oborne, "Mission planner (version 1.3.70)," 2017, [Accessed 27-August-2019]. [Online]. Available: http://planner.ardupilot.com

[106] H. L. Andersen, "Path planning for search and rescue mission using multicopters," Master's thesis, Institutt for Teknisk Kybernetikk, 2014.

[107] M. Torres, D. A. Pelta, J. L. Verdegay, and J. C. Torres, "Coverage path planning with unmanned aerial vehicles for 3d terrain reconstruction," *Expert Systems with Applications*, vol. 55, pp. 441–451, 2016.

[108] A. Bircher, M. Kamel, K. Alexis, M. Burri, P. Oettershagen, S. Omari, T. Mantel, and R. Siegwart, "Three-dimensional coverage path planning via viewpoint resampling and tour optimization for aerial robots," *Autonomous Robots*, vol. 40, no. 6, pp. 1059–1078, 2016.

[109] J. I. Vasquez-Gomez, J.-C. Herrera-Lozada, and M. Olguin-Carbajal, "Coverage path planning for surveying disjoint areas," in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2018, pp. 899–904.

[110] K. R. Jensen-Nau, T. Hermans, and K. K. Leang, "Near-optimal area-coverage path planning of energy-constrained aerial robots with application in autonomous environmental monitoring," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 3, pp. 1453–1468, 2021.

[111] K. O. Ellefsen, H. A. Lepikson, and J. C. Albiez, "Multiobjective coverage path planning: Enabling automated inspection of complex, real-world structures," *Applied Soft Computing*, vol. 61, pp. 264–282, 2017.

[112] C. Papachristos, K. Alexis, L. R. G. Carrillo, and A. Tzes, "Distributed infrastructure inspection path planning for aerial robotics subject to time constraints," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2016, pp. 406–412.

[113] J. M. Santos, T. Krajník, and T. Duckett, "Spatio-temporal exploration strategies for long-term autonomy of mobile robots," *Robotics and Autonomous Systems*, vol. 88, pp. 116–126, 2017.

[114] Z. Kashino, G. Nejat, and B. Benhabib, "A hybrid strategy for target search using static and mobile sensors," *IEEE Transactions on Cybernetics*, vol. 50, no. 2, pp. 856–868, 2018.

[115] J. W. Bae, K. Shin, H.-R. Lee, H. J. Lee, T. Lee, C. H. Kim, W.-C. Cha, G. W. Kim, and I.-C. Moon, "Evaluation of disaster response system using agent-based model with geospatial and medical details," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 9, pp. 1454–1469, 2017.

[116] X. Yu, J. Ma, N. Ding, and A. Zhang, "Cooperative target enclosing control of multiple mobile robots subject to input disturbances," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 6, pp. 3440–3449, 2021.

[117] Z. Beck, W. L. Teacy, A. Rogers, and N. R. Jennings, "Collaborative online planning for automated victim search in disaster response," *Robotics and Autonomous Systems*, vol. 100, pp. 251–266, 2018.

[118] K. Sjöö, A. Aydemir, and P. Jensfelt, "Topological spatial relations for active visual search," *Robotics and Autonomous Systems*, vol. 60, no. 9, pp. 1093–1107, 2012.

[119] P. A. Plonski, J. Vander Hook, C. Peng, N. Noori, and V. Isler, "Environment exploration in sensing automation for habitat monitoring," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 1, pp. 25–38, 2016.

[120] Y. Liu, Q. Wang, H. Hu, and Y. He, "A novel real-time moving target tracking and path planning system for a quadrotor uav in unknown unstructured outdoor scenes," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 11, pp. 2362–2372, 2018.

[121] Z. Khanam, S. Saha, B. Aslam, X. Zhai, S. Ehsan, C. Cazzaniga, C. Frost, R. Stolkin, and K. McDonald-Maier, "Degradation measurement of kinect sensor under fast neutron beamline," in *2019 IEEE Radiation Effects Data Workshop*, 2019, pp. 1–5.

[122] B. Aslam, S. Saha, Z. Khanam, X. Zhai, S. Ehsan, R. Stolkin, and K. McDonald-Maier, "Gamma-induced degradation analysis of commercial off-the-shelf camera sensors," in *2019 IEEE Sensors*, 2019, pp. 1–4.

[123] B. Yamauchi, "A frontier-based approach for autonomous exploration." in *Cira*, vol. 97, 1997, p. 146.

[124] J. A. Caley, N. R. Lawrance, and G. A. Hollinger, "Deep learning of structured environments for robot search," *Autonomous Robots*, vol. 43, no. 7, pp. 1695–1714, 2019.

[125] H. Lau, S. Huang, and G. Dissanayake, "Optimal search for multiple targets in a built environment," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005, pp. 3740–3745.

[126] T. H. Chung, "On probabilistic search decisions under searcher motion constraints," in *Algorithmic Foundation of Robotics VIII*. Springer, 2009, pp. 501–516.

[127] J. Berger and N. Lo, "An innovative multi-agent search-and-rescue path planning approach," *Computers & Operations Research*, vol. 53, pp. 24–31, 2015.

[128] M. Farahat, "Analytical architectural study on nuclear power plants," *Journal of Environmental Science and Engineering B*, p. 189, 2016.

[129] S. Ahmed, M. Weber, M. Liwicki, C. Langenhan, A. Dengel, and F. Petzold, "Automatic analysis and sketch-based retrieval of architectural floor plans," *Pattern Recognition Letters*, vol. 35, pp. 91–100, 2014.

[130] S. Dodge, J. Xu, and B. Stenger, "Parsing floor plan images," in *2017 Fifteenth IAPR International Conference on Machine Vision Applications (MVA)*, 2017, pp. 358–361.

[131] Z. Zeng, X. Li, Y. K. Yu, and C.-W. Fu, "Deep floor plan recognition using a multi-task network with room-boundary-guided attention," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 9096–9104.

[132] M. Luperto, V. Arcerito, and F. Amigoni, "Predicting the layout of partially observed rooms from grid maps," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6898–6904.

[133] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 6, pp. 679–698, 1986.

[134] N. Kiryati, Y. Eldar, and A. M. Bruckstein, "A probabilistic hough transform," *Pattern Recognition*, vol. 24, no. 4, pp. 303–316, 1991.

[135] R. Bormann, F. Jordan, W. Li, J. Hampp, and M. Hägele, "Room segmentation: Survey, implementation, and analysis," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1019–1026.

[136] G. Grisetti, C. Stachniss, W. Burgard, *et al.*, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, p. 34, 2007.

[137] M. Luperto, D. Fusi, N. A. Borghese, and F. Amigoni, "Robot exploration using knowledge of inaccurate floor plans," in *2019 European Conference on Mobile Robots (ECMR)*, 2019, pp. 1–7.

[138] F. Boniardi, B. Behzadian, W. Burgard, and G. D. Tipaldi, "Robot navigation in hand-drawn sketched maps," in *2015 European conference on mobile robots (ECMR)*, 2015, pp. 1–6.

[139] B. Bird, A. Griffiths, H. Martin, E. Codres, J. Jones, A. Stancu, B. Lennox, S. Watson, and X. Poteau, "Radiological monitoring of nuclear facilities: Using the continuous autonomous radiation monitoring assistance robot," *IEEE Robotics & Automation Magazine*, vol. 26, no. 1, pp. 35–43, 2019.

[140] C. West, F. Arvin, W. Cheah, A. West, S. Watson, M. Giuliani, and B. Lennox, "A debris clearance robot for extreme environments," in *Annual Conference Towards Autonomous Robotic Systems*, 2019, pp. 148–159.

[141] W. Harrison, A. Downs, and C. Schlenoff, "The agile robotics for industrial automation competition," *AI Magazine*, vol. 39, no. 4, p. 77, 2018.

[142] J. R. Solberg, K. M. Lynch, and M. A. Maciver, "Active electrolocation for underwater target localization," *The International Journal of Robotics Research*, vol. 27, no. 5, pp. 529–548, May 2008.

[143] N. Ganganath, C.-T. Cheng, and K. T. Chi, "Rapidly replanning a," in *2016 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, 2016, pp. 386–389.

[144] K. Jeddisaravi, R. J. Alitappeh, L. C. Pimenta, and F. G. Guimarães, "Multi-objective approach for robot motion planning in search tasks," *Applied Intelligence*, vol. 45, no. 2, pp. 305–321, 2016.

[145] Y. Choi, Y. Choi, S. Briceno, and D. N. Mavris, "Multi-uas path-planning for a large-scale disjoint disaster management," in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2019, pp. 799–807.

[146] C. Gehring, P. Fankhauser, L. Isler, R. Diethelm, S. Bachmann, M. Potz, L. Gerstenberg, and M. Hutter, "Anymal in the field: Solving industrial inspection of an offshore hvdc platform with a quadrupedal robot," in *12th Conference on Field and Service Robotics (FSR 2019)*, 2019.

[147] N. Pinkam, A. A. R. Newaz, S. Jeong, and N. Y. Chong, "Rapid coverage of regions of interest for environmental monitoring," *Intelligent Service Robotics*, vol. 12, no. 4, pp. 393–406, 2019.

[148] C. Pan, M. Zhou, Y. Qiao, and N. Wu, "Scheduling cluster tools in semiconductor manufacturing: Recent advances and challenges," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 2, pp. 586–601, 2017.

[149] B. Bird, A. Griffiths, H. Martin, E. Codres, J. Jones, A. Stancu, B. Lennox, S. Watson, and X. Poteau, "A robot to monitor nuclear facilities: Using autonomous radiation-monitoring assistance to reduce risk and cost," *IEEE Robotics & Automation Magazine*, vol. 26, no. 1, pp. 35–43, 2018.

[150] S. Papaioannou, P. Kolios, T. Theocharides, C. G. Panayiotou, and M. Polycarpou, "Jointly-optimized searching and tracking with random finite sets," *IEEE Transactions on Mobile Computing*, vol. 19, no. 10, pp. 2374–2391, 2019.

[151] Y. Yuan, D. Cattaruzza, M. Ogier, and F. Semet, "A note on the lifted miller–tucker–zemlin subtour elimination constraints for routing problems with time windows," *Operations Research Letters*, vol. 48, no. 2, pp. 167–169, 2020.

[152] A. Ellenberg, L. Branco, A. Krick, I. Bartoli, and A. Kontsos, "Use of unmanned aerial vehicle for quantitative infrastructure evaluation," *Journal of Infrastructure Systems*, vol. 21, no. 3, p. 04014054, 2015.

[153] S. Lange, N. Sünderhauf, P. Neubert, S. Drews, and P. Protzel, "Autonomous corridor flight of a uav using a low-cost and light-weight rgb-d camera," in *Advances in Autonomous Mini Robots*. Springer, 2012, pp. 183–192.

[154] C. Cazzaniga and C. D. Frost, "Progress of the scientific commissioning of a fast neutron beamline for chip irradiation," in *Journal of Physics: Conference Series*, vol. 1021, no. 1, 2018, p. 012037.

[155] T. Mallick *et al.*, "Characterizations of noise in kinect depth images: A review," *IEEE Sensors journal*, vol. 14, no. 6, pp. 1731–1740, 2014.

[156] J. Srour and J. Palko, "Displacement damage effects in irradiated semiconductor devices," *IEEE Transactions on Nuclear Science*, vol. 60, no. 3, pp. 1740–1766, 2013.

[157] Y. Nobert and J.-C. Picard, "An optimal algorithm for the mixed chinese postman problem," *Networks: An International Journal*, vol. 27, no. 2, pp. 95–108, 1996.

[158] A. Soares, R. Râbelo, and A. Delbem, "Optimization based on phylogram analysis," *Expert Systems with Applications*, vol. 78, pp. 32–50, 2017.

[159] J. He, Y. Geng, F. Liu, and C. Xu, "Cc-kf: Enhanced toa performance in multipath and nlos indoor extreme environment," *IEEE Sensors Journal*, vol. 14, no. 11, pp. 3766–3774, 2014.

[160] C. Bliek1ú, P. Bonami, and A. Lodi, "Solving mixed-integer quadratic programming problems with ibm-cplex: a progress report," in *Proceedings of the Twenty-Sixth RAMP Symposium*, 2014, pp. 16–17.

[161] Z.-W. Lin, S.-Y. Fang, Y.-W. Chang, W.-C. Rao, and C.-H. Kuan, "Provably good max–min-*m*-neighbor-tsp-based subfield scheduling for electron-beam photomask fabrication," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 2, pp. 378–391, 2017.

[162] "Regular polygon-matlab," 2020. [Online]. Available: https://uk.mathworks.com/help/matlab/ref/nsidedpoly.html

[163] C. Moon, J. Kim, G. Choi, and Y. Seo, "An efficient genetic algorithm for the traveling salesman problem with precedence constraints," *European Journal of Operational Research*, vol. 140, no. 3, pp. 606–617, 2002.

[164] J. Kirk, "Traveling salesman problem-genetic algorithm," *Retrieved from the MATLAB File Exchange website: www. mathworks. com/matlabcentral/fileexchange/13680-travelingsalesman-problem-genetic-algorithm*, 2007.

[165] O. Sundstrom and L. Guzzella, "A generic dynamic programming matlab function," in *2009 IEEE Control Applications,(CCA) & Intelligent Control,(ISIC)*, 2009, pp. 1625–1630.

[166] L. Ruan, J. Wang, J. Chen, Y. Xu, Y. Yang, H. Jiang, Y. Zhang, and Y. Xu, "Energy-efficient multi-uav coverage deployment in uav networks: A game-theoretic framework," *China Communications*, vol. 15, no. 10, pp. 194–209, 2018.

[167] C. H. Liu, Z. Chen, J. Tang, J. Xu, and C. Piao, "Energy-efficient uav control for effective and fair communication coverage: A deep reinforcement learning approach," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 9, pp. 2059–2070, 2018.

[168] C. Di Franco and G. Buttazzo, "Energy-aware coverage path planning of uavs," in *2015 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 2015, pp. 111–117.

[169] C. Di Franco and G. Buttazzo, "Coverage path planning for uavs photogrammetry with energy and resolution constraints," *Journal of Intelligent & Robotic Systems*, vol. 83, no. 3-4, pp. 445–462, 2016.

[170] Y. Bouzid, Y. Bestaoui, and H. Siguerdidjane, "Quadrotor-uav optimal coverage path planning in cluttered environment with a limited onboard energy," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 979–984.

[171] D. Ognibene, L. Mirante, and L. Marchegiani, "Proactive intention recognition for joint human-robot search and rescue missions through monte-carlo planning in pomdp environments," in *International Conference on Social Robotics*. Springer, 2019, pp. 332–343.

[172] J. Sandino, F. Maire, P. Caccetta, C. Sanderson, and F. Gonzalez, "Drone-based autonomous motion planning system for outdoor environments under object detection uncertainty," *Remote Sensing*, vol. 13, no. 21, p. 4481, 2021.

[173] Y. Wang, S. Chaudhuri, and L. E. Kavraki, "Online partial conditional plan synthesis for pomdps with safe-reachability objectives," in *International Workshop on the Algorithmic Foundations of Robotics*. Springer, 2018, pp. 127–143.

[174] D.-I. Kim, Y.-S. Song, G. Kim, and C.-W. Kim, "A study on the application of uav for korean land monitoring," *Journal of the Korean Society of Surveying, Geodesy, Photogrammetry and Cartography*, vol. 32, no. 1, pp. 29–38, 2014.

[175] H. Choset and P. Pignon, "Coverage path planning: The boustrophedon cellular decomposition," in *Field and Service Robotics*, 1998, pp. 203–209.

[176] J. Colegrave and A. Branch, "A case study of autonomous household vacuum cleaner," *AIAA/NASA CIRFFSS*, vol. 107, 1994.

[177] I. A. Hameed, D. Bochtis, and C. A. Sørensen, "An optimized field coverage planning approach for navigation of agricultural robots in fields involving obstacle areas," *International Journal of Advanced Robotic Systems*, vol. 10, no. 5, p. 231, 2013.

[178] A. Villa-Henriksen, G. T. Edwards, L. A. Pesonen, O. Green, and C. A. G. Sørensen, "Internet of things in arable farming: Implementation, applications, challenges and potential," *Biosystems Engineering*, vol. 191, pp. 60–84, 2020.

[179] K. Zhou, A. L. Jensen, C. Sørensen, P. Busato, and D. Bothtis, "Agricultural operations planning in fields with multiple obstacle areas," *Computers and Electronics in Agriculture*, vol. 109, pp. 12–22, 2014.

[180] Z. Khanam, S. Saha, S. Ehsan, R. Stolkin, and K. McDonald-Maier, "Coverage path planning techniques for inspection of disjoint regions with precedence provision," *IEEE Access*, vol. 9, pp. 5412–5427, 2020.

[181] S. S. Mansouri, C. Kanellakis, E. Fresk, D. Kominiak, and G. Nikolakopoulos, "Cooperative coverage path planning for visual inspection," *Control Engineering Practice*, vol. 74, pp. 118–131, 2018.

[182] I. A. Hameed, "Intelligent coverage path planning for agricultural robots and autonomous machines on three-dimensional terrain," *Journal of Intelligent & Robotic Systems*, vol. 74, no. 3, pp. 965–983, 2014.

[183] S. Dogru and L. Marques, "Improved coverage path planning using a virtual sensor footprint: a case study on demining," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 4410–4415.

[184] B. Casse, "Council post: The demise of robotics companies: Learning from past mistakes," 2021. [Online]. Available: https://www.forbes.com/sites/forbesbusinesscouncil/2021/07/09/the-demise-of-robotics-companies-learning-from-past-mistakes/?sh=1062b8382b1d