Amin Majd

DIANA: Distributed and Safe Autonomous Navigation for a Swarm of Autonomous Vehicles

Åbo Akademi University

Amin Majd

Born 1981

Previous studies and degrees

BSc. in Computer Sciences, Shahid Bahonar University 2004

MSc. in Computer Sciences, University of Tabriz 2013

DSc. in Information Technology, University of Turku 2019

Cover Art: Amin Majd

DIANA: Distributed and Safe Autonomous Navigation for a Swarm of Autonomous Vehicles

Amin Majd

To be presented, with the permission of the Faculty of Science and Engineering of the Åbo Akademi University, for public criticism in Lilla Auditorium in ASA on May 7, 2021, at 13:00.

Åbo Akademi University
Department of Infromation Technology

2021

Supervisors

Professor Johan Lilius
Department of Information Technology
Åbo Akademi University
Agora, Domkyrkotorget 3, 20500, Turku
Finland

Professor Masoud Daneshtalab
Department of Embedded Systems, division of Intelligent Future Technology
Mälardalen University
Högskoleplan 1, 722 20 Västerås
Sweden

Reviewers

Professor Linas Laibinis
Department of Computer Sciences
Vilnius University
Lithuania

Professor Jüri Vain
Department of Software Science
Tallinn University of Technology
Estonia

Opponent

Professor Jüri Vain
Department of Software Science
Tallinn University of Technology
Estonia

ISBN 978-952-12-4057-7 (printed)
ISBN 978-952-12-4058-4 (digital)
Painosalama, Turku, Finland 2021

برآ، ای آفتاب، ای توشه‌ی امید!

برآ، ای خوشه‌ی خورشید!

تو جوشان چشمه‌ای، من تشنه‌ای بیتاب

برآ، سرریز کن، تا جان شود سیراب

س. کسرایی

Abstract

The autonomous systems are typical examples of complex distributed cyber-physical systems (CPS). The main characteristics of such systems is the use of autonomous vehicles. They are increasingly used in various mission-critical tasks such as surveillance and rescue operations. To execute the required tasks, the autonomous swarm systems should fulfil such important dependability requirements as safety and reliability. On the hand, while designing a swarm system, we should guarantee that the robotic systems in the swarm do not collide with each other and objects in the operating environment, i.e., ensure motion safety. On the other hand, to ensure that the robots have sufficient resources to reliably complete the required goals, we should also achieve efficiency while implementing the swarm mission, i.e., minimise the travelling distance of the robots vehicles. In this thesis, we propose a novel integrated approach that ensures motion safety and efficiency while planning and controlling an operation of swarms of autonomous robots. We validate our approach in different case studies and compare them with some state-of-the art benchmarks. Moreover, we rely on formal modelling to derive the safety constraints guaranteeing that the swarm system can cope with both predicted and dynamically emerging safety hazards. We define an architecture of the controlling software that combines static and dynamic mechanisms for safe and efficient swarm control and navigation. To ensure efficiency, while preserving safety, we propose a new parallel algorithm for swarm mission planning. This algorithm is a combination of evolutionary computing methods, machine learning and deterministic approaches that coordinated by a central management component. The algorithm controls the swarm actions on three different layers: the offline, online and vehicle layer as well as allows us to plan and optimise at the run-time the routes of the vehicles to maximise safety while minimising the travelling distance. Our solution promotes a holistic approach to designing CPS – from a formal requirements definition to a software implementation that fulfils the defined requirements in an efficient way. The results of benchmarking demonstrate that our approach allows safe and efficient control of swarms.

Sammandrag

Autonoma system är typiska exempel på komplexa distribuerade cyber-fysiska system (CPS). De viktigaste användningsområdet av sådana system är autonoma fordon. Autonoma fordon används alltmer i olika uppdragskritiska uppgifter som övervaknings- och räddningsinsatser. För att utföra de uppgifter som krävs bör de autonoma svärmsystem uppfylla viktiga pålitlighetskrav som säkerhet och tillförlitlighet. Samtidigt, när vi utformar ett svärmsystem, bör vi garantera att robotsystemen i svärmen inte kolliderar med varandra och föremål i driftsmiljön, dvs säkerställa rörelsesäkerhet. För att säkerställa att robotarna har tillräckliga resurser för att på ett tillförlitligt sätt fullgöra de nödvändiga målen, bör vi också vara effektiva när vi genomför svärmuppdraget, dvs. minimera robotfordonets färdavstånd. I denna avhandling föreslår vi ett nytt integrerat tillvägagångssätt som säkerställer manöveringssäkerhet och effektivitet samtidigt som man planerar och kontrollerar en operation av svärmar av autonoma robotar. Vi validerar vår strategi i olika fallstudier och jämför dem med några toppmoderna implementationer. Vi använder också formell modellering för att härleda säkerhetsbegränsningarna som garanterar att svärmsystemet klarar av både förutsagda och dynamiskt framkommande säkerhetsrisker. Vi definierar en arkitektur för den styrande programvaran som kombinerar statiska och dynamiska mekanismer för säker och effektiv svärmkontroll och navigering. För att säkerställa effektivitet, samtidigt som säkerheten bevaras, föreslår vi en ny parallell algoritm för planering av svärmuppdrag. Denna algoritm är en kombination av evolutionära datormetoder, maskininlärning och deterministiska tillvägagångssätt som samordnas av en central styrkomponent. Algoritmen kontrollerar svärmåtgärderna i tre olika lager: ett offline-, ett online- och ett fordonslager som låter oss planera och optimera fordonens rutter för att maximera säkerheten och samtidigt minimera körsträckan. Vår lösning främjar en helhetssyn på utformningen av CPS - från en formell kravdefinition till en programvaruimplementation som uppfyller de definierade kraven på ett effektivt sätt. Resultaten av benchmarkingen visar att vårt tillvägagångssätt möjliggör säker och effektiv kontroll av svärmar.

Acknowledgements

This study was carried out during the years 2017-2021 in the Department of Information Technology at Åbo Akademi University. I would like to express my special appreciation and thanks to my supervisors: Professor Masoud Daneshtalab, Professor Johan Lilius that without their supervision, and scientific support this doctoral thesis would not have been possible. I want to especially thank Professor Masoud Daneshtalab for his comprehensive support when I entered the Åbo Akademi University and during my studies. I would also like to thank Associate Professor Elena Troubitsyna for providing the opportunity to pursue my doctoral studies at Åbo Akademi University. I am thankful for the Academy of Finland Foundation,for financial support. All of these support have encouraged me to continue my work, and to aspire to inspire.

I would like to thank my colleagues and co-authors: Associate Professor Elena Troubitsyna, Dr Adnan Ashraf, Dr Inna Pereverzeva, Golnaz Sahebi, Mohammad Loni, and Mohammad Riazati. . Special thanks to administrative and technical staff, especially Christel Engbolm, Minna Carla, Pia-Maria Kallio, and Karl Rönnholm who never wavered their support. I am also grateful to Dennis Biström for translating the abstract of this thesis to Swedish.

I would also like to thank Professor Linas Laibinis and Professor Jüri Vainn for accepting to be the reviewer of my thesis. I also thank Professor Jüri Vain for accepting to be the opponent of my thesis. It would have been a farfetched dream to reach this point if it was not for my family's endless love and encouragement. Thank you for believing in me before I believed in myself. I would like to especially thank my mother, Sorour, my father, Hossein, and my brother, Jahanshah, for all the supports that they provided during my life. Also, I would like to thank my daughter, Diana, because I have spent some of her time to complete my studies. She spent some of her free time in my office, and she inspired the title of this dissertation. Finally, I would like to express my deepest appreciation to my brilliant wife, Golnaz Sahebi, that without her support and inspiration this journey would not have been started. She was my best friend, colleague, and supporter over the last 16 years. You have been extremely supportive of me throughout this entire process and have made countless sacrifices to help me get to this point. This accomplishment would not have been possible without you. Thanks my heartfelt

List of original publications

1. A. Majd, M. Daneshtalab and E. Troubitsyna, “Optimal Placement for Smart Mobile Access Points”, The 18th IEEE International Conference on Scalable Computing and Communications (ScalCom 2018), 2018.

2. A. Ashraf, A. Majd and E. Troubitsyna, “Towards A Realtime, Collision-Free Motion Coordination and Navigation System for a UAV Fleet”, 5th European Conference on the Engineering of Computer Based Systems, (ECBS 17), 2017.

3. A. Ashraf, A. Majd, and E. Troubitsyna, “Online Path Generation and Navigation for Swarms of UAVs”, Journal of Scientific Programming, 2020.

4. A. Majd, E. Troubitsyna, and M. Daneshtalab, “Safety-Aware Control of Swarms of Drones”, 12th ERCIM/EWICS/ARTEMIS Workshop on Dependable Smart Embedded Cyber-physical Systems and Systems-of-Systems at SAFECOMP 2017, (DECSoS 17), 2017.

5. A. Majd, E. Troubitsyna and M. Daneshtalab, “Ensuring fault tolerance and efficiency in autonomous swarm-based monitoring systems”, The 11th IFIP International Conference on Research and Practical Issues of Enterprise Information Systems (Confenis 2017), 2017.

6. A. Majd, M. Loni, G. Sahebi, and M. Daneshtalab, “Improving Motion Safety and Efficiency of Intelligent Autonomous Swarm of Drones”, Journal of Drones, 2020.

7. A. Majd, A. Ashraf, E. Troubitsyna and M. Daneshtalab, “Integrating Learning, Optimization, and Prediction for Efficient Navigation of Swarms of Drones”, 26th Euromicro International Conferences on Parallel, Distributed and Network-Based Processing, (PDP18), 2018.

8. A. Majd, A. Ashraf, E. Troubitsyna and M. Daneshtalab, “Using Optimization, Learning, and Drone Reflexes to Maximize Safety of Swarms of Drones”, IEEE Congress on Evolutionary Computation (IEEE CEC), 2018.

9. A. Majd, A. Ashraf, E. Troubitsyna, and M. Daneshtalab, “A Framework for Intelligent Navigation of Swarms of Drones”, IEEE Transactions on Intelligent Transportation Systems, 2021.

10. A, Majd, M. Riazati, M. Daneshtalab, “Cost-Efficient Coverage of Mobile Access Point Placements in Dynamic Cyber Physical Environments”, IEEE ACCESS, 2021.

11. I. Vistbakka, A. Majd and E. Troubitsyna, “Deriving Mode Logic for Autonomous Resilient Systems”, The 20th International Conference on Formal Engineering Methods (ICFEM 2018), 2018.

12. I. Vistbakka, A. Majd and E. Troubitsyna, “Multi-Layered Approach to Safe Navigation of Swarms of Drone”, 13th ERCIM/EWICS/ARTEMIS Workshop on “Dependable Smart Cyber-Physical Systems and Systems-of-Systems” at SAFECOMP 2018 (DECSoS ’18), 2018.

13. I. Vistbakka, E.Troubitsuna, and A. Majd, “Multi-Layered Safety Architecture of Autonomous Systems: Formalising Coordination Perspective”, 19th IEEE International Symposium on High Assurance Systems Engineering, 2019.

List of publications relevant but not included

1. A. Majd, L. Espinosa, and M. Westerlund, “Multi-View Object Detection Network for Autonomous Ships by Combining Ship and Drone Views”, The FinDrones’20, 2020.

2. M. Loni, A. Majd, A. Zoljoodi, and M. Daneshtalab, “DenseDisp: Resource-Aware Disparity Map Estimation by Compressing Siamese Neural Architecture”, The IEEE Congress on Evolutionary Computation (CEC’20). 2020

3. M. Salimi, A. Majd, M. Loni, T. Seceleanu, C. Seceleanu, M. Sirjani, M. Daneshtalab, and E. Troubitsyna, “Multi-objective Optimization of Real-Time Task Scheduling Problem for Distributed Environments”, 6th Conference on the Engineering of Computer Based Systems, (ECBS’19), 2019.

4. M. Loni, A. Majd, M. Daneshtalab, M. Sjodin and E. Troubitsyna, “Designing Compact Convolutional Neural Network for Embedded Stereo Vision Systems”, The 12th IEEE International Symposium on Embedded Multi-core/Many-core Systems-on-Chip (MCSoC-2018), 2018.

5. A. Majd, G. Sahebi, M. Daneshtalab and E. Troubitsyna, “Optimal Placement for Smart Mobile Access Points”, The 18th IEEE International Conference on Scalable Computing and Communications (ScalCom 2018), 2018.

6. A. Majd, G. Sahebi, M. Daneshtalab, J. Plosila, Sh. Lotfi, H. Tenhunen, “Parallel Imperialist Competitive Algorithms”, Concurrency and Computation Practice and Experience, WILEY, 2018.

7. A. Majd, G. Sahebi, M. Daneshtalab, J. Plosila and H. Tenhunen, “Hierarchal Placement of Smart Mobile Access Points in Wireless Sensor Networks using Fog Computing”, The 25th Euromicro International Conferences on Parallel, Distributed and Network-Based Processing, (PDP’17), 2017.

8. A. Majd, G. Sahebi, M. Daneshtalab, and E. Troubitsyna, “Optimizing Scheduling for Heterogeneous Computing Systems using Combinatorial

Meta-heuristic Solution", The 17th IEEE International Conference on Scalable Computing and Communications (Smart World'17), 2017.

9. A. Majd, M. Daneshtalab, E. Troubitsyna and G. Sahebi, "Optimal Smart Mobile Access Point Placement for Maximal Coverage and Minimal Communication", 5th European Conference on the Engineering of Computer Based Systems, (ECBS' 17), 2017.

10. A. Majd and E. Troubitsyna, "Integrating Safety-Aware Route Optimisation and Run-Time Safety Monitoring in Controlling Swarms of Drones", The 28th International Symposium on Software Reliability Engineering (ISSRE'17), 2017.

11. A. Majd and E. Troubitsyna, "Data-Driven Approach to Ensuring Fault Tolerance and Efficiency of Swarm Systems", IEEE International Conference on Big Data (Big Data'17), 2017.

12. A. Majd, M. Daneshtalab, J. Plosila, N. Khalilzad, G. Sahebi, and E. Troubitsyna, "NOMeS: Near-Optimal Metaheuristic Scheduling for MPSoCs", 19th International Symposium on Computer Architecture and Digital Systems (CADS'17), 2017.

13. A. Majd, Sh. Lofti, G. Sahebi, M. Daneshtalab and J. Plosila, "PICA: Multi-Population Implementation of Parallel Imperialist Competitive Algorithms", The 24th Euromicro International Conferences on Parallel, Distributed and Network-Based Processing, (PDP'16), 2016.

14. A. Majd, M. Abdollahi, G. Sahebi, D. Abdollahi, M. Daneshtalab, J. Plosila and H.Tenhunen, "Parallel Imperialist Competitive Algorithm Based on Multi-Population Technique for Solving Systems of Nonlinear Equation", The 2016 International Conference on High Performance Computing & Simulation (HPCS'16), 2016.

15. A. Majd, G. Sahebi, M. Daneshtalab, J. Plosila and H. Tenhunen, "Placement of Smart Mobile Access Points in Wireless Sensor Networks and Cyber-Physical Systems using Fog Computing", The 16th IEEE International Conference on Scalable Computing and Communications (Smart World'16), 2016.

Contents

Symbols and Abbreviations

AVs	Autonomous Vehicles
NP	Non-deterministic Polynomial
EA	Evolutionary Algorithms
ML	Machine Learning
DIANA	Distributed Autonomous Navigation
ICA	Imperialist Competitive Algorithm
CPS	Cyber Physical System
MAPE	Monitor-Adapt-Plan-Execute
Req	Requirements
AI	Artificial Intelligence
SVM	support vector machine
kNN	k-Nearest Neighbors
CNN	Convolutional Neural Network
PSO	Particle Swarm Optimization
GGA	Greedy Heuristics and Genetic Algorithms
2D	Two Dimensional
UAV	Unmanned Aerial Vehicle (
DEA	Dynamic Evolutionary Algorithm
MICAP	Multi-population ICA
DNN	Deep Neural Network
RGB	Red Green Blue Color Model
SMA	Simple Moving Average
EMA	Eexponential Moving Average
FD	Force Direction
FOb	Force of the Obstacle
SD	Safe Direction
DANA	Dynamic Autonomous Navigation Algorithm
DGA	Dynamic Genetic Algorithm
RRT	Rapidly-Exploring Random Trees
TRL	Total Route Length
MD	Minimum Distance
FRR	Frequency of Route Regeneration
NC	Number of Crashes

LLR	Length of the Longest Route
TT	Total Time
# NCW	Number of Changing Weight
#ND	Number of Drones
#NCD	Number of Changing Drones
#CP	The Number of Covered point
TCW	Total Covered Weights
APS	Average Processing Speed
MPT	Minimum Processing Time
WPT	Worst Processing Time

Chapter 1

Introduction

1.1 Autonomous Vehicles

Autonomous Vehicles (AVs) are increasingly integrated into our everyday life. AVs of all kinds, variety of servo-robots, industrial and medical robots perform a wide spectrum of tasks alongside humans. This raises serious concerns regarding the risks associated with blending the autonomous technologies into the safety-critical activities as well as dependability of the resulting socio-technical systems.

Autonomous swarm (robotic) systems are increasingly used in a variety of domains such as surveillance, observation, and monitoring of rescue areas, etc. Moreover, it is envisaged that autonomous robotic systems will be able to perform complex shipping and transportation operations. To ensure that a swarm can correctly and reliably perform a required mission, we should develop efficient and highly-performant navigation algorithms.

Autonomous robots, in particular, swarms of drones are increasingly used in a variety of applications such as surveillance, good delivery, rescue operations etc. Strong business incentives drive fast development and deployment of the drone technology. However, since the number of incidents involving drones is increasing, it raises a serious concern regarding the safety implications of the employed technology.

For instance, in November 2016, it was reported that "Passenger plane approaching Heathrow (was) in near-miss with drone 650ft to the east of the Shard" [1]. Therefore, ensuring dependability in controlling autonomous systems, such as swarms of drones, constitutes an important engineering problem. The autonomous swarms of drones are typical examples of complex distributed cyber-physical systems (CPS). To execute the required tasks, the swarm systems should fulfil such important dependability requirements as safety and reliability. On the hand, while designing a swarm system, we should guarantee that the drones in the swarm do not collide with each other and objects in the operating environment, i.e., ensure motion safety. On the other hand, the drones are resource-constrained systems,

and hence, we should ensure that the drones have sufficient resources to reliably complete the required task, i.e., achieve high efficiency while implementing the mission. This introduces an additional complexity vector into the dependability assurance. Therefore, the problem of ensuring dependability of swarm systems calls for the integrated approaches that allow the designers to match the system-level dependability requirements with their efficient implementations.

The navigation concept is divided into three main categories: placement, path planning and control instructions to AVs. In this thesis we have focused on placement and path planning. The placement of AVs defines the best configurations of AVs at a different time in the dynamic environments. The goal of placement is to make the best configuration to maximise the total benefit of the swarm. It works as a dynamic optimisation problem. The path planning tries to improve the safety and efficiency of AV's routes. Usually, the shortest path is selected by common path planning algorithms. For real-world application, computing the shortest path is not sufficient, and we need to optimise routes to achieve the maximum level of safety and efficiency at the same time.

The placement problem is an NP-hard problem, and when it is even more difficult for dynamic swarm systems needs more computation resources and time to define the best configurations. Using common evolutionary algorithms (EA) and machine learning (ML) approaches are not applicable for the placement problem in dynamic swarm systems with lots of unpredictable obstacles. EAs work efficiently for optimisation problems. However, they need more time and computational resources to generate an acceptable solution in real-time, or moreover, they sometimes converge to local optimum results. As the navigation and placement approaches need to make the best decision in the shortest possible time, they need to be customised and more light-weight.

On the other hand, ML methods are successfully and widely used in the different research areas. A ML method needs to be trained (to make a model), evaluated and tested. Also, they need a complete data set to be trained correctly. In dynamic environments that we would need to rebuild and retrain the algorithm for all dynamically changing coordination whenever it happens. ML needs to be combined with other methods to be capable to address navigation and placement problems.

1.2 Distributed Autonomous Navigation (DIANA)

Distributed Autonomous Navigation (DIANA) as a general and dynamic framework is proposed in this thesis to address the above mentioned problems.

In this thesis, I tried to work on the development of high-performance algorithms for controlling the swarm of AVs. The framework proposes algorithms for ensuring safe, reliable and efficient control of swarm systems. The theoretical basis consists of ML, EA, some deterministic approaches, and formal modelling. As a result of my research, I have proposed the algorithm that combines EA, supervised and

unsupervised machine learning to allow swarm systems to maintain safe, reliable and efficient operation despite the changes in the internal and external operating conditions. Unsupervised learning is used to identify an unforeseen safety risk, while supervised learning is used to develop risk mitigation strategies as well as reconfigure the swarm to maintain reliability and increase efficiency. Finally, the evolutionary computing provides us with a basis for creating high-performance algorithms that can optimise the safety/reliability and efficiency ratio.

To guarantee safety and reliability, we needed to verify the proposed algorithms. We have also developed formal models and applied advanced verification techniques to ensure the correctness of the proposed algorithms. The DIANA framework can be implemented on a heterogeneous computing platform. Such a platform has a layered architecture, i.e., allow us to perform some computations directly in AVs (for example drones), some computation at the fog layer and finally, offload the most computationally intensive tasks to the cloud.

Finally, we have validated the framework by several case studies illustrating a safe and efficient control of swarms of drones and monitoring missions using a swarm of AVs.

In this thesis, we propose a novel integrated approach to ensuring dependability of swarms of drones. Our approach combines formal modelling and evolutionary computing to derive the architecture and efficient implementation of software that controls the behaviour a swarm of drones and guarantee system dependability.

To ensure efficiency, while preserving safety, we propose a new parallel algorithm for safety-aware swarm mission planning. The algorithm builds on the idea of the evolutionary computing [2]. We consider the route planning as an optimization problem that aims at maximizing safety while minimizing the length of the path of each drone. As a basis of our solution, we take the Imperialist Competitive Algorithm (ICA) [3]. By mimicking the processes associated with a competition of imperialistic countries to acquire colonies, the algorithm iteratively generates the solutions that progressively maximize (or minimize) the value of the defined fitness function. In our definition of the fitness function, we explicitly introduce safety as an argument, i.e., ensure that our route planning finds the safest shortest route for each drone. The proposed solution allows the system to proactively recalculate the routes of the drones to ensure that the swarm continues its mission execution in a safe and efficient manner.

Our solution promotes a holistic approach to designing a particular class of CPSs – from formal requirements definition to software implementation that fulfils the defined requirements in an efficient way. The results of benchmarking demonstrate that our approach guarantees safe and efficient control of swarms.

The main research questions addressed by the thesis:

1. How to combine EA, ML, and coordination algorithms to achieve a safe and efficient real-time implementation of AVs?

2. How to formally verify safety of the proposed architecture?

Chapter 2

Background of the Modalities Used in This Work

2.1 Dependability of Swarms of Drones

The swarms of drones are increasingly used for surveillance, shipping, rescue, etc. A swarm is a group of autonomously functioning drones that in a coordinated manner provide the required services, i.e., execute a number of missions. The swarms of drones are the typical examples of complex distributed CPS (or even systems of CPS). Each drone has an individual goal that contributes to achieving an overall system-level goal of the entire swarm. Usually it is assumed that the system-level goal remains unchanged for the entire mission. The system-level goal can be defined in terms of the individual goals that each drone should achieve. However, the behaviour of the swarm is highly dynamic, i.e., the tactics of achieving the required goal continuously change depending on the situations emerging during the mission execution.

Situation-awareness – an ability of the swarm system to assess its own state and the state of its operating environment – is implemented by diverse set of means. Firstly, each drone can communicate with its peers and the coordinator. In our work, we assume that the communication is reliably maintained during the entire mission. Secondly, each drone is capable of detecting and communicating its own telemetry data: coordinates, speed, energy level, etc. Moreover, each drone has some capability to monitor its environment, i.e., it might be equipped with sensors, a radar or a camera to detect objects within certain proximity.

At a high level of abstraction, a swarm system follows the ”monitor-adapt-plan-execute” (MAPE) pattern [4]. The situation-awareness capabilities of the system provide the necessary run-time feedback on the mission execution. The coordinator collects the required telemetry data, performs the required adaptation to plan further steps of the mission and communicates the plans to the drones. By using its own navigation equipment, each drone executes the received plan. The overall system

behaviour is cyclic: at each cycle (usually called a *time frame*) the coordinator receives the telemetry data and analyses them, computes further steps of the mission using a route planning algorithm, and communicates the planned actions to each drone of the swarm.

Let us now discuss the dependability aspect of controlling the swarms of drones. A generic high-level requirement imposed on a broad range of swarm systems is to ensure the *the mission goal is achieved with a high likelihood despite unforeseen changes and deviations*. The main obvious implications of such a requirement are: firstly, to preserve the structural integrity of drones, i.e., ensure collision avoidance, and secondly, to minimise the risk of premature resource depletion, i.e., minimise the travelling distance of drones.

The problem of collision avoidance, or more generally, *motion safety* poses a significant challenge in engineering a large variety of autonomous vehicles. In the context of the swarm system, the complexity of solution is even higher, because a safety strategy should be devised for all drones in a coordinated manner. To ensure motion safety of a swarm of drones, we propose to define the following three types of requirements:

Req1 The drones do not collide with the static objects (obstacles).

Req2 The drones do not collide with each other.

Req3 The drones do not collide with the dynamically appearing objects not belonging to the swarm, e.g., airplanes or helicopters.

Let us consider **Req1**– expressing a collision avoidance with the static objects. Since the terrain of the flying zone is known for each mission, to ensure that **Req1** is satisfied, we should guarantee that the obstacles occurring on the flying altitude of the drones are faithfully introduced as the constraints of the planning algorithm and no unsafe routes are planned. The hazard "collision with a static obstacle" is known in advance and hence can be avoided by the appropriate design and verification measures.

Our next requirement **Req2** – drones do not collide with each other – is more challenging to address. Indeed, under the assumption of a fault-free system, to deal with **Req2**, we could have adapted the same strategy as for dealing with **Req1** – avoid placing the drones at the positions occupied by other drones while planning their routes. However, such an assumption is unrealistic because, due to internal problem or sudden changes in the environment (e.g., wind gusts), drones might deviate from their planned routes. Therefore, the hazard "mutual collision avoidance" is semi-dynamically emergent: under the nominal conditions, it can be avoided via the routing planning but requires monitoring and routing recalculation if deviations are detected.

Finally, to address **Req3** – the drones do not collide with the dynamically appearing objects not belonging to the swarm – in our system design, we should

include the mechanisms for dealing with the dynamically emerging hazards. Such mechanisms put stringent requirements on the situation-awareness capabilities of the system. To ensure that such a class of hazards can be mitigated, we firstly should guarantee a reliable object detection. Secondly, we should halt autonomous flight of the swarm, and execute controlled step-wise collision avoidance maneuver under the continuous monitoring of the sensor data. The autonomous flight is resumed only after the dynamically appearing object is no longer in a close proximity to the drones.

It is worth mentioning a special case of **Req2** – the situation when the deviations in the drones behaviour are extremely adverse and the time before the predicted collision is insufficient for the route recalculation. In this case, the system should execute the procedure similar to **Req3**: halt the swarm movement and execute a controlled maneuver. After the collision point is avoided, the autonomous flight mode can be resumed. Here we observe a transformation of a semi-dynamically emerging hazard into a dynamically emerging one.

A reverse transformation is also possible. For instance, consider a situation when a swarm should fly over an airport runway or approach zone. Since it is known in advance that an appearance of a dynamic object is likely in the corresponding set of positions, the dynamically emerging hazard can be treated as a semi-dynamically emerging one. When the drones approach the corresponding zone, while planning the continuation of the mission, the system should ask for the information from the air traffic controller.

In the next chapter, we demonstrate how to formalize the proposed motion safety requirements. Meanwhile, let us discuss the second aspect of dependability – reducing the risk of premature resource depletion. Each drone is a resource-constrained system and every time unit spent in the air consumes power. Therefore, achieving motion safety is a necessary, but not always sufficient condition for achieving system dependability. The route planning algorithm should also aim at minimising the travelling distance of each drone to ensure that the mission goal remains achievable under the given resource constraints.

While preplanning a mission, the designers should estimate whether the mission is feasible for the given resource characteristics of the system. Therefore, our route planning algorithm should introduce a predictable bounded overhead to the travelling distance of the swarm. Such an overhead should also be acceptable with respect to the mission type:

Req4 The travelling distance overhead required to achieve safety is predictable and bounded.

This is an important requirement that should be imposed on the route planning algorithm.

Finally, high performance of the algorithm should also be guaranteed to ensure dependability. Indeed, we assume that at each time frame the system implements

the MAPE pattern. Therefore, the following constraint formulated as **Req5** should be satisfied:

Req5 Performance of the algorithm is sufficient to complete calculations within each time frame, i.e.,

$$2 \times \textit{Max Communication Delay} + \textit{Algorithm WCET} \leq \textit{Time frame}$$

In this work, we propose an approach that integrates formal modelling to derive the dynamic architecture of the swarm control system, which satisfies requirements **Req1 - Req3**, and evolutionary computing to develop an efficient algorithm for safety-aware route planning, which comply with the requirements **Req4** and **Req5**.

In Section 2.4, we give an overview of Event-B – the formal framework in which we model requirements of motion safety and define the corresponding dynamic system architecture.

2.2 Evolutionary Algorithms: Basic Concepts

Evolutionary computing comprises a set of optimization algorithms, which are inspired by a biological or societal evolution [2]. The evolutionary algorithms (EA) are widely used in the autonomous systems due to their ability to find, in a highly performance way, near optimal solutions for the computationally hard problems. Let us briefly overview the basics of the evolutionary algorithms.

The aim of an EA is to find an optimal solution for some real world problem. The algorithm keeps on producing generations of the solutions until the generation achieving a near-optimal solution is found. An EA mimics the survival of the fittest principle of the nature. The first step in defining an EA is to establish a link between the real world and a computational world of EA. This step relies on two fundamental concepts: phenotype and genotype. Phenotype is a candidate solution (often called an individual) in the real world. A genotype (often called a chromosome) is an encoding of the phenotype in the domain of EA. For instance, if we are given an optimization problem, in which the possible solutions are integers, then we can represent them by their binary code, e.g., the phenotype 5 is represented by the genotype 101. A representation of a phenotype by a genotype is called encoding, while mapping of a genotype to a phenotype is called decoding.

The vast majority of EA starts from random generation of the initial population of genotypes in the overall search space according to a certain probability distribution. For each genotype, we can evaluate the fitness function, which represents the requirements to which the population should adapt. The fitness function assigns quality measures to the genotypes and drives the population improvement. An evaluation of a fitness function typically requires decoding of a genotype into the corresponding phenotype and computing a certain quality measure. For instance,

if the goal of EA is to find an integer x that maximizes x^2 then to evaluate fitness function of, say, a genotype 101, we first decode the given genotype 101 to the corresponding phenotype 5 and then compute square $5^2 = 25$.

An EA evaluates the fitness function for all genotypes of a given population. The genotypes with a higher values of the fitness function get the higher probability to be chosen as the parents of the next generation. The chosen parents undergo variations to create offsprings. A variation consists of mutation and recombination. Mutation is a unary operator applied to a genome to produce a (slightly) modified mutant – a child (offspring). Mutation is stochastic, i.e., the child depends of the outcomes of random choices. For instance, a mutation of a genotype represented by a bit-string can be achieved by a random flip of a bit. Recombination (or crossover) merges the information from two parent genotypes into offspring genotype. Similarly to mutation, the recombination is also stochastic – the choice of parents' genotype parts and the way of combining is random. Intuitively, recombination can be understood as mating two individuals with the different but desirable features to produce an offspring that combines both of those features.

Since the EAs keep the population size constant, we need to implement a survivor selection mechanism that chooses the individuals that remain in the next generation. Typically, it relies on the fitness ranking over the united set of parents and off-springs and selecting the top fittest segment as the next generation.

The algorithm terminates either when the predefined number of generations has been produced, the time allocated for running the algorithm has elapsed, or the successive generations bring only a negligible improvement (and hence further execution of the algorithm is impractical). There is a large variety of evolutionary algorithms – some of them are inspired by natural phenomena, while other by the social processes. An example of the former is ICA [3]. The algorithm simulates a human social evolution. Its parallel implementation [5] shows a remarkable performance in comparison with the other EAs and offers a promising solution supporting computationally intensive tasks of autonomous systems.

ICA starts by a random generation of a set of countries – the chromosomes – in the search space of the optimization problem. The fitness function determines the power of each country. The countries with the best values of the fitness function become Imperialists, the other countries become Colonies. The Colonies are divided between the Imperialists and hence the overall search space is divided into empires. An association of a Colony with an Imperialist means that only the chromosomes of the Imperialist and its associated colonies will be used to crossover. The intuition behind it as follows: since the Imperialist has a higher value of the fitness function, by crossing it over with an associated Colony, which is known to have a lower value of the fitness function, we inherit the strongest traits of the current population. The mutation and crossover are implemented by the *Assimilation* and *Revolution* operators. Assimilation moves colonies closer to an imperialist in its socio-political characteristics. For instance, it can be implemented by a replacement of a bit of a Colony chromosome by the corresponding bit (or a certain function over such a bit)

of the Imperialist.

Revolution results in a drastic change of a colony characteristic. It can be implemented by a random replacement of a certain bit in the Colony chromosome. As a result of *Assimilation* and *Revolution*, a colony might reach a better position and has a chance to take the control over the entire empire, i.e., to replace the current imperialist. This can, of course, happen only if the evaluation of the fitness function of such a colony gives a higher value (if we are solving a maximization problem) than the value of the fitness function of the current imperialist. The next step of the algorithm is to compute the power of each empire and implement the Imperialistic Competition, which corresponds to the selection of the survivals process. The power of an empire is computed as a sum of the value of the fitness function of imperialist and a weighted value of the sum of the fitness functions of the colonies.

The imperialists try to take a possession of colonies of other empires, i.e., the weakest empire loses its weakest colony. Indeed, the weakest empire does not offer a promising solution in the search space and further assimilation of colonies to the current imperialist would not bring any significant improvement. Therefore, it is practical to reallocate the weakest colony to a more promising empire. In each step of the algorithm, based on their power, all the empires have a chance to take control of one or more of the colonies of the weakest empire. The steps of the algorithm are repeated until a termination condition is reached. As a result, the imperialist of the strongest empire will give us a near-optimal solution.

Similarly to all EAs, ICA suffers from two main problems. Firstly, if we have to deal with a far-reaching search area, we need a large initial population to obtain an accurate and reliable result. Secondly, if computations of the fitness function and ranking are complex, the algorithm will suffer from severe performance degradation. We have developed a solution that overcomes these limitations and can also be easily parallelized. To solve the problem associated with a search in the far-reaching area, we introduce the notion of multi-population, i.e., we divide the overall search space into the clusters and perform the localized search first. The best local solutions are taken as the input to perform the search in the entire space. Obviously, since the local search procedures are independent of each other, they can be implemented in parallel. Moreover, the multi-population based search increases the reliability of the result, since it enables an exploration of a wider area by using migration operation.

To reduce the complexity associated with computation of the fitness function and ranking, we propose to use the simpler operations of *Genetic Algorithms (GAs)*. For instance, the ranking based on converting between phenotypes and genotypes requires much simpler computation as well as crossover. Such an approach results in the significant savings in terms of the required computing resources.

The benchmarking [5] experiments demonstrate that the parallel implementation of the proposed algorithm significantly outperforms similar parallel algorithms. Therefore, it makes our approach a suitable candidate for solving the routing problem of a swarm of drones – the topic that we discuss next.

2.3 Machine Learning

ML is a sub-field of artificial intelligence (AI). Generally, the goal of ML is to learn the structure of data first, and then fit that data into different models to predict and utilize the behaviour of the data. The main difference between traditional computational approaches and machine learning comes from their inputs and outputs. In traditional approaches, the algorithm receives input data (features) and solution (model) and generate the output data (labels). Machine learning receives input and output data and generates a solution. This solution is made based on the samples and their labels. It means ML depends on the quality of the data.

Although machine learning is a field within computer science, it differs from traditional computational approaches. In traditional computing, algorithms are sets of explicitly programmed instructions used by computers to calculate or solve a problem. The machine learning algorithms instead allow for computers to train on the given input data and use statistical analysis in order to output values that fall within a specific range. Because of this, machine learning facilitates computers in building models from sample data in order to automate the decision-making processes based on input data.

The ML methods are categorized into three main categories: supervised, unsupervised, and semi-supervised learning. In supervised learning, the algorithm is trained with input data (instances) that are labelled by their desired outputs. The algorithm learns the model of data by comparing the actual labels and predicted data to find the possible error. There are many different supervised methods such as support vector machines (SVM) [6], Linear regression [7], logistic regression [8], Naive Bayes [9], k-Nearest Neighbors (kNN) [10], etc. In this work, we have used two well-known supervised methods: *k-Nearest Neighbours* and *Linear regression*.

In unsupervised learning, the instances are unlabeled, so the learning algorithm is left to find commonalities among its instances. As unlabeled data are more plentiful than labelled data, ML approaches that promote unsupervised learning are particularly helpful. The main goal of unsupervised learning may be as straightforward as discovering hidden patterns within a dataset, but it may also have a goal of feature learning, which allows the computational machine to automatically discover the representations that are needed to classify raw data. There are also different unsupervised learning algorithm such as hierarchical clustering [11], k-means [12], mixture models [13], etc. In the thesis, we have chosen to use k-means.

2.3.1 k-Nearest Neighbours

The *k-Nearest Neighbours (kNN)* algorithm is a supervised learning method. The *kNN* can be used for classification and regression problems. The *kNN* is a fast and straightforward machine learning method. Similar to other learning methods, the quality of its results depend on the training data. The *kNN* predicts a value for any input data (test set) based on the labels of its k-nearest neighbours based on

the Euclidean distance between its place and other instances in the search space. The *kNN* makes a model for each instance in the test set independently. Means rebuilding the model takes more time. To reduce the total time, *kNN* relies on the hyper-parameter optimization to enable fast and simple operation for the model construction. As a result, a *kNN* works only based on one hyper-parameter. In different ML methods, tuning of hyper-parameters is the most time-consuming operation, and thus using only one hyper-parameter takes less time.

2.3.2 Linear Regression

One of the most popular supervised machine learning algorithms is Linear Regression. It performs a regression task. Regression generates an objective prediction value based on independent variables. This method is used for discovering the main relationship between forecasting and variables. The regression models differ based on the variety of correlations between the independent and dependent variables that are considered and the number of independent variables being used. Linear regression predicts a dependent variable value (y) based on a given independent variable (x). As a result, a regression algorithm determines a linear correlation between x (input) and y(output):

$$y = \Theta_1 + \Theta_2 x \tag{2.1}$$

2.3.3 k-Means

The *K-means algorithm* is an unsupervised and iterative algorithm that tries to categorize the data set into K pre-defined separate clusters where each instance refers to only one cluster. The algorithm tries to make the inter-cluster data points as similar as possible while also keeping the clusters as different (far) as possible. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid (i.e., the arithmetic mean of all the data points that belong to that cluster) is at the minimum. The smaller variation within clusters, the more homogeneous (similar) the data points are within the same cluster.

The way k-means algorithm works is as follows: *(i)* Specify number of clusters K; *(ii)* initialize centroids by first shuffling the data set and then randomly selecting K data points for the centroids without replacement; *(iii)* keep iterating until there is no change to the centroids. i.e., assignment of data points to clusters is not changing.

2.3.4 LeNeT-5

LeNet-5 [14] was used on a large scale to automatically classify hand-written digits on bank cheques in the United States. This network is a convolutional neural network (CNN) [15]. CNNs are the foundation of modern, state-of-the-art deep learning-based computer vision. These networks are built upon three main ideas:

local receptive fields, shared weights and spacial subsampling. The local receptive fields, with shared weights are the essence of the convolutional layer and most architectures described below use convolutional layers in one form or another. Another reason why LeNet is a vital architecture is that before it was invented, character recognition had been done mostly by using feature engineering by hand, followed by a machine learning model to learn to classify the hand-engineered features. LeNet made hand engineering features redundant because the network determines the best internal representation from raw images automatically.

By modern standards, LeNet-5 is a straightforward network. It only has seven layers, among which there are three convolutional layers (C1, C3 and C5), two sub-sampling (pooling) layers (S2 and S4), and one fully connected layer (F6), that are followed by the output layer [14]. Convolutional layers use five by five convolutions with stride 1. Sub-sampling layers are two by two average pooling layers. Tanh sigmoid activations are used throughout the network. There are several interesting architectural choices that were made in LeNet-5 that are not very common in the modern era of deep learning.

First, individual convolutional kernels in the layer C3 do not use all of the features produced by the layer S2, which is very unusual by today's standard. One reason for that is to make the network less computationally demanding. The other reason was to make convolutional kernels learn different patterns. This makes perfect sense: if different kernels receive different inputs, they will learn different patterns.

Second, the output layer uses 10 Euclidean Radial Basis Function neurons that compute L2 distance between the input vector of dimension 84 and manually predefined weights vectors of the same dimension. The number 84 comes from the fact that it is essentially the weights represent by 7x12 binary mask, one for each digit. This forces the network to transform the input image into an internal representation that will make outputs of layer F6 as close as possible to hand-coded weights of the ten neurons of the output layer.

LeNet-5 was able to achieve error rate below one percent on the MNIST data set, which was very close to state of the art at the time (produced by a boosted ensemble of three LeNet-4 networks).

2.4 Overview of the Event-B Framework

Event-B [16] is a state-based framework that promotes the correct-by-construction approach to system development and formal verification by theorem proving. In Event-B, a system model is specified using the notion of an *abstract state machine* [16]. An abstract state machine encapsulates the model state, represented as a collection of variables, and defines operations on the state, i.e., it describes the dynamic behaviour of a modelled system. A machine also has an accompanying component, called *context*, which includes user-defined sets, constants and their

properties given as model axioms.

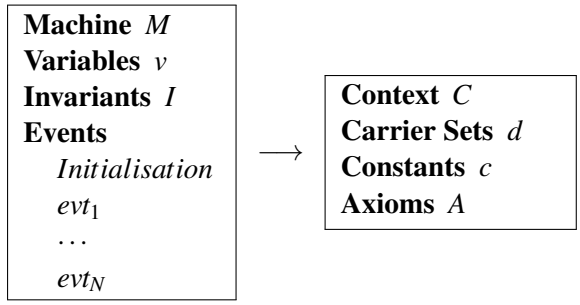


Figure 2.1: Event-B machine and context

A general form for Event-B models is given in Fig. 2.1. The machine is uniquely identified by its name M. The state variables, v, are declared in the **Variables** clause and initialised in the *Initialisation* event. The variables are strongly typed by the constraining predicates I given in the **Invariants** clause. The invariant clause might also contain other predicates defining essential properties (e.g., safety invariants) that should be preserved during system execution.

The dynamic behaviour of the system is defined by a set of atomic *events*. Generally, an event has the following form:

$$evt \mathrel{\widehat{=}} \textbf{any } a \textbf{ where } G_e \textbf{ then } R_e \textbf{ end},$$

where e is the event's name, a is the list of local variables, the *guard* G_e is a predicate over the local variables of the event and the state variables of the system. The body of an event is defined by a *multiple* (possibly non-deterministic) assignment over the system variables. In Event-B, an assignment is represented by the corresponding next-state relation R_e. Later on, using the concrete syntax in our Event-B models, we will rely on two kinds of assignment statements: deterministic ones, expressed in the standard form $x := E(x,y)$, and non-deterministic ones, represented as $x :| \mathit{some\_condition}(x,y,x')$. In the latter case, the state variable x gets non-deterministically updated by the value x' which may depend on the initial values of the variables x and y.

The guard defines the conditions under which the event is *enabled*, i.e., its body can be executed. If several events are enabled at the same time, any of them can be chosen for execution nondeterministically.

If an event does not have local variables, it can be described simply as:

$$evt \mathrel{\widehat{=}} \textbf{when } G_e \textbf{ then } R_e \textbf{ end}.$$

Event-B employs a top-down refinement-based approach to system development. Development typically starts from an abstract specification that nondeterministically models the most essential functional requirements. In a sequence of refinement steps,

we gradually reduce non-determinism and introduce detailed design decisions. In particular, we can add new events, split events, as well as replace abstract variables by their concrete counterparts, i.e., perform *data refinement*.

The consistency of Event-B models, i.e., verification of well-formedness and invariant preservation as well as correctness of refinement steps, is demonstrated by discharging a number of verification conditions – proof obligations. For instance, to verify *invariant preservation*, we should prove the following logical formula:

$$A(d,c),\ I(d,c,v),\ G_e(d,c,x,v),\ R_e(d,c,x,v,v') \vdash I(d,c,v'),(INV) \tag{2.2}$$

where A are the model axioms, I are the model invariants, d and c are the model constants and sets respectively, x are the event's local variables and v, v' are the variable values before and after event execution. The full definitions of all the proof obligations are given in [16].

The Rodin platform [17] provides an automated support for formal modelling and verification in Event-B. In particular, it automatically generates the required proof obligations and attempts to discharge them. The remaining unproven conditions can be dealt with by using the provided interactive provers.

2.5 Related work

The problem of motion safety of semi-autonomous robotic systems is currently attracting significant research attention [18]. A comprehensive overview of the problems associated with autonomous mobile robots is given in [19]. The analysis carried out in [20] shows that the most prominent routing schemes do not guarantee motion safety. Our approach resolves this issue and ensures not only safety but also efficiency at online routing.

Macek et al. [21] proposed a layered architectural solution for robot navigation. However, in their work, they focused on the safety issues associated with the navigation of a single vehicle and did not consider the problem of collision-free navigation in the context of swarms of robots. Aniculaesei et al. [22] presented a formal approach that employs formal verification to ensure motion safety. Petti and Fraichard [23] proposed an approach that relies on partial motion planning to ensure safety. Their solution supports navigation of a single vehicle. In our work, we have discretized the movement zone and have developed a highly efficient approach that computes the next safe states for an entire swarm and provides a mechanism for online route regeneration and collision avoidance.

Dong et al. [24] presented a software platform for real-time cooperative control of multiple drones. Their work focuses on monitoring and control of multiple drones from a ground control station. The approach does not generate paths for the drones. Instead, the complete flight information (including the drone paths) is provided to the ground control station that sends control commands to the drone fleet. Olivieri and Endler [25] presented an approach for movement coordination of

swarms of drones using smart phones and mobile communication networks. Their work focuses on the internal communication of the swarm and does not provide a solution for collision-free route generation. Bürkle et al. [26] proposed a multi-agent system architecture for team collaboration in a swarm of drones. They also developed a simulation platform for patrolling or surveillance drones, which monitor a protected area against potential intrusions. However, they did not address path planning and collision avoidance for the swarm.

Ivanovas et al. [27] proposed an obstacle detection and avoidance approach for a drone. Their approach uses computer vision techniques for detecting static obstacles in stereo camera images. The main focus of their approach is on how some block matching algorithms can be used for obstacle detection. However,they did not present a path planning and collision avoidance approach for multiple drones. Barry and Tedrake [28] proposed an obstacle detection algorithm for drones that allows the algorithm to detect and avoid collisions in real-time. Similarly, Lin [29] presented a real-time path planner for drones that detects and avoids moving obstacles. These approaches are only applicable for individual drones and they do not provide support for a swarm of drones. In our work, we focused on collision prediction and avoidance and efficient navigation of swarms of drones.

A comprehensive literature review on motion planning algorithms for drones can be found in [30]. The approaches reviewed in [30] are applicable to the preliminary, offline motion planning phase to plan and produce an efficient path or trajectory for a drone before the start of the mission. A more recent survey on motion planning of drones can be found in [31]. Augugliaro et al. [32] also presented a planned approach for generating collision-free trajectories for a drone fleet. In contrast to these approaches, our proposed approach combines offline motion planning with a more realistic online route generation approach to produce efficient collision-free routes.

In our previous works [33] and [34], we proposed a path planning and navigation approach for a swarm of drones. We have combined the offline path planning with an online navigation approach and used machine learning and evolutionary algorithms to generate efficient paths while maximizing safety of the drones in a swarm. We have also used collision prediction and drone reflexes to prevent collisions with unforeseen obstacles.

Sujit and Beard [35] proposed a particle swarm optimization (PSO) based algorithm for path planning for a swarm of drones. In their approach, whenever a drone detects a moving obstacle, the anytime algorithm generates a new path for the drone depending on the time allowed to compute a new path before a collision can occur. Although the anytime algorithm avoids collisions with moving obstacles, it works only in certain cases in which the obstacles can be detected from a significant distance. In comparison to the PSO-based algorithm, our proposed approach combines an efficient path generation algorithm with a drone reflex computation algorithm to avoid moving obstacles collisions in various kinds of scenarios. Moreover, our clustering algorithm reduces the computation time by allowing to selectively regenerate

paths in a close proximity of a moving obstacle. Silva Arantes et al.[36] presented a drone path planning approach for critical situations requiring an emergency landing of a drone. Their approach uses greedy heuristics and genetic algorithms (GGA) to generate and optimize feasible paths under different types of critical situations caused by equipment failures.

Currently, the problem of motion safety of autonomous robotic systems attracts significant research attention. A comprehensive overview of the problems associated with autonomous mobile robots is given in [37]. The analysis carried out in [38], shows that the most prominent routing schemes do not guarantee motion safety. Our approach resolves this issue and ensures not only safety but also efficiency of routing. Macek et all [39] have proposed a layered architectural solution for robot navigation. They focus on a problem of safe navigation of a vehicle in an urban environment. Similarly to our approach, they distinguish between the global route planning and a collision avoidance control. However, in their work, they focus on the safety issues associated with the navigation of a single vehicle and do not consider the problem of route optimization that is especially acute in the context of swarms of robots.

A formal approach that employs formal verification to ensure motion safety has been proposed by Aniculaesei et al. [40]. They employ UPPAAL to verify that a moving robot engages brakes and safely stops upon detection of an obstacle. Since in our work we have focused on finding an algorithm that optimizes the safety/efficiency ratio, we have only employed formal verification to ensure that our algorithm preserves the required safety properties. The solution proposed in our work is more performant and flexible – it allows the system to dynamically recalculate the route to prevent a collision and avoids unnecessary stopping of drones.

Petti and Fraichard [41] have proposed an approach that relies on a partial motion planning to ensure safety. They state that a calculation of an entire route is such a complex and computationally-intensive problem that the only viable solution is a computation of the next safe states and navigation within them. The solution is proposed for navigation of a single vehicle. In our work, to overcome the problem of heavy computational costs and hence insufficiently quick response, we have on the one hand, discretized the search space, and on the other hand, developed a highly performant algorithm that guarantees the desired responsiveness. As a result, we could not only calculate the entire safe and efficient routes, but also solve this task for a swarm of drones.

The problem of dynamic configuration of swarm-based monitoring systems is novel, and to the best of our knowledge, the similar integrated solutions have not been described in the literature. Therefore, in our overview of the related work, we describe the research focusing on a similar problem of sensors placement in the mobile sensor networks. Finding the optimal placement to improve connectivity, reliability and energy consumption is an NP-hard problem [42]. There are different heuristic methods to solve this problem [43], [44]. Thomas et al. [45] have devel-

oped a heuristic algorithm to increase the network lifetime by iteratively moving a router node to a better location. In [46], the authors present several trajectory control algorithms with different assumptions regarding locating capabilities to achieve the objectives of re-ducing the hop count and overhead. Youssef et al. [47] have employed genetic algorithms (GAHO) for selecting the best spot for placing each gateway so that that sensor data can be delivered to a gateway with the least latency. Krause et al. [48] presented a data-driven approach (pSPIEL) that addresses the three central aspects of this problem: measuring the predictive quality of a set of sensor locations, predicting the communication cost incurred by these placements, and designing an algorithm that is guaranteed to optimize the NP-hard tradeoff. Dhumathi et al. [49] analyzed the total coverage area of a wireless sensor network, identified the types of sensor nodes and coverage sensing distance, and calculated the coverage sensing distance for the combination of all sensor types based on the radius of each node. Majd et al. [50] presented two static methods for positioning drones. They aimed at improving the deployment of dynamic nodes using a genetic algorithm (DGA) to maximize the coverage. These methods have focused either on maximizing the coverage or minimizing the energy consumption. However, DIANA is the first framework to address both issues simultaneously. Another difference between our work and the related works is that our approach utilizes a multi-population implementation to find more reliable results in solving a complex NP-hard multi-objective problem [51]. Furthermore, our approach is specifically tailored to swarm configuration.

Chapter 3

Proposed Method

3.1 Thesis Contributions

Today, due to the complexity of modern AVs, AI methods have emerged as a vital solution. Generally, AI are time-demanding computational models for complex problems, particularly when the search spaces and the number of objectives are large. In this thesis, we focus on safe and efficient navigation and placement of AVs. These two problems are optimization problems that can be solved in static and dynamic environments. Among of EA's we have selected ICA due to its high performance and accuracy.

In Paper 1, we have used ICA to solve the optimal placement of swarm of drones in an 2D static situation. The proposed method obtained good results, but did not address dynamic environments. In Paper 2 and 3, we have presented a real-time, collision-free motion coordination and navigation system for an Unmanned Aerial Vehicle (UAV) fleet. This approach uses geographical locations of the UAVs and their movement interaction about static and moving obstacles to predict and avoid: (1) UAV-to-UAV collisions, (2) UAV-to-static-obstacle collisions, and (3) UAV-to-moving-obstacle collisions. Our collision prediction approach leverages efficient runtime monitoring to make timely predictions. A distinctive feature of the proposed system is its ability to foresee a risk of a collision in real-time and proactively find the best ways to avoid the predicted collisions in order to ensure the safety of the entire fleet.

In Paper 4, as the placement and safe navigation of a swarm of AVs are multi-objective problems, we have proposed a dynamic EA method to be applied in the real-time environments. In Paper 5 and Paper 6, a two-layer hybrid method to solve safe and efficient placement problem has been presented. These two layers are called the off-line and real-time layers. The off-line layer computes some predicted tasks such as the distance between two different points. In the online layer, an efficient dynamic EA suggests the most efficient and safe paths. To avoid any collision between drones, a central decision centre checks the safety of suggested

paths.

The obtained results show that the proposed work has better performance than other similar methods. In the Paper 7, to improve the speed and accuracy of our approach, we have added a supervised machine learning algorithm that works synchronously with our dynamic EA method to generate two independent solutions for our path planning problem. In our approach, we need to build a model very fast, be able to rebuild it efficiently in dynamic environments. Therefore, *kNN* has been selected as a well-know, fast, accurate and straightforward supervised learning method. The *kNN* in our approach uses the most similar instances based on the number of drones in the future, and our training set updates continuously. Avoiding collision between drones and dynamic obstacles is the most challenging problem in the navigation of AVs. Therefore, we have added one more component to analyze the movement of dynamic obstacles and predict their next movements. This component uses linear regression to predict the movements of dynamic obstacles.

We improved our proposed method in Paper 8. In this paper, to maximize safety, we expand the generated routes with dynamically computed drone reflexes. The drone reflex computation module mimics a self-preservation control mechanism of humans. The reflexes are automatic immediate or mechanical responses to particular hazardous situations, such as quickly moving the hand away from a hot surface. They aim at relieving and confining the effects and damages of suddenly occurring hazards. In our proposed approach, when a drone detects a possible collision with an unforeseen obstacle, the drone reflexes computation module quickly computes a reflex movement for the drone to prevent and mitigate the collision. The reflex movement is computed in the AVs layer (drone's layer).

In all complex problems, when size of the problem increases, the computation time will be increased. In Paper 9, we introduced our proposed framework **DIANA**. To overcome complexity, we have used k-Means to cluster the flying zone. The k-Means is a well-known unsupervised machine learning algorithm. By dividing the flying zone into several distinct areas, the problem is broken down into smaller areas and thus it is more straightforward to solve. Also, in this paper, for the first time, we have used a deep neural network (*LeNet-5*) to generate 30 percent of the initial population of the dynamic EA method. In normal EA methods, the initial population is generated randomly, but in this work, the *LeNet-5* method generates different chromosomes based on the current situation of the environment. Also, the algorithm can start its exploration based on some chromosomes that have more chances to be selected as the best solution. In Paper 10, for the optimal dynamic placement of drones, we have used the **DIANA** framework.

In Paper 11, we demonstrated formally how to derive the resilience-enhancing mode transition logic from the goals that the system should achieve. In Paper 12, we have proposed a novel multi-layered approach to ensuring the safety of drone navigation. The approach aims at maintaining the most optimal ratio between efficiency of mission execution and safety at a hierarchical distributed way. We formalize the proposed approach in Event-B and derive the coordination and reconfiguration

mechanisms, ensuring efficiency and safety of mission execution. The obtained results show the **DIANA** can achieve efficient and safe results in the navigation and placement of swarm of AVs in a dynamic environment. Finally, Paper 13 builds on the previous results by proposing a multi-layered architecture of autonomous systems. We define the notations of strategic, tactic, activity safety and underlying communication model to implement the defined complementary safety mechanism.

3.2 The DIANA Framework

Figure 3.1 presents the structure of the proposed framework for intelligent navigation of swarms of drones called **DIANA**. The *Offline Part* uses a *Dynamic Evolutionary Algorithm (DEA)* to generate drone routes before the start of the mission. Moreover, it computes and uses the shortest paths between the start and destination locations of the drones. The shortest paths are computed by using the Dijkstra's algorithm [52]. Since a drone swarm is a highly dynamic system, we augment our offline module with an *online* approach that provides run-time means for monitoring and reconfiguration.

The information obtained from the *Dynamic Monitoring* component has two main purposes. On one hand, it is a feedback mechanism. On the other hand, it allows us to detect the changes in the drone swarm and in the flying zone. Such changes may invoke swarm reconfiguration and regeneration of the drone routes. In addition, the *Prediction* module uses the run-time monitoring data to predict the movement of drones and moving obstacles in the flying zone. We feed the monitoring data and prediction results to both *DEA* and *kNN* algorithms. As result, these modules compute the alternative routes. The routes are compared by the *Decision Center* that chooses the best solutions from the proposed alternatives. The *Navigation Center* then issues the corresponding commands to the drones. The *Clustering Algorithm* splits the set of drones into smaller subsets called clusters. A cluster is formed by first checking the geographical locations of all drones and then by grouping the drones base on their proximity to one another. The *Safe Area Computation* and *Reflexes Computation* modules compute the safe area and drone reflexes, respectively.

They implement our proposed drone reflexes approach [33]. The safe area computation module uses the information of the known and predicted obstacles to compute a safe area for each drone. Moreover, when a drone suddenly detects a possible collision with an unpredicted obstacle, the reflexes computation module quickly computes a reflex movement for the drone to prevent and mitigate the collision.

For the DEA module, **DIANA** uses our parallel implementation of the ICA[53] with an integrated migration operation (referred henceforth as the MICAP algorithm). The MICAP algorithm computes the initial drone routes in the offline part as well as the subsequent new routes during the execution of the mission. For the

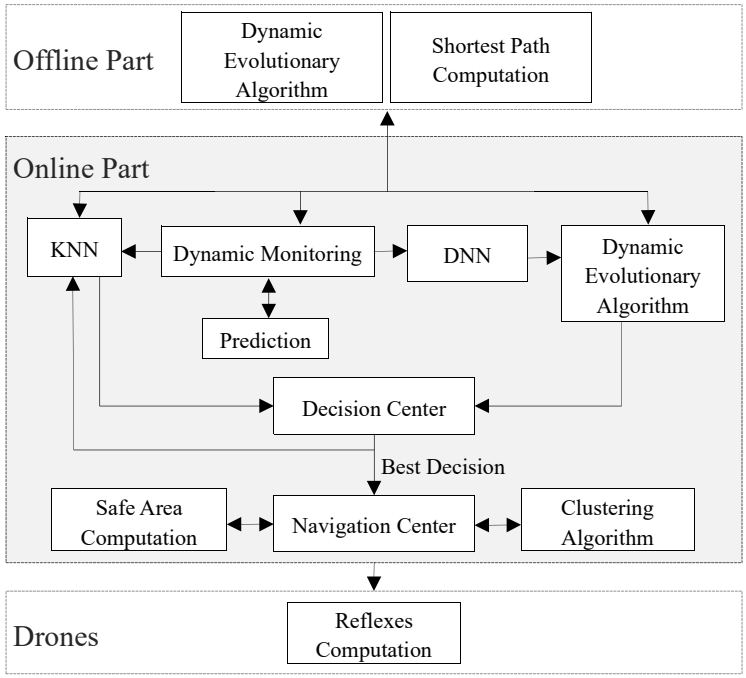


Figure 3.1: The proposed **DIANA** framework

learning module, we use k *kNN*-based learning algorithm[54] that runs alongside the MICAP algorithm.

The main steps of the **DIANA** approach are presented in Algorithm 1. The algorithm starts with the preplanning of the mission – the offline execution of MICAP to find the initial routes (Line 1-4). It then obtains the current actual and predicted state of the system by invoking the dynamic monitoring and prediction modules (Line 7) and runs the MICAP algorithm and the kNN-based learning algorithm until the mission completes. In each iteration, it compares the results of MICAP with that of the kNN-based learning algorithm and selects the best solutions (Line 18 and 20). In the next step, our clustering algorithm generates drone clusters (Line 8). It is based on the k-means clustering algorithm [55]. In each iteration, the safe area computation module computes and updates the safe area for each drone (Line 9). If a drone detects an unpredicted obstacle in its safe area (Line 10), the drone reflex computation module is invoked to prevent and mitigate the collision (Line 12-13). Moreover, the drone clusters and the best drone routes are regenerated (Line 15-18).

The pseudocodes of the proposed MICAP, kNN, prediction, safe area computation, and drone reflex computation algorithms are presented in Procedure 2, 3, 4, 5, and 6, respectively. All five procedures are invoked by the main **DIANA** algorithm presented in Algorithm 1. The MICAP, kNN, prediction, and safe area

Algorithm 1 DIANA

{**Offline part in the cloud layer**}
Compute the shortest path for each drone $\in D$
Best-Routes $\leftarrow$ Call MICAP(Initial-State)
Send(Best-Routes) $\rightarrow$ Navigation Center
{**Online part in the cloud layer**}
while Mission is in progress **do**
  Current-State $\leftarrow$ Call Dynamic Monitoring and Prediction modules
  Generate drone clusters by using k-means clustering
  FD $\leftarrow$ Call Compute-Safe-Area(Current-State)
  if drone d_q detects an unpredicted obstacle in its safe area **then**
    {**Online part in the drones layer**}
    Reflex-Position $\leftarrow$ Call Compute-Drone-Reflex(Obstacle-Position, FD)
    Prevent and mitigate the collision by moving the drone to Reflex-Position
    {**Online part in the cloud layer**}
    $\text{Safe}_{d_q} \leftarrow \{\forall d_j \in D \mid Distance\text{˙j} \leftrightarrow q \leq r\}$
    $\text{Layer2}_{d_q} \leftarrow \{\forall Cl_p^i \exists 1 \leq j \leq c | Cl_j^i \cap \text{Safe}_{d_q} \neq \emptyset\}$
    $D_{temp} \leftarrow \text{Layer2}_{d_q} \cup \text{Safe}_{d_q}$
    Call Compute-Send-Best-Routes(D_{temp})
  else
    Call Compute-Send-Best-Routes(D)
  end if
end while

Procedure 1 Compute-Send-Best-Routes(Current-State)

Best-EC-Result $\leftarrow$ Call MICAP(Current-State)
Best-kNN-Result $\leftarrow$ Call KNN(Current-State)
Best-Routes $\leftarrow$ Max(Best-EC-Result, Best-kNN-Result)
Send(Best-Routes) $\rightarrow$ Navigation Center
Send(Best-Routes)$\rightarrow$ KNN-Dataset

computation modules can be implemented on a remote server in the cloud layer, while the latency-sensitive operation involving the computation of drone reflexes runs in the drones layer or on a fog or edge server. Our MICAP algorithm, learning algorithm, prediction algorithm, drone reflex computation algorithm, and the clustering algorithm are described in the following subchapters.

3.2.1 Dynamic Evolutionary Algorithm

A swarm of drones is a typical example of a complex distributed networked system [56]. Each drone can be seen as a mobile sensing node that is capable of collecting monitoring data and communicating with some other drones in the swarm as well as

Procedure 2 MICAP(Current-State)

if Processor P_i **then**
  for j=1 to Population Size **do**
    Initial-Population [j] $\leftarrow Country_j$
    Cost_Initial-Population[j] $\leftarrow \alpha \cdot$ Safety-Level + $\frac{\beta}{Path-Length}$
  end for
  Initial-Population $\leftarrow$ Sort Initial-Population
  for j=1 to #Empire **do**
    Imperialist[j] $\leftarrow$ Initial-Population [j]
  end for
  for j=#Empire+1 to Population Size **do**
    Assigned as a Colony to an Empire
  end for
  for All Colony **do**
    Assimilate Colony $\rightarrow$ Imperialist
  end for
  for All Colony **do**
    if Colony $\geq$ Imperialist **then**
      Colony $\leftrightarrow$ Imperialist
    end if
  end for
  for j=1 to #Empire **do**
    Empire Power[j] $\leftarrow$ Cost(Imperialist$_j$) + $\xi \cdot$ Mean Cost(Empire$_j$ Colonies)
  end for
  Temp $\leftarrow$ Worst Country$_{WorstEmpire}$
  Empire$_{l \neq WorstEmpire} \leftarrow$ Temp
  if #Iterations % Migration Gap=0 **then**
    Best Country $\leftarrow$ Worst Country$_{ProcessorP_{(i+1)\%\#Processors}}$
    Worst Country $\leftarrow$ Best Country$_{ProcessorP_{(i+1)\%\#Processors}}$
  end if
  if termination condition **then**
    return Best Country
  end if
end if

Procedure 3 kNN(Current-State)

for All Instance $\in$ Dataset **do**
  Find Nearest-Neighbor {see Section 3.2.3}
end for
return Nearest-Neighbor

Procedure 4 Prediction

1: {see Section 3.2.4}
2: Invoke three parallel instances of the Tracker
3: Invoke three parallel instances of the Predictor
4: **return** prediction results

Procedure 5 Compute-Safe-Area(Current-State)

1: **for** i in 1 to number of drones **do**
2: $Ds_i \leftarrow$ (Drones in safe area $\cup$ Obstacles in safe area)
3: **for** j in 1 to $|Ds_i|$ **do**
4: $FD_i = \sum_{k=1}^{o} (F_{O_j \rightarrow D_i} \cdot (r - Dis_{D_i \rightarrow O_j}))$
5: **end for**
6: **end for**
7: **return** FD

Procedure 6 Compute-Drone-Reflex

1: $FOb \leftarrow F_{O_{unpredicted\_obstacles} \rightarrow D_i} \cdot (r - Dis_{D_i \rightarrow O_{unpredicted\_obstacles}})$
2: $SD \leftarrow FOb + FD_i$
3: **return** SD

with the cloud-based navigation center. Finding an efficient and safe route for each drone is a complex optimization problem. Therefore, we need to rely on certain heuristics to achieve the required objectives. In this thesis, we use a dynamic EA to compute the drone routes.

Evolutionary computing comprises a set of optimization algorithms, which are inspired by a biological or societal evolution [57, 58]. EAs are widely used in the swarm systems due to their ability to find, in a high-performant way, near-optimal solutions for the computationally hard problems. An EA mimics the survival of the fittest principle of the nature.

Most EAs start from a random generation of the initial population of genotypes – the encodings of candidate solutions – in the overall search space according to a certain probability distribution. For each genotype, we can evaluate the fitness function, which represents the requirements to which the population should adapt. A fitness function assigns quality measures to the genotypes and drives the population improvement. The genotypes with the higher values of the fitness function get a higher probability to be chosen as the parents of the next generation. The chosen parents undergo variations to create offsprings. A variation consists of mutation and recombination. Mutation is a unary operator applied to a genome to produce a (slightly) modified mutant – a child (offspring) [58]. Mutation is stochastic, that is, the child depends on the outcomes of random choices. Recombination (or crossover) merges the information from two parent genotypes into offspring genotypes. Similarly to mutation, the recombination is also stochastic. Since the EAs keep the

population size constant, we need to implement a survivor selection mechanism that chooses the individuals that remain in the next generation. Typically, it relies on the fitness ranking over the united set of parents and offsprings and selecting the top fittest segment as the next generation. The algorithm terminates either when the predefined number of generations have been produced, time allocated for running the algorithm has elapsed, or the successive generations bring only a negligible improvement.

There is a large variety of EAs. Some of them are inspired by natural phenomena, while others, such as ICA [59], by the social processes. The algorithm simulates a human social evolution [58]. Its parallel implementation[53] has shown a remarkable performance in comparison with the other EAs and offers a promising solution supporting compute-intensive tasks of swarm-based systems.

Figure 3.2 presents a flowchart highlighting the main steps in the ICA. The algorithm starts by a random generation of a set of countries – the genotypes – in the search space of the optimization problem [58]. The fitness function determines the power of each country. The countries with the best values of the fitness function become *imperialists*, while the other countries become *colonies*. The colonies are divided among the imperialists and hence the overall search space is divided into empires. An association of a colony with an imperialist means that only the genotype of the imperialist and its associated colonies are used for crossover. The intuition behind it as follows: since the imperialist has a higher value of the fitness function, by crossing over with an associated colony, which is known to have a lower value of the fitness function, we inherit the strongest traits of the current population.

The mutation and crossover are implemented by *assimilation* and *revolution* operators. The Assimilation moves colonies closer to an imperialist in its socio-political characteristics. For instance, it can be implemented by replacing a certain bit in a colony genotype with the corresponding bit of the imperialist. Revolution

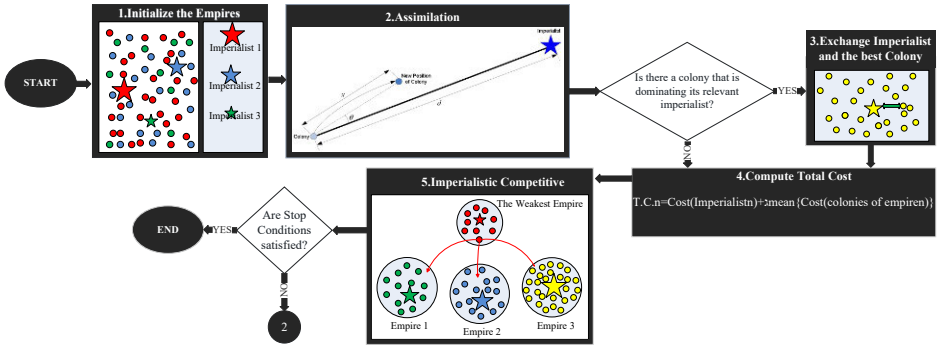


Figure 3.2: A flowchart highlighting the main steps in the imperialistic competitive algorithm

results in a drastic change of a colony's characteristics. It can be implemented by a random replacement of a certain bit in the colony genotype. As a result of assimilation and revolution, a colony might reach a better position and get a chance to take over the control of the entire empire, that is, to overthrow the current imperialist. This can happen only if the evaluation of the fitness function of such a colony gives a higher value (when solving a maximization problem) than the value of the fitness function of the current imperialist.

The next step of the algorithm computes the power of each empire and implements the imperialistic competition, which corresponds to the selection of the survivals process. The power of an empire is computed by aggregating the fitness value of the imperialist and a weighted sum of the fitness values of the colonies. The imperialists also try to take possession of colonies of other empires, that is, the weakest empire loses its weakest colony. In each step of the algorithm, based on their power, all the empires get a chance to take control of one or more of the colonies of the weakest empire. The steps of the algorithm are repeated until a termination condition is reached. As a result, the imperialist of the strongest empire produces the best solution. To improve performance of ICA, we introduce the notion of multi-population, that is, we divide the overall search space into multiple populations and perform a local search within each one of them. The best local solutions are then taken as input to perform the search in the entire search space. The multi-population based search also allows to use the inter-population *migration* operation, which migrates the best country from one population and uses it to replace the worst country in another population. Since the local search procedures are independent of each other, they can be implemented in parallel. Moreover, the multi-population based search enables a wider exploration of the search space, which helps to find high quality solutions.

3.2.2 Deep Neural Network

In most of the evolutionary algorithms, the initial population of chromosomes is generated randomly. As **DIANA** deals with very long chromosomes in a complex problem, a randomly generated initial population may not provide good results. Therefore, **DIANA** uses a deep neural network (DNN) to generate a subset of the initial population of chromosomes. It uses a convolutional neural network (CNN) to generate 30% of the initial population based on the previous experience data set. The choice if a CNN is influenced by the fact that it provides a suitable structure for the problem under consideration. **DIANA** finds safe and efficient drone routes in a three-dimensional grid or mesh of points, which is similar to a colored photo (two dimensions for the photo size and one dimension for representing the color in RGB). In the grid, each point can possibly be used for the placement and movement of drones and can be assigned different weights in different situations. CNNs are based on three core ideas: (1) local receptive fields, (2) shared weights, and (3) spacial subsampling [60].

The CNN component in **DIANA** starts with a small set of data. Due to the lack of data in the beginning, CNN does not produce high-quality chromosomes. In the worst case scenario, the CNN component may produce a population similar to the one produced by a random population generated. However, as more and more data become available to **DIANA**, the results of the CNN component improve and it starts producing high-quality chromosomes.

The proposed CNN comprises eight layers as presented in Figure 3.3. It is based on an extension of *LeNet-5* [60]. Our extension uses a fully-connected layer (1024×1 FC layer) for the first layer to convert different three-dimensional area to a linear input data for the same network. With this extension, we do not need to rebuild the whole network for different problems. Also, in the last layer, we use a one-dimensional layer, with the layer size equal to the chromosome size. Therefore, the values of the last layer can be used as chromosomes. Our network receives information of all locations from the dynamic monitoring component and produces chromosomes for the initial population. During this process, the network is also trained to produce high-quality chromosomes.

When compared with other contemporary CNNs, *LeNet-5* is a simple network with seven layers. These layers include two convolutional layers (Conv 1, and Conv 2), two subsampling or pooling layers (Pool 1 and Pool 2), two fully connected layers (120 FC), and the output layer (is a fully connected that its size is equal to the length of the chromosome) [60]. The convolutional layers in *LeNet-5* use five by five convolutions with stride one, while the subsampling layers are two by two average pooling layers.

3.2.3 k-Nearest Neighbor (kNN)

The *kNN* algorithm [54] is a popular learning method. In this thesis, we use it for classification. The main idea of the algorithm is as follows. We select K samples in the training set. The algorithm predicts numerical target – the nearest neighbor – based on the similarity measure, which in our case is the distance function. We compute the numerical target as the average of the Euclidean distances of the K nearest neighbors.

In our approach, the algorithm takes as the input K solutions generated for the same system state. Then it computes an average solution. For each drone in the swarm, it takes K routes proposed for it. Then it computes an average route in terms of the Euclidean distance. By computing such a route for each drone in the swarm, the learning component calculates the complete solution for the swarm. The decision center uses the fitness function to compute fitness values for the solutions produced by MICAP and the learning algorithm. It then chooses the solution with the highest fitness value.

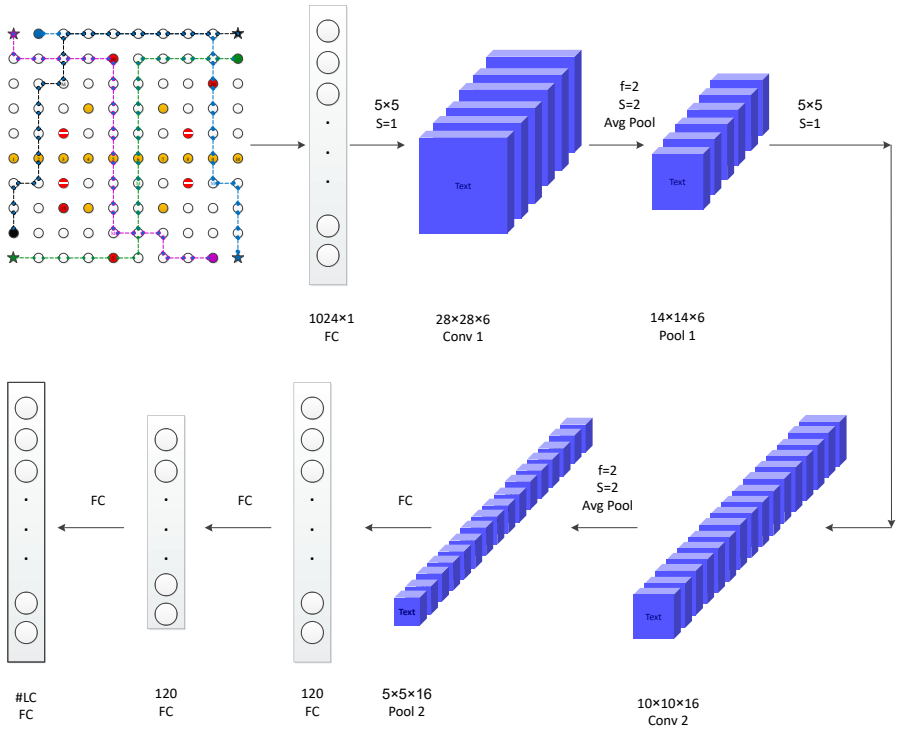


Figure 3.3: A CNN for generating the initial population of chromosomes

3.2.4 Prediction Module

The prediction module implements a two-step approach [61, 62, 34], which allows to make predictions under real-time constraints. The approach involves a set of the *trackers* that offer a representative view of the trends to the *predictors*, thus achieving the two-step approach.

A tracker filters out the noise and can be used to yield a more regular view of the trend of an obstacle or drone movement [34]. It takes as an input a raw measure s_i monitored at time t_i, and a set of previously collected n measures, that is $\overrightarrow{S_n}(t_i) = (s_{i-n}, ..., s_i)$, and outputs a representation of the trends l_i at time t_i. Formally, a tracker is a function $Tracker(\overrightarrow{S_n}(t_i)) : \mathbb{R}^n \rightarrow \mathbb{R}$. Multiple applications of a tracker provides a sequence of values that yield a regular trend of the movement. There are different classes of linear and non-linear trackers, such as simple moving average (SMA), exponential moving average (EMA), and cubic spline (CS)[61]. SMA is a simple linear method, having however the well-known shortcomings of increased oscillations, when n is small, and of significant delay, when n is large. CS is a non-linear method that is more expensive to compute than SMA and EMA, but instead of returning a new tracker value for each raw measure, it returns a new tracker value after n measures. More sophisticated (time-series) models often require training

periods to compute the parameters and/or off-line analyses. Similarly, the linear (auto) regressive models, such as ARMA and ARIMA, usually require frequent updates to their parameters in the case of highly variable systems [63]. Therefore, the proposed approach implements a tracker based on the EMA model, which limits the delay without incurring oscillations and computes a tracker value for each measure [62].

EMA is the weighted mean of the n measures in the vector $\overrightarrow{S_n}(t_i)$, computed at time t_i, where $i > n$, and the weights decrease exponentially. An EMA based tracker is defined as

$$EMA(\overrightarrow{S_n}(t_i)) = \alpha \cdot s_i + (1-\alpha) \cdot EMA(\overrightarrow{S_n}(t_{i-1})) \tag{3.1}$$

where $\alpha = \frac{2}{n+1}$. The initial value $EMA(\overrightarrow{S_n}(t_n))$ is set to the arithmetic mean of the first n values

$$EMA(\overrightarrow{S_n}(t_n)) = \frac{\sum_{j=0}^{n} s_j}{n} \tag{3.2}$$

The predictor takes as input a set of tracker values $\overrightarrow{L_q}(t_i) = l_{i-q}, ..., l_i$ and outputs the future tracker value at time t_{i+k}, where $k > 0$. Formally, a predictor is a function $Predictor_k(\overrightarrow{L_q}(t_i)) : \mathbb{R}^q \rightarrow \mathbb{R}$. With the use of the trackers that provide high correlation among values, even simple linear predictors are sufficient to predict the future trend of the movement. The predictor is characterized by the prediction window k and the past time window q. Using a simple linear regression model [64], the predictor uses the last q tracker values $\overrightarrow{L_q}(t_i)$ [62]. It is based on a straight line defined as

$$l = \Theta_0 + \Theta_1 \cdot t \tag{3.3}$$

where Θ_0 and Θ_1 are unknown constants, called regression coefficients, which can be estimated at runtime based on the tracker values $\overrightarrow{L_q}(t_i)$ in the past time window. One common approach to estimate these regression coefficients is to use the least-square estimation method [64]. The least-square estimators of Θ_0 and Θ_1, say $\hat{\Theta}_0$ and $\hat{\Theta}_1$, are computed as

$$\hat{\Theta}_0 = \bar{l} - \hat{\Theta}_1 \cdot \bar{t} \tag{3.4}$$

and

$$\hat{\Theta}_1 = \frac{\sum_{j=i-q}^{i} (l_j \cdot t_j) - \frac{\left(\sum_{j=i-q}^{i} l_j\right) \cdot \left(\sum_{j=i-q}^{i} t_j\right)}{q}}{\sum_{j=i-q}^{i} t_j^2 - \frac{\left(\sum_{j=i-q}^{i} t_j\right)^2}{q}} \tag{3.5}$$

where

$$\bar{l} = \frac{1}{q} \sum_{j=i-q}^{i} l_j \quad and \quad \bar{t} = \frac{1}{q} \sum_{j=i-q}^{i} t_j \tag{3.6}$$

The predictor returns a predicted future tracker value $\hat{l}_{i+k}$ that corresponds to time t_{i+k}. It is computed as follows:

$$Predictor_k(\overrightarrow{L_q}(t_i)) = \hat{l}_{i+k} = \hat{\Theta}_0 + \hat{\Theta}_1 \cdot t_{i+k} \tag{3.7}$$

The prediction results depend upon the selection of proper values for the tracker and predictor parameters. Therefore, it is necessary to find a value for n that represents a good trade-off between a reduced delay and a reduced degree of oscillations [63]. Similarly, the values of q and k should be selected carefully.

In a three-dimensional flying zone, the current location of an obstacle or a drone monitored at time t_i contains three values (x, y, z), which represent the three axis. Therefore, for predicting the future location of an obstacle or a drone at time t_{i+k}, we use three parallel trackers and three parallel predictors.

3.2.5 Drone Reflexes

The computation of the drone reflexes is elaborated in the example depicted in Figure 3.4 and 3.5 [33]. Figure 3.4 shows the safe area for drone d_1. It shows the radius of the safe area ($r = 4$), the repulsive forces of drone d_2 and d_3 to d_1 (F2 and F3), and the total force direction (FD). In each iteration, our algorithm computes FD based on all drones in the safe area. The computation of the repulsive forces is based on the distances of drone d_2 and d_3 from d_1 (Dis_2 and Dis_3). In this example, Dis_2 and Dis_3 are $\sqrt{8}$ and 3, respectively. In each iteration, the repulsive forces and FD are computed in the online part in the cloud layer, as shown in Algorithm 1 (Line 9). For example to prevent drone d_1 from colliding with d_2, the repulsive force for d_1 is computed as $F2 = (r - Dis_2)$ along with the reflex direction in the opposite direction of d_2.

Figure 3.5 illustrates that when drone d_1 detects an unpredicted obstacle, it computes its reflex position based on the total force direction (FD) and the force of the obstacle to d_1 (FOb). FOb is a unit vector in the opposite direction of the obstacle. Finally, the drone computes the safe direction (SD) for reflex movement. SD is a vector from the drone to a safe position in the flying zone. It is computed as the sum of FD and FOb, as shown in Procedure 6.

3.2.6 Clustering Algorithm

The proposed clustering approach is based on the k-means clustering algorithm[55]. Figures 3.6 to 3.8 present an example of our clustering approach. Figure 3.6 shows a configuration of the drones in the flying zone at step i. It also illustrates the drone clusters at step i and the movement directions of the drones for the next step i+1.

Figure 3.7 depicts the new configuration of the drones in step i+1. It shows the effect of the drone movements on the formation of the clusters.

In Figures 3.8, r is the radius of the safe area. In this example, drone d_6 suddenly finds an obstacle and our algorithm first recomputes the routes of all drones in the safe area (d_2, d_3, d_4, d_7, d_9, and d_A) and then it recomputes the routes for other drones in the close proximity of the safe area (d_1, d_5, and d_8).

Let $Conf_i = \{Cl^i_1, Cl^i_2, \ldots, Cl^i_c\}$ be the configuration of the drones in the ith step, where $Cl^i_c = \{d_j, d_v, \ldots, d_q\}$ and $Conf_i = \bigcup_{j=1}^{c} Cl^i_j = D$, where $D = \{d_1, d_2, ..., d_{nd}\}$ is the set of all drones. When drone d_q detects an obstacle, Algorithm 1 computes $Safe_{d_q}$ (Line 15), which represents the set of all drones around d_q with a distance less than r. In the next step, the algorithm computes the set of all clusters that have common members with $Safe_{d_q}$, depicted as $Layer2_{d_q}$ (Line 16). Then, the algorithm computes new paths for all drones in $Safe_{d_q}$ and $Layer2_{d_q}$ (Line 17-18). $Safe_{d_q}$ and

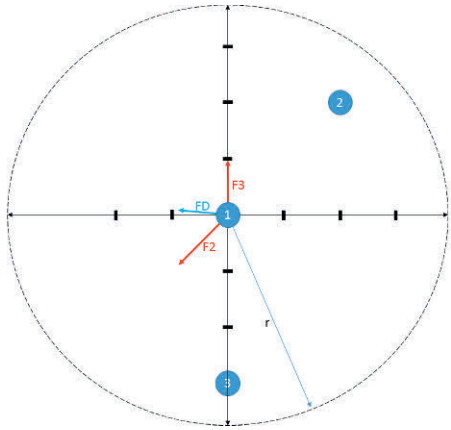


Figure 3.4: First part of drone reflex computation

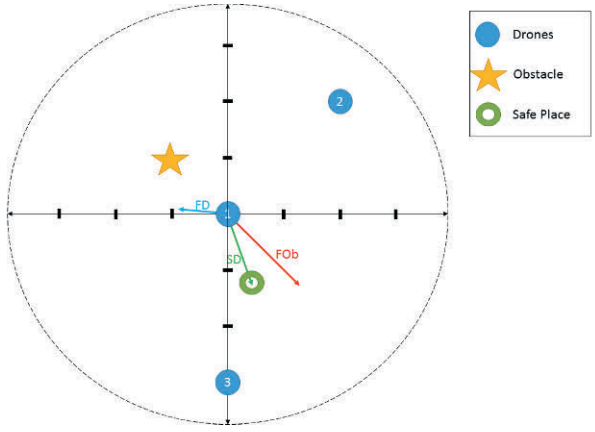


Figure 3.5: Second part of drone reflex computation

$Layer2_{d_q}$ are subsets of D :

$$Layer2_{d_q} \cup Safe_{d_q} \subseteq D \tag{3.8}$$

3.3 Safety-Aware Routing Planning and Run-Time Safety Monitoring

In this section, we present the implementation of the dynamic system architecture specified and verified in Section 3.4. We present both the algorithm for route planning and the overall control algorithm. Let us start by discussing the algorithm for swarm routing planning. As we have previously discussed, the flight area including the positions of the drones, can be represented by the set of locations *AREA*. We

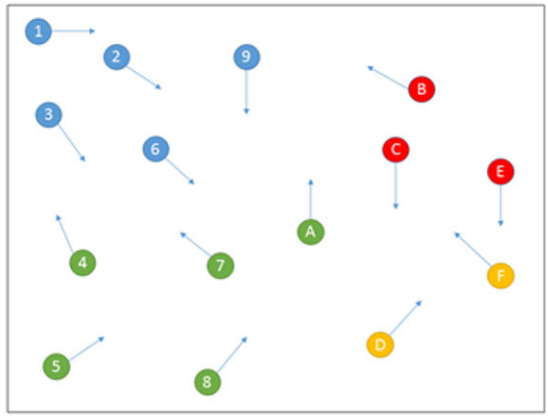


Figure 3.6: Step 1 of the drone clustering computation

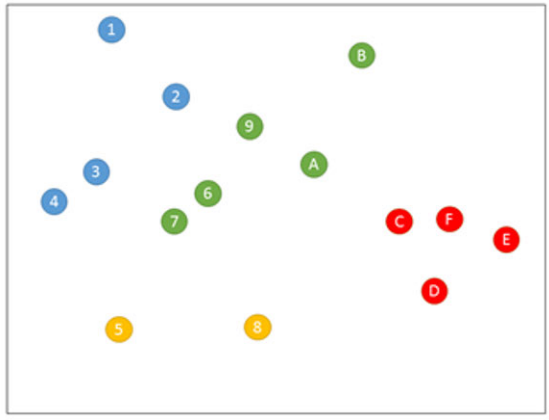


Figure 3.7: Step 2 of the drone clustering computation

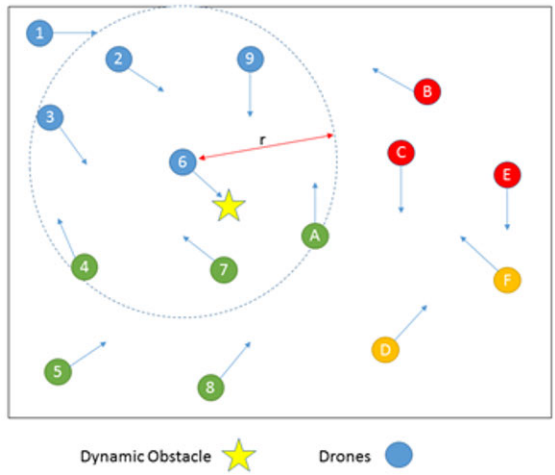


Figure 3.8: Step 3 of the drone clustering computation

assume that the entire flight zone is represented by a grid, i.e., the distances between a pair of neighboring locations are the same, as shown in Fig.3.10(a). The initial and the destination positions are known for all drones. The drones move from a location to location. Our goal is to find an optimal routing, where routing is defined as a union of each individual drone route, i.e., routing represents a plan of a mission for all drones.

We give an ID to each swarm routing and define the set of phenotypes as a set of routing IDs. To explain the principle of defining a chromosome, let us consider an example shown in Fig. 3.10(b). For the drone $d1$ the shortest path from the initial location to the destination is a sequence $\langle 20,19,18,17,16,11,6\rangle$, correspondingly the shortest paths for the drone $d2$ is $\langle 21,22,17,12,7,2,3\rangle$ and for $d3$ $\langle 11,12,13,14,9,10,15\rangle$. We note that the path of each drone can be succinctly represented by a "turning" point – we call it a middle point, which would be 16 for $d1$, 12 for $d2$, and 9 for $d3$. Hence, a chromosome representing such a routing can be defined as a triple $\langle\langle 16,12,9\rangle\rangle$. In general, for n drones a chromosome is an n-tuple consisting of the middle points of the corresponding drones.

To ensure collision avoidance with the static objects, we should explicitly define the locations, which are occupied by the obstacles as well as store the locations, which could be occupied by the dynamically appearing objects. Our route planning starts by generating all shortest paths between each pair of locations in our grid and storing them in a database. The database of the shortest paths is then used to compose the routes of the individual drones as a concatenation of the route from the current to the middle point and from the middle point to the final destination. The shortest routes are computed using the algorithm proposed by the Dijkstra [65]. For each given source node in the graph, the algorithm finds the shortest path between that node and all other nodes. As an input to our implementation of the Dijkstra's

algorithm, we define the adjacency matrix of the flight zone *AREA* with a explicit representation of the obstacles. The pseudocode of the Dijkstra's algorithm is given next:

$dist[s] \longleftarrow 0$
For All $v \in V - \{s\}$ **Do**
    $dist[v] \longleftarrow \infty$
$S \longleftarrow \emptyset$
$Q \longleftarrow V$
While $Q \neq \emptyset$ **Do**
  $\{u \longleftarrow mindistance(Q, dist)$
  For All $v \in neighbors[u]$ **Do**
    If $dist[v] > dist[u] + w(u,v)$ **Then**
      $d[v] \longleftarrow d[u] + w(u,v)$
  Return dist
  $\}$

The algorithm starts by initializing the vector of distances from the given node to the rest of the nodes in the graph. If there is no direct connection between the nodes, the distance between them is initially defined as infinity. The distance between each pair of node is computed as a sum of distances between all the nodes, which the path visits. After a new path is found, its distance is compared with the distance of the previously found path and the minimum of two is taken as the current minimal distance. The algorithm terminates after all possible paths in the graph have been explored.

Now we should define the fitness function to evaluate the fitness of each country (chromosome). As we discussed in section 2.1, our goal is to devise an algorithms that optimizes the safety/performance ratio. To achieve this, while evaluating fitness of each swarm routing, we should not only evaluate the corresponding path lengths, but also the number of cross points between all drone and dynamic object trajectories as well as the time gap associated with them. The first argument of our fitness function is the distance metric:

$$Distance\ Metric \ = \ \sum_{i=1}^{nd} Distance_{Current_i \rightarrow Middle_i} + Distance_{Middle_i \rightarrow Destination_i}$$

It defines the total length of the drone routes according to the given routing. For instance, for our example in Fig.3.10(b) the distance metric of the routing defined by the chromosome $\langle 16, 12, 9 \rangle$ is the sum of the lengths of the drone paths: 6+6+ 6= 18. The second argument defines the number of cross points associated with the given routing. For our example, the number of cross points is 3: in the location 17 between the routes 1 and 2, in the location 12 between the routes 2 and 3, and in the location 11 between the routes 1 and 3 correspondingly. The third argument of the fitness function is the safety level of the time gap at the cross point. We introduce

three safety levels: 0 if there is no cross points, 1 if the time gap at the cross point is above the safety threshold, and 2 if the time gap is below the threshold. For instance, for our example at Fig.3.10(b) the time gap at cross point 17 is 1, because the drones arrive at that point at times 3 and 2, the time gap for the cross point 12 is 2, because the drones arrive there at times 3 and 1, and for the cross point 11 it is 5. As a matter of illustration, we can assume that the time gaps below threshold 2 are classified as level 2, while the time gaps at and above threshold 2 as level 1. Hence, the cross point 17 obtains level 2 , while the cross points 11 and 12 the level 1. We define our route optimization task as a minimization problem with the following fitness function:

$$Fitness\,Function \;=\; Distance\,Metric + \alpha \times Number\,of\,CrossPoint \;+\; \beta \times Level$$

Here α and β are the weight coefficients defined as follows:

$$1 \;\leq \alpha \leq \frac{nd}{2} \qquad and \qquad 1 \leq \beta \leq \sqrt{np} \times nd$$

where *nd* is the number of drones and *np* is the total number of points. These values allow us to adapt the fitness function evaluation based on the level of complexity of the flying zone and the number of drones. For our example in Fig. 3.10(b), the value of the fitness function is computed as follows: $18 \;+\; 1{,}5 \times 3 + 5 \times 3 = 37.5$. The evaluation of the fitness function for the initial population is shown in Fig.3.9.

Current Position	Initial Population	Total Distance	# Cross Point	Critical Level	Fitness	Tournament Number	Mating Pool	r_c	Offspring after Crossover	r_m	Offspring after Mutation
20 21 11	17|24| 13	18	4	1	39	1, 3	17|24|13	0.6	16|22|16	0.2	17|24|13
	09|17| 18	20	4	2	56	2, 4	16|12|19		17|14|18	0.6	17|14|18
	13|09|23	22	5	1	44.5	1, 4	16|12|19	0.9	16|12|19	0.1	13|09|23
	16|12|90	18	3	1	37.5	2, 3	13|09|23		13|09|23	0.4	16|12|09

Total Distance	# Hot Point	Critical Level	Fitness	Next Generation	Best	Current Position	Total Distance	# Cross Point	Critical Level	Fitness	Tournament Number
18	6	2	57	17|24|13	16|12|09	18 22 12	14	1	1	30.5	2, 4
20	3	1	39.5	17|14|18			17	2	2	50	1, 3
22	4	2	58	13|09|23			18	4	1	39	1, 4
18	2	1	44.5	16|12|09			14	2	2	47	3, 2

Mating Pool	r_c	Offspring after Crossover	r_m	Offspring after Mutation	Total Distance	# Cross Point	Critical Level	Fitness	Next Generation	Best
16|12|09	0.5	17|09|09	0.1	17|09|11	16	4	2	52	17|24|13	17|24|13
17|24|13		16|21|14	0.3	16|21|14	14	1	1	30.5	16|21|14	
17|24|13	0.7	18|09|23	0.4	18|09|23	18	3	1	37.5	18|09|23	
13|09|23		12|24|13	0.8	12|24|13	14	2	1	32	12|24|09	

Figure 3.9: An example of the two iterations of algorithm

In our large scale experiments, after evaluating the fitness values of the initial population, we have chosen the imperialists – the countries with the fitness function values smaller than a certain threshold – and the colonies – the other countries. Due to a very small size of the population in our example, we skip this step and explicitly

pairwise compare the fitness values. The chromosomes with the lowest fitness values are chosen for cross over and mutation, as shown in the *Tournament Number* and *Mating Pool* columns in Fig.3.9. The next column defines the probabilities of crossover (*rc*) while the results of applying the crossover operator are shown in the *Offspring after Crossover* column. In a similar way, we define the probabilities of mutation. The *Offspring* after *Mutation* column shows the results of the mutation operator applied to the offsprings. Next, we calculate the fitness function for the mutated offsprings. To produce a new generation, from the initial population and the pool of mutated offsprings, we chose the chromosomes with the lowest values of the fitness function. After that we start the next iteration of the algorithm with the new generation as the current population. After several iterations of the algorithm, we find the routing that achieves our goal of minimizing the distance of travelling and associated danger, i.e., maximizes safety.

Now let us present the dynamic component of our approach. The pseudocode of the entire approach is shown in Fig.3.11.

As we discussed earlier, the mission planning allows us to maximize safety with respect to the predicted hazards. To monitor safety and ensure optimality of the swarm movement, we propose to augment the static part with the dynamic run-time routing recalculation as well as the mechanisms of controlled (i.e., not autonomous) collision avoidance.

Let us illustrate the first scenario. In Fig.3.10(c) we present a snapshot of drone positions after one unit of time has elapsed. Drone 2 and Drone 3 have moved according to the planned routes with the planned speed. However, due to some internal problems, Drone 1 moved twice as fast as it was supposed to. If Drone 1 regains the planned speed and the initial routing is not changed then Drone 1 and Drone 2 will collide in the location 17. Hence, we should invoke the route recalculation and change the routing. This goal is achieved using our proposed algorithm. As shown in Fig.3.10(c), the new routing avoids the crosspoint 17 by

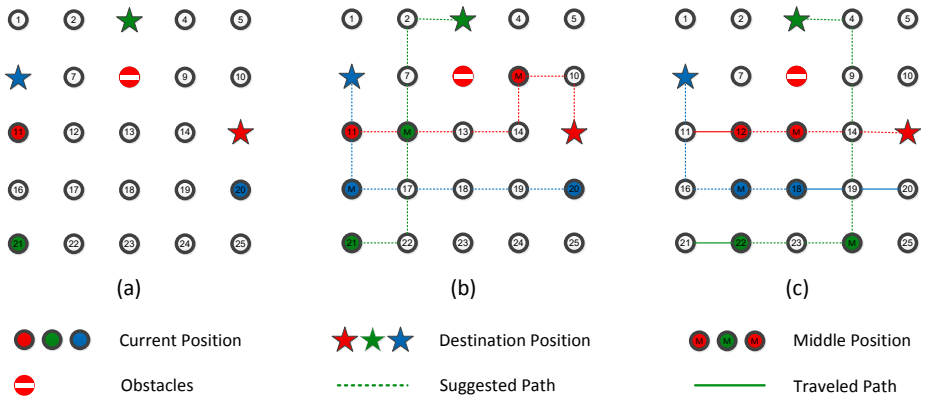


Figure 3.10: An illustrative example of drone route planning

rerouting Drone 2 to the route $\langle 22, 23, 2429, 24, 9, 4, 3 \rangle$ and finding a shorter path for Drone 3 $\langle 12, 13, 14, 15 \rangle$.

```
1. BEGIN
//**Offline Part
2.   Call Dijkstra'Algorithm to Compute the Shortest Path between all nodes
3.   Read Current Position of all Drones and Dynamic Obstacles
3.   Call Generate Countries
4.   Call Evaluation Operation
5.   Select the Best Routes
//**Online part
6.   While (all Drones arrived to their Destinations) Do
7.       Begin
8.           IF (# Cross point (Best Routes) == 0)     //**No cross point and only monitoring
9.              Begin
10.                While (the Best routes and current positions are match) Do only monitoring
11.                Call Evaluation Operation
12.             End
13.          Else IF (the Dangerous level==2)          //**Emergency time
14.                Begin
15.                   Run Critical Navigation Instructions
16.                   Go to 8.
17.                End
18.             Else
19.                Begin                              //**Finding the better routes
20.                   Call Assimilation and Revolution Operations
21.                   Call Evaluation Operation
22.                   Run Competition Operation
23.                   Go to 8.
24.                End
25.      End
26.  END.
```

Figure 3.11: The pseudocode of the overall algorithm

However, if a collision is predicted with a dynamically appearing object in a close proximity to drones, we need to activate the controlled collision avoidance mode and in a step-wise manner resolve the dangerous situation. If a collision is predicted between the drones then the priority to move to the next position is given to the drone which is closer to the cross point. Then after the safe time gap, the next drone moves to the next position and the situation is reassessed. After the collision danger is no longer present, the routing is recomputed and the autonomous flying mode is resumed.

In case of a danger of collision with a dynamically appearing a the control is also performed in a step wise way. In this case, however, each proposed move is first verified to be safe with respect to the on-line sensor readings.

3.4 Safety Constrains in Route Planning and Mission Execution

In this section, we use the Event-B framework to formalise the safety motion requirements **Req1-Req3.** We start our modelling by proposing a rigorous definition of the concepts introduced in section 2.1.

The swarm is represented by a finite non-empty set of drones *SWARM*. It can be seen as a set that contains the ids of all drones in the swarm. Without loss of generality, we assume that the goal of the mission is to ensure that each drone reaches a certain location. The initial and final locations of each drone are known in advance. The locations that the drones can fly over constitute the flight zone. For each mission we can define the corresponding flight zone, which can be represented by a finite set of locations – a non-empty finite set *AREA*.

We assume that the interval $[0..maxtime]$ is the interval covering the entire duration of a mission. The maximal duration of a mission is modelled as the pre-defined constant *maxtime* in the swarm to reach the required destinations.

Since the terrain of the flight zone is known, we can explicitly define the obstacles that should be avoided in the mission planning. Such "no fly" locations – mountains, tall constructions, etc. – occupy certain locations in our flight zone. Assume that there are *m* obstacles located in the flight zone. All the obstacles can be represented by a subset of locations that they occupy, i.e., defined as the following relation in the model context:

$$\{Obs \in 1..m\} \longleftrightarrow AREA \tag{3.9}$$

We use the term *swarm routing* to represent the union of all individual drone routes, i.e., swarm routing is essentially the mission plan. A route of each drone is defined as a sequence that maps time in the $[0..maxtime]$ interval to a location in the fly zone. The constant

$$\{Routes\_Init \in \{SWARM \times 0..maxtime\}\} \longrightarrow AREA \tag{3.10}$$

designates the initial routing generated by the planning algorithm. To ensure that the initial route planning avoids static obstacles, i.e., **Req1** is satisfied before the swarm is deployed, we formulate the following axiom:

$$ran(Routes\_Init) \cap \mathsf{ran}(Obs) = \emptyset, \tag{3.11}$$

where ran denotes a range of a function.

To explicitly model the motion safety requirements, we introduce an abstract constant total function $\{update\_Routes$ defined as

$$update\_Routes \in \{SWARM \times 0..maxtime\}\} \longrightarrow AREA.$$

It is an abstraction used to represent the outcomes of the route planning algorithm. Specifically, for every drone and time moment it returns the position where a drone has to move. We intentionally abstracted away from the actual implementation of this function since for our modelling it is enough to have it in such a form.

We introduce the constant *DO_Traj_Init* to model the system knowledge about dynamically appearing objects. It represents apriori information about possibility

of dynamic object appearance (e.g., the scheduled flights). The constant type is defined as the following partial function {

$$DO\_Traj\_Init \in \{DO \times 0..maxtime\} \pfun \mathbb{P}(AREA),$$

where *DO* is the set of dynamic objects. The function result is a set of area locations where the corresponding object can appear.

The dynamic architecture of the system follows the MAPE pattern. It is modelled by the machine SwarmOfDrones. The outline of the specification is given in Fig. 3.12.

Machine SwarmOfDrones
Sees SwarmOfDrones_c
Variables $mode, time, safety\_level,\ CurLoc,\ Status, Routes, DO\_Traj$
Invariants
$time \in 0..maxtime \land mode \in MODES \land safety\_level \in \{0,1,2\} \land$
$CurLoc \in SWARM \times 0..time \longrightarrow AREA \land$
$Status \in SWARM \longrightarrow STATUSES \land$
$Routes \in SWARM \times 0..maxtime \longrightarrow AREA \land$

$DO\_Traj \in DO \times 0..maxtime \pfun AREA \land$
$safety\_level = 2 \Rightarrow mode{=}CONTR \land$
$\mathsf{ran}(Routes) \cap \mathsf{ran}(Obs) = \emptyset \land$
$(mode{=}CONTR \land safety\_level \neq 2 \Rightarrow$
$(\forall d1, d2 \cdot d1 \in SWARM \land d2 \in SWARM \Rightarrow CurLoc(d1 \mapsto time) \neq CurLoc(d2 \mapsto time))) \land$
$(\forall dr, do \cdot dr \in SWARM \land do \in DO \land do \mapsto time \in dom(DO\_Traj) \Rightarrow CurLoc(dr \mapsto time) \notin DO\_Traj(do \mapsto time))$
Events
Initialisation ...
autonomous_moving ...
monitoring ...
recalculations ...
coordinated_moving ...
dynamic_object ...

Figure 3.12: Specification of SwarmOfDrones

The system operates in two modes: autonomous and controlled. The variable *mode*, which takes either value *AUTONOM* or *CONTR* designates the current system mode. The initial mode is autonomous. The variable *CurLoc* designates the current locations of all drones in the swarm. As we discussed previously, to guarantee motion safety, we should ensure that all deviations from the planned routes are detected and the corresponding safety precaution measures are implemented. To model the state of the drones, we introduce a variable *Status*. It is defined as a function $Status \in SWARM \longrightarrow STATUSES$, where *STATUSES* is a set consisting of the constants *OK* and *DEV* representing correspondingly the nominal and abnormal drone behaviour. In the system implementation, the decision about the drone status is made on the basis of the analysis of the currently received telemetry data and the mission plan.

The variable *Routes*

$$Routes \in SWARM \times 0..maxtime \longrightarrow AREA$$

represents the current swarm routing, i.e., the currently executed mission plan (i.e., the position of a drone at a particular time).

The variable *DO_Traj* models the dynamically appearing object and is defined similarly to the *DO_Traj_Init* constant.

The dynamic behaviour of the system modelled by the machine SwarmOfDrones is graphically represented in Fig. 3.13. Each rectangle corresponds to the specification event.

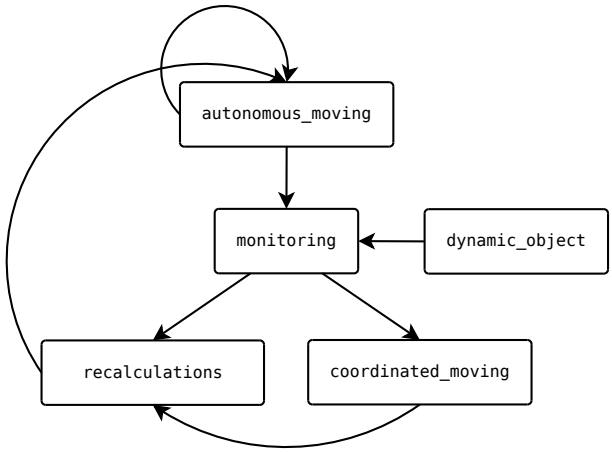


Figure 3.13: System behaviour

The event autonomous_moving shown below models autonomous flight of the swarm. The guard of the event ensures that the system is currently in the autonomous mode, the mission goal has not been achieved yet, and the safety of the planned mission is maintained. In the process of autonomous flying, the positions of the drones change. If they change according to the current mission plan then the status of the drones remains nominal. Otherwise, the deviations are detected and the statuses of the corresponding drones is altered.

The event dynamic_object models a non-deterministic appearance of an object in the flight zone. It results in the corresponding changes in the value of the variable DO_Traj. There is an implicit assumption on which we rely while formulating the requirements of motion safety – a dynamically appearing object cannot suddenly appear in the location that is currently occupied by some drone of the swarm. This is a reasonable assumption, which, essentially, means that the swarm is not protected against objects hitting drones from a blind zone. Formal modelling helped us to make this assumption explicit.

To explain the main mechanism of ensuring motion safety implemented by the remaining specification events, let us consider the following scenario. Assume that

autonomous_moving $\widehat{=}$
  any *new_CurLoc*
  where $mode{=}AUTONOM \wedge time \in 0..maxtime\text{-}1 \wedge$
    $safety\_level \neq 2 \wedge Status[SWARM] = \{OK\} \wedge$
    $new\_CurLoc \in SWARM \times 0..time+1 \longrightarrow AREA \wedge$
    $(\forall dr,tt \cdot dr \in SWARM \wedge tt \in 0..time \Rightarrow new\_CurLoc(dr \mapsto tt) = CurLoc(dr \mapsto tt)) \wedge$
    $(\forall dr \cdot dr \in SWARM \Rightarrow new\_CurLoc(dr \mapsto time+1) = update\_CurLoc(dr) \vee$
      $new\_CurLoc(dr \mapsto time{+}1) = Routes(dr \mapsto time{+}1))$
  then $time{:=}time+1$
    $Status{:=}update\_Status$
    $CurLoc{:=}new\_CurLoc$
  end

dynamic_object $\widehat{=}$
  any do, aa
  where $aa \subseteq AREA \wedge do \in DO \wedge$
    $\forall dr \cdot dr \in SWARM \Rightarrow CurLoc(dr \mapsto time) \notin aa$
  then $DO\_Traj(do \mapsto time){:=}aa$

as a result of the route planning, it is established that the drones *d1* and *d2* cross at the location *l*. Since the speed of each drone and distance from the initial point till *l* are known, we calculate that *d1* will reach the point *l* at the time *t1* and *d2* at the time *t2* and $t1 - t2 > \Delta$, were Δ is a constant defining the time gap derived from the predefined safe proximity distance. However, while the mission is in progress, due to some internal failure, *d1* moves slower than expected and can reach the point *l* at the time *t1'*, such that $t1 - t2 < \Delta$, i.e., the likelihood of collision increases. The similar reasoning can be also applied to the situation, when a dynamic object appears at the flight zone.

In our specification, we introduce a variable *safety_level* that can take values 0, 1 or 2. The zero value designated a situation when the current swarm routing does not contain any cross points, i.e., the routes of the drones do not intersect. The value 1 represents two possible situations. The first one corresponds to the fact that there are cross points in the current routing but the time gap is above the safety threshold for each of them. The second situation describes the fact that a dynamic object has appeared in the flight zone but the risk of collision is low. If the variable *safety_level* assigned the value 2 then the risk of collision is high, i.e., either according to the current routing drones can collide or the dynamically appeared object is close to some drones.

Obviously, the system should monitor the safety level and take appropriate actions, as modelled by the event monitoring defined below:

If, as a result of safety level monitoring, it is established that the safety level obtains value 2 then the system ceases the autonomous mode and activates the controlled flight mode. The system behaviour in the controlled mode is modelled by the event coordinated_moving.

In the coordinated mode the drones in a step-wise coordinated manner perform

monitoring $\widehat{=}$
any $dr, calc, update\_mode, tt, do$
where $mode$=$AUTONOM \wedge$
$((Status(dr) = DEV) \vee (tt \in time+1..maxtime \wedge Routes(dr \mapsto tt) \in DO\_Traj(do \mapsto time))) \wedge$
$calc \in \{1,2\} \wedge update\_mode \in MODES \wedge$
$((calc = 2 \wedge update\_mode$=$CONTR) \vee (calc = 1 \wedge update\_mode$=$AUTONOM))$
then $safety\_level$:=$calc$
$mode$:=$update\_mode$

coordinated_moving $\widehat{=}$
any new_CurLoc
where $safety\_level = 2 \wedge$
$time \in 0..maxtime-1 \wedge$
$new\_CurLoc \in SWARM \times 0..time+1 \longrightarrow AREA \wedge$
$(\forall dr, tt \cdot dr \in SWARM \wedge tt \in 0..time \Rightarrow new\_CurLoc(dr \mapsto tt) = CurLoc(dr \mapsto tt)) \wedge$
$\forall dr \cdot dr \in SWARM \Rightarrow new\_CurLoc(dr \mapsto time+1) = update\_Routes(dr \mapsto time+1) \wedge$
$\forall d1, d2 \cdot d1 \in SWARM \wedge d2 \in SWARM \Rightarrow new\_CurLoc(d1 \mapsto time+1) \neq new\_CurLoc(d2 \mapsto time+1) \wedge$
$\forall dr, do, tt \cdot dr \in SWARM \wedge do \in DO \wedge tt \in time+1..maxtime \Rightarrow CurLoc(dr \mapsto time+1) \notin DO\_Traj(do \mapsto tt)$
then $CurLoc$:=new_CurLoc
$Status$:=$SWARM \times \{OK\}$
$time$:=$time+1$
$safety\_level :\in \{0,1\}$

a safe manoeuvre. At each step, the coordinator assesses the situation and plans the next safe move.

The event recalculations is enabled when an acceptable level of safety is maintained or re-established in the system:

recalculations $\widehat{=}$
where $(safety\_level = 1 \wedge mode$=$AUTONOM) \vee (safety\_level \neq 2 \wedge mode$=$CONTR)$
then $Routes$:=$update\_Routes$
$Status$:=$SWARM \times \{OK\}$
$mode$:=$AUTONOM$
$safety\_level :\in \{0,1\}$

It maintains or resumes the autonomous mode and models the outcome of swarm routing re-computation. Such a swarm routing recalculation aims at increasing safety and minimising the travelling distance.

In the proposed formal model, we have formalised the safety motion requirement and proved that they are preserved by the system architecture. The requirement **Req1** – collision avoidance with the static objects – is defined by the following axiom

$$ran(Routes\_Init) \cap ran(Obs) = \emptyset$$

and the invariant

$$ran(Routes) \cap ran(Obs) = \emptyset$$

.

The requirement **Req2** – drones do not collide with each other – is defined by the axiom that ensures safety of the initial mission planning

$$\forall d1, d2 \cdot d1 \in SWARM \wedge d2 \in SWARM \Rightarrow CurLoc\_Init(d1 \mapsto 0) \neq CurLoc\_Init(d2 \mapsto 0))$$

and the invariant that guarantees that safety is preserved during the mission execution

$$mode{=}CONTR \wedge safety\_level \neq 2 \Rightarrow (\forall d1, d2 \cdot d1 \in SWARM \wedge d2 \in SWARM \Rightarrow CurLoc(d1 \mapsto time) \neq CurLoc(d2 \mapsto time))$$

The guards of the specification events ensure that if the acceptable safety level is exceeded then the system switches the autonomous function and operates in the controlled mode.

Finally, **Req3** – collision avoidance with the dynamically appearing objects is also formulated via the corresponding axiom

$$(\forall dr, do \cdot dr \in SWARM \wedge do \in DO \wedge do \mapsto 0 \in dom(DO\_Traj\_Init) \; CurLoc\_Init(dr \mapsto 0) \notin DO\_Traj\_Init(do \mapsto 0))$$

and the invariant:

$$(\forall dr, do \cdot dr \in SWARM \wedge do \in DO \wedge do \mapsto time \in dom(DO\_Traj) \Rightarrow CurLoc(dr \mapsto time) \notin DO\_Traj(do \mapsto time))$$

The guards of the events ensure that the controlled mode is activated if a dynamic objects appears in a close proximity to the drones.

The model has been formally verified using the Rodin platform. The platform has automatically generated the corresponding proof obligations and facilitated their automatic and interactive proving. Therefore, we have demonstrated by proofs, that the proposed system specification preserves the defined motion safety requirements **Req 1-Req 3.**

In Section 2.1, we have discussed that the safety-related requirements are necessary but not sufficient to guarantee dependability of a swarm system. To achieve that, we also need to bound the travel distance overhead and achieve high performance of routing calculation. To address this issue, we have developed a highly efficient evolutionary algorithm for safety-aware routing planning. The algorithm is used to compute the initial mission planning as well as dynamically recalculate the swarm routing at run-time.

The algorithm has been verified in a number of scenarios. The algorithm has been implemented using both shared memory and message passing styles. Our experimental configuration consisted of four processors (Intel Core i5-45705, 2.90

GHz (64-bit), 24GB) connected in a ring topology. The algorithm has demonstrated a remarkable performance: for a grid 100 $\times$ 100 points and eight drones, we algorithm was able to generate approx 70 routing alternatives per second. Hence, the routing calculation time is negligible comparing to the communication time. Therefore, we believe that our solution meets the requirement **Req 5**.

To validate whether the proposed solution also satisfies the requirement **Req 4** – the travelling distance overhead added to achieve safety is bounded and acceptable – let us present the results of two extreme benchmarks. The first benchmark focuses on resolving the problem of the high number of potential cross points, as shown in Fig.3.14(top). The drones should fly in the opposite directions and hence, there is a very high risk of collision between each other. Our algorithm has successfully and efficiently managed to solve the collision avoidance problem: no collisions occurred and the travel distance has increased only by $5,5\%$ as shown in Table 3.1.

	Number of Obstacles	Number of Drones	Number of Points	Shortest Distance without Safety	Shortest Distance with Safety	Difference Distance
Benchmark 1	0	6	64	72	76	5.5%
Benchmark 2	8	4	100	68	73	7.3%

Table 3.1: Benchmarking Statistics

The second benchmark aims at validating the algorithms under a challenging flying zone topology: the static objects (e.g., mountains) are densely located and leave only narrow curved corridors for flying as shown in Fig.3.14(bottom). The algorithm has succeeded in finding a safe and efficient routing – the resulting increase in the travel distance is only $7,3\%$. Therefore, we argue that the requirement **Req4** is fulfilled as well.

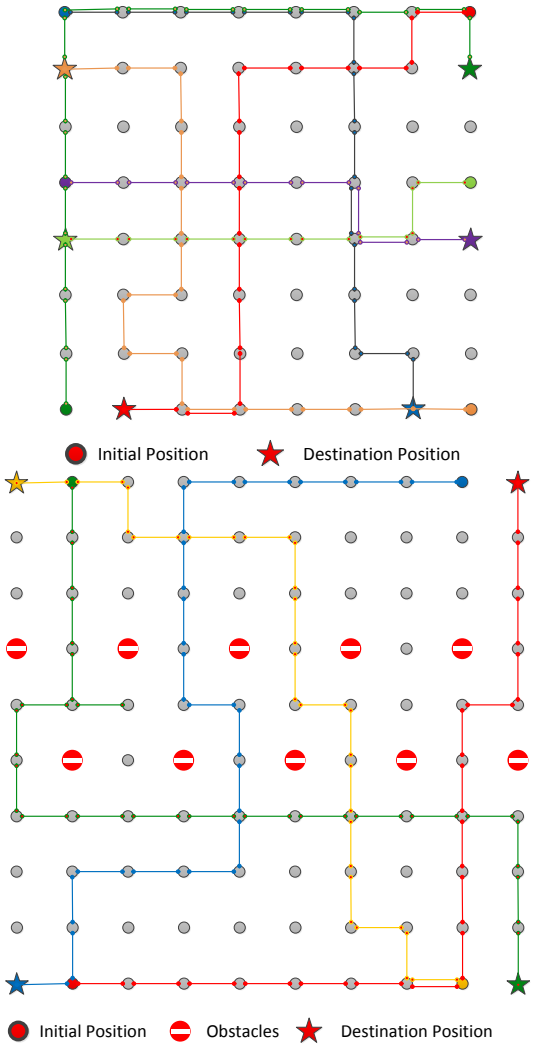


Figure 3.14: Benchmarking illustration

Chapter 4

Experimental Results

In this chapter, the experimental results and validating the thesis contributions are presented. Only some of the results from the original papers have been selected, and they are presented based on the order of the published papers.

4.1 DIANA-Safe Navigation

In this section, we present some important implementation details and experimental results. We assume that the flying zone *AREA* is represented by a three-dimensional grid. However, for a simpler illustration, we describe the main steps with a two-dimensional grid shown in Figure 3.10 (a). Our goal is to find an efficient, collision-free route for each drone from the initial start location of the drone to the destination location.

4.1.1 Implementation Details

We use the example shown in Figure 3.10 (b) to explain the principles used for defining the countries (chromosomes) in the MICAP algorithm. For the drone d_1, initially situated at the location 20, the shortest path from the initial location to the destination is a sequence $\langle 20, 19, 18, 17, 16, 11, 6 \rangle$ [58]. We note that the path of each drone can be succinctly represented by its *turning* point – we call it the *middle point*, which would be 16 for d_1. Therefore, the proposed approach uses middle points to optimize the drone routes. By using different values for the drone's middle point, it is possible to generate different routes for the drone. Thus, it allows to explore different alternatives in the search space. Let us assume that the middle points for the two other drones in Figure 3.10 (b) are 12 and 9. A country representing all the drone routes for the swarm in Figure 3.10 can be then defined as a triple $\langle\langle 16, 12, 9 \rangle\rangle$. In general, for *nd* drones a country is an *nd*-tuple consisting of the middle points of the corresponding drones.

In the offline, planned part of the proposed approach, all the locations occupied

by static obstacles and the initial locations of the moving obstacles are marked as occupied or unsafe. The offline route planning module then generates all the shortest paths between each pair of locations in the flying zone *AREA*, while avoiding all locations marked as occupied or unsafe, and stores the generated routes in a database, which is then used to compose the routes for the individual drones as a concatenation of the shortest routes from initial locations to the middle points and from the middle points to the final destinations. The shortest routes are computed by using the algorithm proposed by Dijkstra [52].

The fitness function to evaluate the fitness of each country optimizes the safety/performance ratio. The first argument of our fitness function is the distance metric

$$\begin{aligned} Distance\ Metric\ = \sum_{i=1}^{nd} & Distance_{Current_i \rightarrow Middle_i} \\ & + Distance_{Middle_i \rightarrow Destination_i} \end{aligned}$$

It defines the total length of the drone routes according to the given solution [58]. For our example in Figure 3.10 (b), the distance metric of the routing defined by the country $\langle 16,12,9 \rangle$ is the sum of the lengths of the drone paths: 6+6+6=18. The second argument of the fitness function defines the number of cross points associated with the given solution. For our example in Figure 3.10 (b), the number of cross points is 3: in the location 17 between the routes 1 and 2, in the location 12 between the routes 2 and 3, and in the location 11 between the routes 1 and 3, correspondingly.

The third argument is the safety level of the time gap at the cross point [58]. We introduce three safety levels: 0 if there is no cross points, 1 if the time gap at the cross point is above the safety threshold, and 2 if the time gap is below the threshold. For our example in Figure 3.10 (b), the time gap at cross point 17 is 1, because the drones arrive at that point at the times 3 and 2, the time gap for the cross point 12 is 2, because the drones arrive there at the times 3 and 1, and for the cross point 11, it is 5. As a matter of illustration, we can assume that the time gaps below threshold 2 are classified as level 2, while the time gaps at and above threshold 2 as level 1. Hence, the cross point 17 obtains level 2, while the cross points 11 and 12 obtain level 1 each. The safety level of a complete routing solution for the swarm can then be computed by aggregating the individual safety levels of all cross points in the solution: 2+1+1=4. We define our route optimization task as a minimization problem with the following fitness function:

$$\begin{aligned} Fitness\ Function\ = \ & Distance\ Metric\ + \alpha \cdot \\ & Number\ of\ CrossPoint\ + \ \beta \cdot Level \end{aligned}$$

Here α and β are the weight coefficients defined as

$$1 \ \leq \alpha \leq \frac{nd}{2} \qquad 1 \leq \beta \leq \sqrt{np} \times nd$$

where *nd* is the number of drones and *np* is the total number of points. These values allow us to adapt the fitness function evaluation based on the level of complexity of the flight zone and the number of drones. For our example in Figure 3.10 (b), the value of the fitness function is computed as $18 + 1.5 \times 3 + 5 \times 4 = 42.5$.

We have implemented the proposed DIANA approach on using a shared memory model. We used the message passing interface (MPI) to parallelize the proposed algorithm and MPICH[1] to run the algorithm. To implement DIANA, we used four processors in a ring topology. Our algorithm was tested on Intel® Xeon® E5-1620 v3 @ 3.50 GHz processors with 16 GB memory and NVIDIA® GeForce® GTX 1080 graphics processing units.

4.1.2 Experiment Design and Setup

We present results from two benchmark implementations. Benchmark 1 is based on a small $100 \times 100 \times 20$ flying zone with 16 drones, 3 dynamic obstacles moving on straight lines from different starting positions, 2 dynamic obstacles moving randomly from different starting positions, and 8 unforeseen/unpredicted static obstacles. Benchmark 2 is based on a large $1000 \times 1000 \times 100$ flying zone with 250 drones, 10 dynamic obstacles moving on straight lines from different starting positions, 15 dynamic obstacles moving randomly from different starting positions, and 60 unpredicted static obstacles. Figure 4.1 and 4.2 depict the predicted routes of the moving obstacles in Benchmark 1 and 2, respectively. The experiment design is summarized in Table 4.1. We ran each benchmark first with one flight per drone and then with 1000 flights per drone to evaluate the performance of the proposed learning approach and to demonstrate how DIANA learns over time and uses this knowledge to generate safer and shorter drone routes.

We compare DIANA results with the results of the following six alternative approaches:

1. **Dynamic Autonomous Navigation Algorithm (DANA):** A baseline approach which is similar to DIANA except that it does not use learning and prediction.

2. **Dynamic Genetic Algorithm (DGA):** A well-known approach [66], which addresses a similar problem.

3. **Particle Swarm Optimization (PSO) based approach:** Sujit and Beard [35] PSO based path planning approach, which generates paths for a swarm of drones.

4. **Greedy heuristics and Genetic Algorithms (GGA) approach:** Silva Arantes et al. [36] approach, which uses greedy heuristics and genetic algorithms to generate and optimize paths for a drone under critical situations.

[1] https://www.mpich.org/

5. **Rapidly-Exploring Random Trees (RRT):** LaValle [67] sampling-based path planning algorithm.

6. **RRT*:** Karaman and Frazzoli [68]'s extension of RRT that allows to plan optimal paths.

In addition, we compare the results of the proposed DIANA approach with three additional variants of DIANA namely DIANA_S, DIANA_P, and DIANA_N. As described in Section 4.1.1, the proposed DIANA approach optimizes the safety/performance ratio. In contrast:

1. **DIANA_S** optimizes safety, but does not optimize performance.

2. **DIANA_P** optimizes performance, but does not optimize safety.

3. **DIANA_N** is similar to DIANA, but it does not implement the proposed clustering algorithm and the proposed drone reflexes approach.

4.1.3 Results and Analysis

Tables 4.2 to 4.6 present a comparison of the results of DIANA and its three variants with DGA, DANA, PSO, GGA, RRT, and RRT*. The comparison is based on

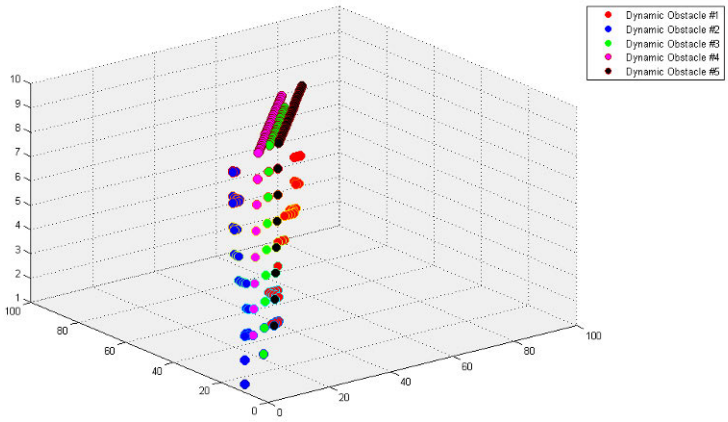


Figure 4.1: Predicted routes of the moving obstacles in Benchmark 1

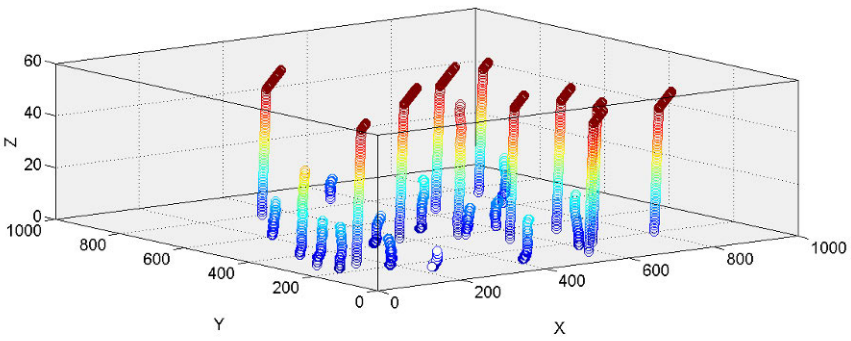


Figure 4.2: Predicted routes of the moving obstacles in Benchmark 2

the following six metrics. The best achieved results with respect to the evaluation criteria are highlighted in the tables using **bold** font.

1. *Total route length (TRL):* the total length of all drone routes measured as the number of steps in the drone routes. To be minimized to generate shorter routes.

2. *Minimum distance (MD):* the minimum distance between a drone and an obstacle. To be maximized to generate safer routes.

Table 4.1: Experiment design

Benchmark	Benchmark 1	Benchmark 2
Flying zone	100×100×20	1000×1000×100
Problem size	Small	Large
Number of drones	16	250
Number of unpredicted static obstacles	8	60
Number of moving obstacles	5	25
Number of flights per drone	1000	1000

3. *Frequency of route regeneration (FRR):* the number of times the drone routes are regenerated. To be minimized for reducing the re-computation overhead.

4. *Number of crashes (NC):* the number of drone collisions. To be minimized to generate safer routes.

5. *Length of the longest route (LLR):* the total number of steps in the generated longest route. To be minimized to generate shorter routes.

6. *Total time (TT):* total runtime of the algorithm in minutes. To be minimized to reduce the algorithm runtime.

Table 4.2 presents Benchmark 1 results with one flight per drone. The results show that DIANA_P generated the shortest routes and outperformed DANA, DGA, PSO, GGA, RRT, and RRT* in terms of TRL and LLR. On the other hand, DIANA_S produced the safest routes with the MD of 7. Moreover, DIANA_N minimized the FRR more efficiently than all other approaches. Finally, DIANA, DIANA_S, DANA, DGA, and RRT produced routes with NC of 0, while RRT* performed best in terms of TT. Table 4.3 presents Benchmark 1 results with 1000 flights per drone. It shows that DIANA_P generated the shortest routes in terms of TRL, DIANA_S produced the safest routes with an MD of 5, DIANA performed best in terms of FRR and TT, and RRT* generated the shortest routes in terms of LLR. Moreover, both DIANA and DIANA_S produced routes with NC of 0.

Table 4.2: Benchmark 1 results with one flight per drone

	DIANA	DIANA_S	DIANA_P	DIANA_N	DANA	DGA	PSO	GGA	RRT	RRT*
TRL	3448	4122	**3386**	3580	4057	4146	3745	3612	3608	3534
MD	5	**7**	0	0	4	2	0	0	1	0
FRR	14	23	12	**9**	22	21	19	17	16	16
NC	**0**	**0**	3	2	**0**	**0**	3	1	**0**	2
LLR	242	265	**234**	251	266	258	255	242	257	248
TT	1.9	2.1	2.0	2.2	2.7	3.0	2.4	2.4	2.0	**1.8**

Table 4.3: Benchmark 1 results with 1000 flights per drone

	DIANA	DIANA_S	DIANA_P	DIANA_N	DANA	DGA	PSO	GGA	RRT	RRT*
TRL	2657	4358	**2649**	3454	3925	4038	3506	3422	3508	3412
MD	4	**5**	0	0	0	0	0	0	0	0
FRR	**11245**	18246	11843	80146	19522	21010	17840	14531	16000	16000
NC	**0**	**0**	942	982	7	61	412	29	2119	2208
LLR	259	267	241	266	284	264	277	284	248	**236**
TT	**1137**	1643	1211	1391	1477	1874	1811	1707	1275	1359

Table 4.4 presents Benchmark 2 results with one flight per drone. The results show that DIANA_P performed best in terms of TRL and FRR, DIANA_S produced the safest routes in terms of MD, both DIANA and DIANA_S produced routes with the NC of 0, RRT* generated the shortest routes in terms of LLR, and DIANA

performed best in terms of TT. Table 4.5 presents Benchmark 2 results with 1000 flights per drone. It shows that DIANA outperformed DANA, DGA, PSO, GGA, RRT, and RRT* in terms of TRL, FRR, and TT. Moreover, both DIANA and DIANA_S produced the safest routes in terms of MD and NC, while DIANA_N generated the shortest routes in terms of LLR.

The results show that DIANA and its three variants DIANA_S, DIANA_P, and DIANA_N produced the safest and shortest drone routes for both benchmarks with one flight per drone and 1000 flights per drone. For Benchmark 1 with one flight per drone, DIANA and its variants performed best in all metrics except TT. Similarly, for Benchmark 1 with 1000 flights per drone and for Benchmark 2 with one flight per drone, DIANA and its variants performed best in all metrics except LLR. Finally, for Benchmark 2 with 1000 flights per drone, DIANA and its variants performed best in all six metrics. Other than DIANA and its three variants, RRT* also produced some good results in terms of TT and LLR for Benchmark 1 and 2, while DANA, DGA, and RRT generated some good results in terms of NC for Benchmark 1. Both PSO and GGA did not perform best with respect to any metric.

Table 4.4: Benchmark 2 results with one flight per drone

	DIANA	DIANA_S	DIANA_P	DIANA_N	DANA	DGA	PSO	GGA	RRT	RRT*
TRL	460891	642510	**448909**	464532	548450	528940	493521	484766	546287	464521
MD	9	**11**	0	0	7	0	0	0	0	0
FRR	286	621	**208**	383	736	524	539	412	250	250
NC	**0**	**0**	18	13	**0**	6	9	4	47	61
LLR	2440	2984	2670	2494	2674	2845	2511	2498	2670	**2418**
TT	**14.2**	16.8	15.8	14.8	18.3	20.1	19.5	16.3	15.8	14.7

Table 4.5: Benchmark 2 results with 1000 flights per drone

	DIANA	DIANA_S	DIANA_P	DIANA_N	DANA	DGA	PSO	GGA	RRT	RRT*
TRL	**278460**	584757	378541	438497	488975	497725	402587	422498	378279	286524
MD	**7**	**7**	0	0	0	0	0	0	0	0
FRR	**143527**	468244	282264	298445	664507	328068	462147	222652	250000	250000
NC	**0**	**0**	1430	1288	896	1128	1315	608	4105	3890
LLR	2561	2996	2798	**2468**	2740	2834	2604	2531	2567	2531
TT	**8491**	16441	14123	11761	17077	21491	16644	14106	11301	10171

Overall, DIANA generated the shortest drone routes in terms of aggregate TRL of all experiments, while RRT* performed second best and DIANA_P performed third best (as shown in Table 4.6). DIANA produced 29%, 28%, 17%, 18%, 20%, and 2% shorter drone routes than DANA, DGA, PSO, GGA, RRT, and RRT*, respectively. In terms of the aggregate LLR of all experiments, RRT* performed best, while DIANA_N performed second best and DIANA performed third best. Similarly, in terms of aggregate MD, DIANA_S generated the safest routes in all experiments, while DIANA performed second best and DANA performed third best. Moreover, both DIANA and DIANA_S generated collision-free routes in all experiments with aggregate NC of 0. DIANA also performed best in terms of aggregate FRR and aggregate TT of all experiments. Therefore, DIANA and its

three variants generated the safest and shortest drone routes while minimizing the computation overhead.

Tables 4.2 to 4.5 also present a comparison of the results of the proposed DIANA approach with its three variants. As expected, DIANA_P generated the shortest drone routes in most of the cases, but it did not prevent the drones from colliding with one another. On the other hand, DIANA_S generated longer, but the safest routes. It maximized the MD metric more efficiently than all of the other approaches. DIANA_N produced the best results with respect to FRR for Benchmark 1 with one flight per drone and with respect to the LLR metric for Benchmark 2 with 1000 flights per drone, but did not perform well in most of the cases. Moreover, since it does not implement the proposed drone reflexes approach, it could not prevent the drones from colliding into some unpredicted static obstacles.

Table 4.6: Aggregate results of all experiments

	DIANA	DIANA_S	DIANA_P	DIANA_N	DANA	DGA	PSO	GGA	RRT	RRT*
TRL	**745456**	1235747	833485	910063	1045407	1034849	903359	914298	931682	757991
MD	25	**30**	0	0	11	2	0	0	1	0
FRR	**155072**	487134	294327	378983	684787	349623	480545	237612	266266	266266
NC	**0**	**0**	2393	2285	903	1195	1739	642	6271	6161
LLR	5502	6512	5943	5479	5964	6201	5647	5555	5742	**5433**
TT	**9644**	18103	15352	13169	18575	23388	18477	15832	12594	11547

4.2 DIANA-Efficient Placement

Now, we will demonstrate the efficiency of our proposed method on three various case studies. Since this method targets dynamic positioning, we try to consider the search space as a dynamic space. In the first case study, we have a fixed search space according to Table 4.7 as mentioned earlier, each point has a specific weight which shows its importance to be covered. In this case study, we randomly assign weights to the points, collect the experimental results, and then compare the efficiency of our method with other methods.

In the second case study, we randomly change the number of drones. In other words, we simulate situations at which some drones fail due to some technical problems.

The third case study is a combination of the first and second ones: we dynamically change both the weights of the points and the number of drones. In the all case studies, we changed the weights, number of drones or both 10,000 times.

Our search space is 100*100 points and the maximum number of drones are 1000 and the minimum number is equal to 750. Each drone has the covering radius ($r = \sqrt{2}$). Also each drone can cover 9 points in our search spaces.

In first case study, we changed the weights 10,000 times, and in the second one, we changed the number of drones 10,000 times. Finally, in the third case study, we changed both the weights and the number of the drones 10,000 times.

Table 4.7: Experiment design on the same size of search space $= 100 \times 100$,the parameters are Number of Changing Weights (#NCW), Number of Drones (#ND), and Number of Changing Drones (#NCD).

	Weights	**#NCW**	**#ND**	**#NCD**
Case 1	Dynamically Change	1000	1000	1000
Case 2	No Change	0	750-1000	1000
Case 3	Dynamically Change	1000	750-1000	1000

Table 4.8: Case study 1 results (changed the weights 10,000 times)

	DIANA+	**DQN**	**NSGA-II**	**A(1)-A(3)**	**OD3P**	**CPlace**
#CP	87464826	85989478	77466577	77594538	73285761	86751565
TCW	165724276	157846858	135275719	141384679	117687594	160872332
APS	4356.27	9567.77	18946.64	7648.11	4967.25	6827.44
MPT	0.338	0.745	15.142	0.678	0.468	0.482
WPT	0.512	1.214	22.246	1.164	0.582	0.865

The comparison result of the three case studies are shown in Tables 4.8, 4.9, and 4.10. In these tables, five importance scales are used to compare the results:

- The number of covered points (#CP): this is equal to the total number of covered points during 10,000 changes.
- Total covered weights (TCW): this shows the quality of the coverage and the accuracy of the algorithm to choose the optimum points.
- Average processing speed (APS): this shows the average time needed to process a new configuration after every change.
- Minimum processing time (MPT): the minimum time required to determine a new configuration.
- Worst processing time (WPT): the maximum time required to determine a new configuration.

Figure 4.3 shows the efficiency of two decision-making components. In other words, it shows the ratio between the number of times the result of the evolutionary method is used and the number of times the result of the learning method is used.

Table 4.9: Case study 2 results (changed the number of drones 10,000 times)

	DIANA+	**DQN**	**NSGA-II**	**A(1)-A(3)**	**OD3P**	**CPlace**
#CP	79485378	72485764	61034672	71802525	70496622	76285437
TCW	148506792	137854682	116285710	11976853	109867512	142665448
APS	4427.24	9428.44	19678.57	7738.24	5021.34	6992.65
MPT	0.346	0.812	17.477	0.712	0.443	0.497
WPT	0.521	1.208	22.467	1.121	0.591	0.845

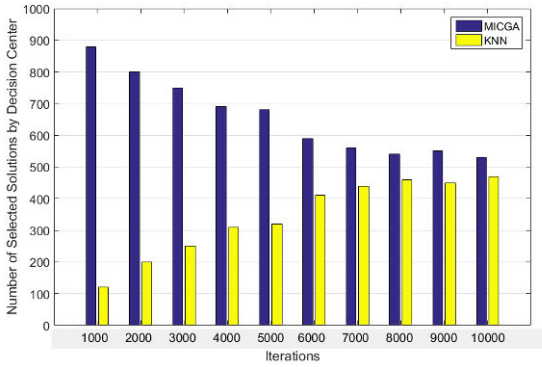
Table 4.10: Case study 3 results (changed both the weights and the number of the drones 10,000 times (10,000 for each))

	DIANA+	**DQN**	**NSGA-II**	**A(1)-A(3)**	**OD3P**	**CPlace**
#CP	78642892	70984573	59422984	70984522	68495764	77454265
TCW	146850332	129576483	98246765	120132546	98258612	143667522
APS	4419.37	9589.41	20119.47	7658.64	5234.21	7120.88
MPT	0.342	0.985	18.995	0.744	0.485	0.501
WPT	0.552	1.247	21.475	1.182	0.607	0.886

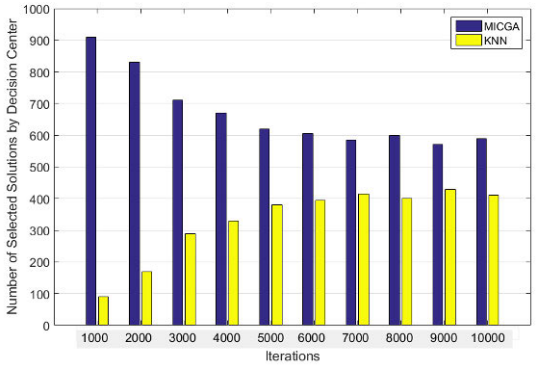
One of the issues that should be considered is that one of our methods is a machine-learning method and depends on a proper data set. The main question is that how this can perform in a dynamic environment. As mentioned earlier, we applied 10,000 times changes for each of the case studies. To investigate the efficiency of this algorithm, we divided the original set of cases into ten subsets of 1000 cases and examined the success rate of each decision-making method to find an acceptable solution. The bar charts in Figure 4.3 show this fact.

As can be seen in the bar charts, during the first 1000 changes, most of the solutions were provided by the MICGA, since the learning algorithm lacks a proper data set. For the following ranges, the efficiency of the learning algorithm has improved. But the more interesting point in these histograms is that both methods reach a relatively equal situation, which has been repeated in the last ranges and proves the necessity of both methods. In other words, the methods are efficiently supporting each other.

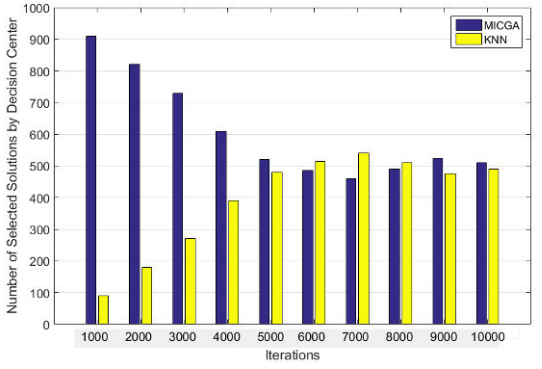
This improving trend shows that our evolutionary algorithm, in addition to generating the beneficial solutions, is doing the data set update task well. The histogram C, which shows that as the problems becomes more complex, there is more need for the cooperation between two methods.



(a)



(b)



(c)

Figure 4.3: Histograms to present the efficiency of the decision making components

To show the efficiency of our method, we compare it with five successful positioning methods for drones. The first method is deep Q-learning (DQN). In DQN, to handle the dynamic swarm topology and time-varying link condition, they have designed a deep Q-learning (DQN) model to determine two UAV nodes, and then use an optimization algorithm to locally fine-tune the position of UAV node to the overall network performance. The second method with which we have compared our results is a Non-Dominated Sorting Genetic Algorithm ll (NSGA-ll). Using NSGA-ll, and a set of optimum points has been considered for an UAV.

In order to improve the accuracy of the comparison, we implemented three other algorithms A[1]-A[3] [69] , OD3P [70], and CPlace [71] that were able to be used on our benchmarks. We experimented with them in the same setting as ours and the result are shown in Tabels 4.8, 4.9, and 4.10. To investigate and compare the proposed method with the methods mentioned above, all they were implemented and run on the same system.

Regarding the number of covered points, our method and CPlace performed better than the others although our method was getting better quality results in all three cases. Further investigation showed that increase in the complexity of the search space and the rate of the changes in the dynamic environment the difference in the quality of our method and the previous method becomes more evident. The same behavior can be seen in the second metric (TCW), which our method outperforms the other ones when the complexity increases.

According to the obtained results, it is clear that due to its static optimization, NSGA-ll need more time to solve this problem, while our method is faster in solving these problems.

Although the CPlace algorithm has a higher accuracy than OD3P, it needs a longer time to do the calculations to find the appropriate places. However, as shown in Table 4.8, 4.9, and 4.10, it is obvious that our proposed method has managed to achieve the highest speed and accuracy as compared with the other methods.

The results show that the proposed method has been able to find more valuable points with a lower computation time. Moreover, the results show that our method has been able to solve various problem in a stable time (less than a second after each change).

Chapter 5

Conclusions

The primary focus of this thesis was related to developing a hybrid framework to improve the safety and efficiency of autonomous vehicles. Due to the increasing popularity of AVs, and their diversity, we selected a swarm of drones as a fast group of AVs, that need a swift reaction and decision making in real-time environments. In the first step, we focused on the safety and efficiency of navigation in a swarm of AVs. We have improved our approach step by step by defining new challenges and proposing new solutions.

Therefore, the main conclusions of this thesis can be summarized as follows:

- To achieve the safety and efficiency of AVs in the navigation problem, we had to solve it as an optimization problem. Therefore, we started to use the dynamic implementation of an evolutionary algorithm. This approach is suitable for environments without any dynamic obstacles.

- To present an efficient method for real dynamic environments, we developed a hybrid framework with different modules for make a fast and reliable decisions for a swarm of autonomous vehicles. Different modules work together under the supervision of a central decision centre to simplify the problem and support each other to generate the most reliable solution. The modules encompass the chosen evolutionary algorithm along with supervised and unsupervised learning algorithms. This combination help to avoid collisions between drones and other objects (static and dynamic objects).

- Also, we solved the dynamic placement of drones to maximize the coverage and minimize the communication cost.

The proposed method have been tested in different complex case studies. These case studies simulated based on the high diversity of dynamic behaviour. For example, in the Paper 10 the weight of points in the case studies have been changed 10000 times.

Also, we have compared the obtained experimental results of our method with other related works. The comparisons showed the proposed framework computed safer and more efficient solutions in shorter computation times.

Chapter 6

Overview of Original Publications

Articles published including the results and analysis from the thesis are summarized below.

6.1 Paper 1: Optimal Placement for Smart Mobile Access Points

In this paper, we proposed a method to solve the efficient placement problem of smart mobile access points (SMAPs). The placement should be simultaneously informative and achieve a low communication cost. The proposed method generates a near-optimal solution to maximize the coverage area and minimizing communication cost.

The proposed algorithm selects the placements of SMAPs achieving a specified volume of certainty, with near-maximal coverage and minimal communication cost, which can ensure providing a near-optimal solution for this hard problem. In this paper, we have used the multi-population MICAP, and MICAP selects SMAP placements at informative and cost-effective locations by the following actions:

1. Discretize the search space area for covering by defining a finite number of points (P).

2. Measure the communication cost for each SMAP and define the corresponding communication cost matrix.

3. Execute MICAP to present the best placement with the minimal communication cost and maximal covering.

The evaluation shows our solution improves efficiency and stability comparing for the other related works.

6.2 Paper 2: Towards A Real time, Collision-Free Motion Coordination and Navigation System for a UAV Flee

In this paper, we presented a real-time, collision-free motion coordination and navigation system for a UAV fleet that used geographical locations of the UAVs and successfully detected, static and dynamically appearing, dynamic obstacles to predict and avoid: 1) UAV-to-UAV collisions, 2) UAV-to-static obstacle collisions, and 3) UAV-to-dynamic obstacle collisions.

The proposed system, includes two main functions: 1) a complex event processing (CEP) and collision prediction module, and 2) a collision avoidance tool.

Also, in this paper presented a simulation-based implementation of the motion coordination and navigation system, along with an experimental evaluation concerning a series of experiments.

6.3 Paper 3: Online Path Generation and Navigation for Swarms of UAVs

This paper is an extended version of Paper 2. In this paper, we focus on collision prediction and avoidance, as well as online path generation and navigation for swarms of UAVs. Same as Paper 2, we present our online, collision-free path generation and navigation system for swarms of UAVs. Our collision prediction strategy leverages efficient run-time monitoring and complex event processing (CEP) to make timely predictions. A distinctive feature of the proposed system is its ability to predict potential collisions and proactively find the best ways to avoid predicted collisions in order to ensure the safety of the entire swarm. We also present a simulation-based implementation of the proposed system, along with an experimental evaluation involving a series of experiments, and compare our results with the results of four other existing approaches.

6.4 Paper 4: Safety-Aware Control of Swarms of Drone

This paper proposes an innovative method for ensuring motion safety of swarms of drones. Our method combines safety-explicit route planning with the run-time safety monitoring and route recalculation aiming at increasing safety and minimizing the travelling distance.

The route planning of swarm of drones is based on two critical convexes, maximizing safety while minimizing the length of the path of each drone. We start by explicitly establishing the requirements that should be verified to guarantee motion safety of a swarm: drones do not collide with the static objects and each other. Then, we use dynamic multi-population ICA to generate the most efficient path for a swarm of drones in a 2-D flying zone. Iteratively, the proposed method

produces the solutions that progressively maximize the value of the specified fitness function.

The key contribution of this paper is the ”middle point” concept. A middle point can be any point in the flying zone that shows that a drone should come to this point from the initial point, and then the drone should move to the destination. Each chromosome contains a certain number of middle points, one per each drone.

6.5 Paper 5: Ensuring fault tolerance and efficiency in autonomous swarm-based monitoring systems

This paper proposes a novel approach to combining optimization and learning methods to achieve fault tolerance and efficiency in swarm-based monitoring. In our solution, there are two main components. The first component is a high-performance dynamic EA for swarm configuration. The second component is a fast and accurate reinforced learning algorithm for incorporating the run-time data and feedback about the system state.

We have proposed a multi-population implementation of the ICA to find a near-optimal swarm configuration, that allows us to maximize the area coverage while minimizing the energy consumption. The k-NN algorithm has been used as the learning component.

Our synchronous algorithm generates a new configuration for a distributed swarm based on the environment state. Both the EA and machine learning algorithms work in parallel to compute alternative swarm configurations. The results of dynamic EA and machine learning algorithm are compared, and a more optimal solution is chosen.

The results show that our method, which integrates EA and machine learning to fine tune the swarm configuration, provides us with a promising solution to guarantee fault tolerance and efficiency of the swarm-based monitoring systems.

6.6 Paper 6:Improving Motion Safety and Efficiency of Intelligent Autonomous Swarm of Drones

In this paper, we proposed a novel approach to ensuring motion safety of swarms of drones. Our approach consists of five components including: *(1) offline-part*, *(2) dynamic evolutionary*, *(3) critical instruction*, *(4) run-time safety monitoring*, and *(5) decision center*. We started by explicitly defining the conditions that should be verified to ensure motion safety of a swarm, which are *(i) swarms do collide with the static objects*, *(ii) with each other* and/or *(iii) with the objects that dynamically appear in the fly zone of the swarm*. In addition, we considered the route planning as an optimization problem aiming to maximize safety while minimizing the length of the path of each drone to achieve higher efficiency. The main novel contribution

of this paper compared to a similar solution, DANA [72], is the consideration of dynamic obstacles in the environment with on-line re-scheduling.

To solve such a complex multi-criteria optimization problem, we have relied on evolutionary computing paradigm [73], [74]. Combination of the genetic algorithm and the the imperialist competitive algorithm (MICGA) [74] is the basis of our proposed solution. By mimicking the processes associated with a competition of imperialistic countries to acquire colonies, the algorithm iteratively generates the solutions that progressively maximize (or minimize) the value of the defined fitness function. In our definition of the fitness function, we explicitly introduce safety as an argument, i.e., ensure that our route planning finds the safest shortest route for each drone.

6.7 Paper 7: Integrating Learning, Optimization, and Prediction for Efficient Navigation of Swarms of Drones

In this paper, we have developed our approach described in Paper 5 by integrating optimization, learning, and prediction for generating efficient and safe paths for swarms of drones.

The proposed method satisfy three main requirements to predict and avoid: (1) drone-to-static-obstacle collision, (2) drone-to drone collisions, and (3) drone-to-moving-obstacle collisions. The proposed method generates more knowledge about the behaviour of moving obstacles to present to our solution generator components. This knowledge generates based on a prediction method based on the last movements of moving-obstacles.

To find the most efficient paths, we have used a parallel and dynamic ICA and machine learning algorithm again. Our experiments has shown that the dynamic EA and the learning algorithm generate a more accurate and efficient solution based on the knowledge that they receive from the prediction component.

Both the learning and optimization algorithms work in parallel to compute alternative routing solutions. The results are compared, and a more efficient solution is chosen. Since with each run the training set increases, eventually, the learning algorithm becomes capable of proposing better solutions. We believe that our proposed approach provides a promising solution to ensure efficient, collision-free navigation of the drones in a swarm. We also present a parallel implementation of the proposed approach and evaluate it against two benchmarks. The results demonstrate that the proposed approach allows us to significantly reduce the route lengths and computation overhead while producing efficient and safe routes.

6.8 Paper 8: Using Optimization, Learning, and Drone Reflexes to Maximize Safety of Swarms of Drones

In this paper, we have improved the proposed method again, and tried to find a solution for some other open questions in this thesis.

To find efficient drone routes, we have proposed a parallel and dynamic implementation of the ICA that allows us to find efficient collision free routes for the drones in a swarm. The learning component is based on the K-nearest neighbour (KNN) learning algorithm. Each path produced by our parallel ICA for a given swarm and environment state is evaluated to train the system. Both the learning and optimization algorithms work in parallel to compute alternative routing solutions. Then the results are compared and the most efficient solution is chosen. Since with each run the training set increases, eventually, the learning algorithm becomes capable of proposing better solutions. To maximize safety, we augment the generated routes with dynamically computed drone reflexes. The drone reflexes computation module mimics a self-preservation control mechanism of humans. The reflexes are the automatic immediate or mechanical responses to particular hazardous situations, such as quickly moving the hand away from a hot surface. They aim at mitigating and confining the effects and damages of suddenly occurring hazards.

In our proposed approach, when a drone detects a possible collision with an unforeseen obstacle, the drone reflexes computation module quickly computes a reflex movement for the drone to prevent and mitigate the collision. We also present a parallel implementation of the proposed approach and evaluate it against two benchmarks. The results show that the proposed approach produces highly efficient and safe routes.

6.9 Paper 9: A Frame-work for Intelligent Navigation of Swarms of Drones

In this paper, we present a framework for intelligent navigation of swarms of drones called DIANA (Dynamic Intelligent Autonomous Navigation Algorithm). It integrates evolutionary optimization, machine learning, prediction, clustering, and automatic immediate responses (reflexes) of drones to ensure safe and efficient operations of swarms of drones. We assume that a swarm executes certain missions, in which each drone flies from its start location to its destination location [18]. The proposed approach uses geographical locations of the drones as well as the static and moving obstacles in the flying zone to predict and avoid: (1) drone-to-drone collisions, (2) drone-to-static-obstacle collisions, and (3) drone-to-moving-obstacle collisions.

DIANA comprises five main components: (1) a high-performance dynamic evolutionary algorithm (EA) for optimizing drone routes, (2) a reinforcement learning algorithm for incorporating the feedback and run-time data about the system state,

(3) a light-weight prediction approach to predict the movement of drones and moving obstacles in real-time, (4) a clustering algorithm to split the set of drones into smaller subsets called clusters, and (5) a reactive module to dynamically compute drone reflexes to prevent collisions with unforeseen obstacles in the flying zone. It uses the dynamic EA and the reinforcement learning algorithm to generate safe and efficient drone routes and then augments the generated routes with dynamically computed drone reflexes to prevent collisions with unforeseen obstacles in the flying zone.

To find efficient drone routes, we propose a parallel and dynamic implementation of the imperialistic competitive algorithm (ICA)[59] that allows us to find efficient collision-free routes for the drones in the swarm. The learning component is based on the *k*-nearest neighbor (kNN) learning algorithm [54]. Each placement produced by our parallel ICA for a given swarm and environment state is evaluated to train the system. Both the learning and optimization algorithms work in parallel to compute alternative routing solutions. The results are compared and a more efficient solution is chosen. Since with each run the training set increases, eventually the learning algorithm becomes capable of proposing better solutions.

The proposed prediction approach is based on a light-weight prediction algorithm [61, 62] that allows to predict the movement of drones and moving obstacles under real-time constraints. Therefore, a distinctive feature of the proposed framework is its ability to foresee a risk of a collision in real-time and proactively find best ways to prevent the predicted collisions in order to ensure safety of the entire swarm [18].

To generate clusters of drones, DIANA uses the K-means clustering algorithm [55]. Moreover, to maximize safety, we augment the generated routes with dynamically computed drone reflexes [33]. The drone reflexes computation module mimics a self-preservation control mechanism of humans. The reflexes are the automatic immediate or mechanical responses to particular hazardous situations, such as quickly moving the hand away from a hot surface. They aim at mitigating and confining the effects and damages of suddenly occurring hazards. In our proposed approach, when a drone detects a possible collision with an unforeseen obstacle, the drone reflexes computation module quickly computes a reflex movement for the drone to prevent and mitigate the collision.

We also present a parallel implementation of the proposed framework and evaluate it against two benchmarks. The results show that the proposed framework maximizes safety, generates highly efficient drone routes, has a low computation overhead, and is highly suitable for large-sized problem instances involving hundreds of drones.

6.10 Paper 10: Cost-Efficient Coverage of Mobile Access Point Placements in Dynamic Cyber Physical Environments

The main goal of the proposed method in this paper is the introduction of an efficient solution to multi-objective optimization problems that maximize the coverage while minimizing the number of agents and communication costs. The placements in a static setting is an NP hard problem [75]. Adding the dynamism concept will make it more complicated. Additionally, the method's latency in finding a solution is more critical, as a new solution after any change in problem dynamics must be found in a short time. In the proposed method, we leverage the deep learning and linear-regression-based method to improve the recognition level of the environment.

We also use the dynamic EC and a supervised learning algorithm to find the optimum placement. Also, the correct operation of all above-mentioned mechanisms is monitored and controlled through the global module. To our knowledge, the proposed method is the only one to solve the dynamic problem of the mobile agents, moreover, it has the following features: In this work, considering the dynamic nature of the problem, we propose a hybrid and dynamic method. In order to have an appropriate functionality, our algorithm has the ability of better recognition of the environment and forecasting event in the future. Also, the method introduces the result as quick as possible. We could not use none of already existing solutions. For example, machine learning algorithms could have an acceptable efficiency if they received inputs similar to the data used during training. In other words, they cannot solve the problems without having prior experience. Conversely, the evolutionary methods try to solve the problem without getting help from the prior experiences. As a result of the above-mentioned issues, we introduce a hybrid method which consists of several algorithms. In this method, we use a combination of dynamic evolutionary, supervised and unsupervised machine learning algorithms, and some deterministic control algorithms which have the responsibility to supervise the whole system and make appropriate decisions. In this section we introduce each component shortly. Later, the exact operation of each of them and their correlation will be explained.

Our algorithm needs the relation with environments. This is done through two parts. One part of our algorithm is responsible to receive the information from the environment. This information is collected from sensors, cameras, or drones. We call this the dynamic monitoring component. This component, in addition to sending information to the MICGA and KNN components to make the decisions and provide with the solutions, sends these information to another component, which is called prediction. This component, after recognizing the dynamic obstacles, probes their behavior, and gives the probability of the presence in a specific place in the future to the dynamic monitoring component, using an optimized linear regression method.

Our solution includes two different methods to find the placement solution, which work in parallel and send their results to the decision center. In this work, we use the dynamic MICGA and KNN in our solution. One of them is an evolutionary method, which has the ability to find solutions without needing prior information (history), and only based on the present situation. The other one (KNN), is a supervised machine learning algorithm that always makes decisions according to the prior experiences. It is also possible to use other learning methods, however KNN is a simpler method with the ability to train the model in the minimum possible time. It is a very time-consuming task in other learning methods, which makes them inefficient in real-time applications.

Evolutionary algorithms begin finding the solution based on an initial random population. This is very ineffective and time-consuming in dynamic environments. In our solution, a semi-random initial population is used that will cause the algorithm to converge to the solution faster. In other words, if we can generate an initial population using the prior knowledge based on the current situation, the convergence speed will increase, and the algorithm reaches the intended solution in a fewer number of iterations. We used a simple and popular CNN to generate a portion of the initial population. We modified it and allowed it to suggest a portion of the initial population using the appearance of the two-dimensional problem space.

The proposed solution by the two above-mentioned methods are sent to the decision center to assess. This component first investigates whether or not any of the proposed solutions meet the required conditions to be sent to the agents. If so, the most suitable one would be selected and announced to the agents. Then, a copy of it is sent to update the placement data sets. If neither of them was satisfactory, for instance is not satisfying safety requirements, this component sends some predefined instructions to the agents until finding a suitable solution. This predefined instruction can be a stop moving until further notice. Thus, this component has the responsibility to keep the agents safe. In other words, this component (decision center) is responsible for final assessment of the suggested solutions and prevention of accidents in the case of any faults in the system.

6.11 Paper 11: Deriving Mode Logic for Autonomous Resilient Systems

In this paper, we have proposed an approach to the formal development of flexible, autonomous systems. The proposed approach allows a developer to derive a resilience enhancing mode logic in a structured, disciplined way. The main objects, which the system should satisfy, serve as a base for determining the mode transition logic.

We have considered distributed autonomous systems that are composed of asynchronously communicating heterogeneous components – agents. Each agent has specific capabilities. Our goal reachability and degree of satisfaction conditions

are defined as corresponding functions over the agent capabilities. Since mode transitions, in general, incur complex agent coordination and system reconfiguration, we need a formal structured approach to ensure the correctness of the mode transition logic. In this paper, we rely on Event-B a state-based approach to correct-by-construction system development to specify and verify the mode logic. moreover, we propose a specification pattern for modelling mode transitions triggered by changes in reachability and degree of satisfaction of non-functional conditions.

6.12 Paper 12: Multi-Layered Approach to Safe Navigation of Swarms of Drone

In this paper, a novel multi-layered approach to safe and efficient navigation of swarms of drones has been presented and formalised it in Event-B. We have defined and validated the architecture that supports a multi-layered distributed approach to ensuring the safety and efficiency of swarm navigation. We have formally defined the requirements that ensure correct coordination of drones using the notion of modes.

The Rodin platform was used to automate modelling and verification efforts. The framework has demonstrated good scalability and provided us with a suitable basis for designing such a complex distributed system as a swarm of drones. We believe that the following aspects were critical for the success of the development. The first aspect is support for refinement and decomposition. It allowed us to start from a centralised succinct system model and derive a complex and tangled coordination mechanism gradually in correctness preserving way. The second aspect is support for highly iterative development provided by the Rodin platform.

Proofs provided us with immediate feedback on our models and helped to spot many intricate interdependencies between modes, phases and effects of faults. We believe that our work has offered a promising solution to the problem of ensuring the dependability of swarm systems.

6.13 Paper 13: Multi-Layered Safety Architecture of Autonomous Systems: Formalising Coordination Perspective

In this paper, we have presented a novel multi-layered approach to ensuring the safety of autonomous systems. Our approach introduced the notions of strategic, tactical and active safety and defined the multi-layered architecture implementing them. The proposed architecture is novel since it allows us to enhance safety in three ways: at the pre-deployment state – by performing safety explicit mission planning; at the run-time – by replanning, the mission and reconfiguring the system to maintaining safety and efficiency; at critical state – when an unexpected hazard

emerged and drones should promptly react to mitigate the risks. This paper complements our work on developing high-performance optimization algorithms for safe navigation of autonomous systems. They allow us to safely navigate the drones and optimize the travel distance, resource consumption and quality of the payload data ratio. The algorithms also ensure the inter-drone and drone-obstacle collision avoidance. Since the proposed multi-layered safety architecture requires complex coordination and intercommunication, we have employed the Event-B framework to formalize and verify the proposed coordination and communication mechanisms.

Bibliography

[1] S. S. Nikolić, An innovative response to commercial uav menace: Anti-uav falconry, Vojno delo 69 (4) (2017) 146–167.

[2] A. Eiben, J. Smith, Introduction to Evolutionary Computing, Springer, 2003.

[3] E.Atashpaz-Gargari, C. Lucas, Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition, in: IEEE Congress on Evolutionary Computation, 2007, pp. 4661–4666.

[4] J. Kephart, D. Chess, The vision of autonomic computing, Computer Volume 36 (1) (2003) 41–50.

[5] G. S. M. J. P. A.Majd, S.Lotfi, Pica: Multi-population implementation of parallel imperialist competitive algorithms, in: 24th Euromicro Imternational Conference of Parallel, Distributed and Network-Based Processing, IEEE, 2016, pp. 248–255.

[6] J. A. Suykens, J. Vandewalle, Least squares support vector machine classifiers, Neural processing letters 9 (3) (1999) 293–300.

[7] G. A. Seber, A. J. Lee, Linear regression analysis, Vol. 329, John Wiley & Sons, 2012.

[8] M. A. Mansournia, A. Geroldinger, S. Greenland, G. Heinze, Separation in logistic regression: causes, consequences, and control, American journal of epidemiology 187 (4) (2018) 864–870.

[9] H. Zhang, The optimality of naive bayes, AA 1 (2) (2004) 3.

[10] S. Tan, Neighbor-weighted k-nearest neighbor for unbalanced text corpus, Expert Systems with Applications 28 (4) (2005) 667–671.

[11] A.-A. Liu, Y.-T. Su, W.-Z. Nie, M. Kankanhalli, Hierarchical clustering multi-task learning for joint human action grouping and recognition, IEEE transactions on pattern analysis and machine intelligence 39 (1) (2016) 102–114.

[12] A. Likas, N. Vlassis, J. J. Verbeek, The global k-means clustering algorithm, Pattern recognition 36 (2) (2003) 451–461.

[13] A. Maligo, S. Lacroix, Classification of outdoor 3d lidar data based on unsupervised gaussian mixture models, IEEE Transactions on Automation Science and Engineering 14 (1) (2016) 5–16.

[14] R. Al-Jawfi, Handwriting arabic character recognition lenet using neural network., Int. Arab J. Inf. Technol. 6 (3) (2009) 304–309.

[15] S. Albawi, T. A. Mohammed, S. Al-Zawi, Understanding of a convolutional neural network, in: 2017 International Conference on Engineering and Technology (ICET), Ieee, 2017, pp. 1–6.

[16] J.-R. Abrial, Modeling in Event-B, Cambridge University Press, 2010.

[17] Rodin, Event-b platform (2007).
URL http://www.event-b.org/

[18] A. Ashraf, A. Majd, E. Troubitsyna, Towards a realtime, collision-free motion coordination and navigation system for a UAV fleet, in: Proceedings of the Fifth European Conference on the Engineering of Computer-Based Systems, ECBS '17, ACM, 2017, pp. 11:1–11:9. doi:10.1145/3123779.3123805.

[19] R. Siegwart, I. R. Nourbakhsh, D. Scaramuzza, Introduction to Autonomous Mobile Robots, Intelligent robotics and autonomous agents, MIT Press, 2011.

[20] T. Fraichard, A short paper about motion safety, in: Proceedings 2007 IEEE International Conference on Robotics and Automation, 2007, pp. 1140–1145. doi:10.1109/ROBOT.2007.363138.

[21] K. Macek, D. Vasquez, T. Fraichard, R. Siegwart, Safe vehicle navigation in dynamic urban scenarios, in: 2008 11th International IEEE Conference on Intelligent Transportation Systems, 2008, pp. 482–489. doi:10.1109/ITSC.2008.4732685.

[22] A. Aniculaesei, D. Arnsberger, F. Howar, A. Rausch, Towards the verification of safety-critical autonomous systems in dynamic environments, in: Proceedings of the The First Workshop on Verification and Validation of Cyber-Physical Systems, 2016, pp. 79–90. doi:10.4204/EPTCS.232.10.

[23] S. Petti, T. Fraichard, Partial motion planning framework for reactive planning within dynamic environments, in: Proceedings of the IFAC/AAAI International Conference on Informatics in Control, Automation and Robotics, 2005.

[24] X. Dong, B. M. Chen, G. Cai, H. Lin, T. H. Lee, Development of a comprehensive software system for implementing cooperative control of multiple

unmanned aerial vehicles, in: IEEE International Conference on Control and Automation, 2009, pp. 1629–1634. doi:10.1109/ICCA.2009.5410149.

[25] B. J. O. de Souza, M. Endler, Coordinating movement within swarms of UAVs through mobile networks, in: 2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops), 2015, pp. 154–159. doi:10.1109/PERCOMW.2015.7134011.

[26] A. Bürkle, F. Segor, M. Kollmann, Towards autonomous micro UAV swarms, Journal of Intelligent & Robotic Systems 61 (1) (2011) 339–353. doi:10.1007/s10846-010-9492-x.

[27] A. Ivanovas, A. Ostreika, R. Maskeliūnas, R. Damaševičius, D. Połap, M. Woźniak, Block matching based obstacle avoidance for unmanned aerial vehicle, in: L. Rutkowski, R. Scherer, M. Korytkowski, W. Pedrycz, R. Tadeusiewicz, J. M. Zurada (Eds.), Artificial Intelligence and Soft Computing, Springer International Publishing, 2018, pp. 58–69.

[28] A. J. Barry, R. Tedrake, Pushbroom stereo for high-speed navigation in cluttered environments, in: 2015 IEEE International Conference on Robotics and Automation (ICRA), 2015, pp. 3046–3052. doi:10.1109/ICRA.2015.7139617.

[29] Y. Lin, Moving obstacle avoidance for unmanned aerial vehicles, Ph.D. thesis, Arizona State University (2015).

[30] C. Goerzen, Z. Kong, B. Mettler, A survey of motion planning algorithms from the perspective of autonomous UAV guidance, Journal of Intelligent and Robotic Systems 57 (1) (2009) 65. doi:10.1007/s10846-009-9383-1.

[31] F. Kendoul, Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems, Journal of Field Robotics 29 (2) (2012) 315–378. doi:10.1002/rob.20414.

[32] F. Augugliaro, A. P. Schoellig, R. D'Andrea, Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach, 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (2012) 1917–1922doi:10.1109/IROS.2012.6385823.

[33] A. Majd, A. Ashraf, E. Troubitsyna, M. Daneshtalab, Using optimization, learning, and drone reflexes to maximize safety of swarms of drones, in: 2018 IEEE Congress on Evolutionary Computation (CEC), 2018.

[34] A. Majd, A. Ashraf, E. Troubitsyna, M. Daneshtalab, Integrating learning, optimization, and prediction for efficient navigation of swarms of drones, in: 2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP), 2018, pp. 101–108. doi:10.1109/PDP2018.2018.00022.

[35] P. B. Sujit, R. Beard, Multiple UAV path planning using anytime algorithms, in: 2009 American Control Conference, 2009, pp. 2978–2983. doi:10.1109/ACC.2009.5160222.

[36] J. d. Silva Arantes, M. d. Silva Arantes, C. F. Motta Toledo, O. T. Júnior, B. C. Williams, Heuristic and genetic algorithm approaches for UAV path planning under critical situation, International Journal on Artificial Intelligence Tools 26 (01) (2017) 1760008. arXiv:https://doi.org/10.1142/S0218213017600089, doi:10.1142/S0218213017600089.

[37] R. Siegwart, I.R.Nourbakhsh, Introduction to Autonomous Mobile Robots, MIT Press, 2004.

[38] Th.Fraichard, A short paper about motion safety, in: Proc. Of the IEEE Int. Conf. on Robotics and Automation, IEEE, 2007.

[39] T. K.Macek, D.Vasquez, R.Siegwart, Safe vehicle navigation in dynamic urban scenarios, in: Proc. of 11th International IEEE Conference on Intelligent Transportation Systems, IEEE, 2008, pp. 482–489.

[40] F. H. A. R. A. Aniculaesei, D. Arnsberger, Towards the verification of safety-critical autonomous systems in dynamic environments, in: In Proc. of the 1st International Workshop on Verification and Validation of Cyber Physical Systems – V2CPS, EPTCS 232, 2016, pp. 79–90.

[41] S. Petti, T. Fraichard, Partial motion planning framework for reactive planning within dynamic environments, in: Proc. of the IFAC/AAAI Int. Conf. on Informatics in Control, Automation and Robotics, 2005.

[42] J. Suomela, Computational complexity of relay placement in sensor networks, in: International Conference on Current Trends in Theory and Practice of Computer Science, Springer, 2006, pp. 521–529.

[43] S. Toumpis, G. A. Gupta, Optimal placement of nodes in large sensor networks under a general physical layer model., in: SECON, Citeseer, 2005, pp. 275–283.

[44] A. Majd, G. Sahebi, M. Daneshtalab, J. Plosila, H. Tenhunen, Placement of smart mobile access points in wireless sensor networks and cyber-physical systems using fog computing, in: 2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld), IEEE, 2016, pp. 680–689.

[45] Y. T. Hou, Y. Shi, H. D. Sherali, S. F. Midkiff, On energy provisioning and relay node placement for wireless sensor networks, IEEE Transactions on Wireless Communications 4 (5) (2005) 2579–2590.

[46] C.-C. Shen, O. Koc, C. Jaikaeo, Z. Huang, Trajectory control of mobile access points in manet, in: GLOBECOM'05. IEEE Global Telecommunications Conference, 2005., Vol. 5, IEEE, 2005, pp. 6–pp.

[47] W. Youssef, M. Younis, Intelligent gateways placement for reduced data latency in wireless sensor networks, in: 2007 IEEE International Conference on Communications, IEEE, 2007, pp. 3805–3810.

[48] A. Krause, C. Guestrin, A. Gupta, J. Kleinberg, Near-optimal sensor placements: Maximizing information while minimizing communication cost, in: Proceedings of the 5th international conference on Information processing in sensor networks, 2006, pp. 2–10.

[49] S. Bi, R. Zhang, Placement optimization of energy and information access points in wireless powered communication networks, IEEE transactions on wireless communications 15 (3) (2015) 2351–2364.

[50] A. Majd, M. Daneshtalab, E. Troubitsyna, G. Sahebi, Optimal smart mobile access point placement for maximal coverage and minimal communication, in: Proceedings of the Fifth European Conference on the Engineering of Computer-Based Systems, 2017, pp. 1–2.

[51] A. Majd, S. Lotfi, G. Sahebi, Review on parallel evolutionary computing and introduce three general framework to parallelize all ec algorithms, in: The 5th Conference on Information and Knowledge Technology, IEEE, 2013, pp. 61–66.

[52] E. W. Dijkstra, A note on two problems in connexion with graphs, Numerische Mathematik 1 (1) (1959) 269–271. doi:10.1007/BF01386390.

[53] A. Majd, G. Sahebi, M. Daneshtalab, J. Plosila, S. Lotfi, H. Tenhunen, Parallel imperialist competitive algorithms, Concurrency and Computation: Practice and Experience 30 (7) (2018) e4393. doi:10.1002/cpe.4393.

[54] M. Mejdoub, C. Ben Amar, Classification improvement of local feature vectors over the KNN algorithm, Multimedia Tools and Applications 64 (1) (2013) 197–218. doi:10.1007/s11042-011-0900-4.

[55] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, A. Y. Wu, An efficient k-means clustering algorithm: Analysis and implementation, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (7) (2002) 881–892. doi:10.1109/TPAMI.2002.1017616.

[56] N. Heo, P. K. Varshney, Energy-efficient deployment of intelligent mobile sensor networks, IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans 35 (1) (2005) 78–92. doi:10.1109/TSMCA.2004.838486.

[57] C. Guestrin, A. Krause, A. P. Singh, Near-optimal sensor placements in gaussian processes, in: Proceedings of the 22nd International Conference on Machine Learning, ICML '05, ACM, 2005, pp. 265–272. doi:10.1145/1102351.1102385.

[58] A. Majd, E. Troubitsyna, M. Daneshtalab, Safety-aware control of swarms of drones, in: S. Tonetta, E. Schoitsch, F. Bitsch (Eds.), Computer Safety, Reliability, and Security (SAFECOMP), Lecture Notes in Computer Science, Volume 10489., Springer International Publishing, 2017, pp. 249–260.

[59] E. Atashpaz-Gargari, C. Lucas, Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition, in: IEEE Congress on Evolutionary Computation, 2007, pp. 4661–4667. doi:10.1109/CEC.2007.4425083.

[60] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324. doi:10.1109/5.726791.

[61] M. Andreolini, S. Casolari, M. Colajanni, Models and framework for supporting runtime decisions in web-based systems, ACM Transactions on the Web 2 (3) (2008) 17:1–17:43. doi:10.1145/1377488.1377491.

[62] A. Ashraf, B. Byholm, I. Porres, A session-based adaptive admission control approach for virtualized application servers, in: 2012 IEEE Fifth International Conference on Utility and Cloud Computing, 2012, pp. 65–72. doi:10.1109/UCC.2012.22.

[63] M. Andreolini, S. Casolari, Load prediction models in web-based systems, in: Proceedings of the 1st International Conference on Performance Evaluation Methodolgies and Tools, ACM, 2006. doi:10.1145/1190095.1190129.

[64] D. Montgomery, E. Peck, G. Vining, Introduction to Linear Regression Analysis, Wiley Series in Probability and Statistics, John Wiley & Sons, 2012.

[65] E. Dijkstra, A note on two problems in connexion with graphs (1) (1959) 269–271.

[66] S. Indhumathi, D. Venkatesan, Improving coverage deployment for dynamic nodes using genetic algorithm in wireless sensor networks, Indian Journal of Science and Technology 8 (16) (2015).

[67] S. M. LaValle, J. J. Kuffner, B. Donald, et al., Rapidly-exploring random trees: Progress and prospects, Algorithmic and computational robotics: new directions 5 (2001) 293–308.

[68] J. Bialkowski, S. Karaman, E. Frazzoli, Massively parallelizing the rrt and the rrt, in: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2011, pp. 3513–3518.

[69] H. Huang, A. V. Savkin, An algorithm of efficient proactive placement of autonomous drones for maximum coverage in cellular networks, IEEE Wireless Communications Letters 7 (6) (2018) 994–997.

[70] C. Caillouet, T. Razafindralambo, D. Zorbas, Optimal placement of drones for fast sensor energy replenishment using wireless power transfer, in: 2019 Wireless Days (WD), IEEE, 2019, pp. 1–6.

[71] M. Alharthi, A.-E. M. Taha, H. S. Hassanein, Dynamic controller placement in software defined drone networks, in: 2019 IEEE Global Communications Conference (GLOBECOM), IEEE, 2019, pp. 1–6.

[72] A. Majd, E. Troubitsyna, M. Daneshtalab, Safety-aware control of swarms of drones, in: International Conference on Computer Safety, Reliability, and Security, Springer, 2017, pp. 249–260.

[73] E.Atashpaz-Gargari, C. Lucas, Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition, in: IEEE Congress on Evolutionary Computation, 2007, pp. 4661–4666.

[74] A. Majd, S. Lotfi, G. Sahebi, M. Daneshtalab, J. Plosila, Pica: multi-population implementation of parallel imperialist competitive algorithms, in: 2016 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP), IEEE, 2016, pp. 248–255.

[75] V. Tzoumas, M. A. Rahimian, G. J. Pappas, A. Jadbabaie, Minimal actuator placement with bounds on control effort, IEEE Transactions on Control of Network Systems 3 (1) (2015) 67–78.

Amin Majd

DIANA: Distributed and Safe Autonomous Navigation for a Swarm of Autonomous Vehicles

In this thesis, I tried to work on the development of high-performance algorithms for controlling the swarm of Autonomous Vehicles. The framework proposes algorithms for ensuring safe, reliable and efficient control of swarm systems. The theoretical basis consists of machine learning, evolutionary algorithm, some deterministic approaches, and formal modeling. As a result of my research, I have proposed the algorithm that combines evolutionary algorithm, supervised and unsupervised machine learning to allow swarm systems to maintain safe, reliable and efficient operation despite the changes in the internal and external operating conditions.

ISBN 978-952-12-4058-4