Czech Technical University in Prague
Faculty of Electrical Engineering

DIPLOMA THESIS

Bc. Ondřej Hrstka

Intelligent Surveillance Algorithms for Harbor Security

Department of cybernetics
Diploma thesis supervisor: **Ing. Ondřej Vaněk**

Prague, 2013

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 5.7.2013 ...Hrdlička.....

podpis

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

DIPLOMA THESIS ASSIGNMENT

Student: Bc. Ondřej H r s t k a

Study programme: Open Informatics

Specialisation: Artificial Intelligence

Title of Diploma Thesis: Intelligent Surveillance Algorithms for Harbor Security

Guidelines:

1. Study problems of harbor security, focus on boat patrolling and aerial surveillance.
2. Create an agent-based computational model of Mombasa harbor, containing:
   a) Harbor environment, coastal lines and ports,
   b) Merchant vessel agents agents,
   c) Robber agents,
   d) Patrol agents (UAVs and patrol boats).
3. Study and implement existing surveillance and tracking algorithms.
4. Design surveillance and tracking algorithms accounting for differing attributes of patrol agents (i.e. speed of boats vs. speed of UAVs).
5. Compare performance of designed algorithm with those from (3).

Bibliography/Sources: Will be provided by the supervisor.

Diploma Thesis Supervisor: Ing. Ondřej Vaněk

Valid until: the end of the winter semester of academic year 2013/2014

prof. Ing. Vladimír Mařík, DrSc.
Head of Department

L.S.

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, June 3, 2013

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: Bc. Ondřej Hrstka

Studijní program: Otevřená informatika (magisterský)

Obor: Umělá inteligence

Název tématu: Inteligentní sledovací algoritmy pro bezpečnost v přístavech

Pokyny pro vypracování:

1. Nastudujte problém zabezpečení přístavu, soustřeďte se na patrolovací čluny a sledování ze vzduchu.
2. Vytvořte agentní výpočetní model přístavu v Mombase, který bude obsahovat:
   a) prostředí přístavu, tj. pobřežní linie a nákladní mola,
   b) agenty reprezentující obchodní lodě,
   c) agenty reprezentující zloděje,
   d) patrolovací agenty (patrolovací čluny a bezpilotní letadla).
3. Nastudujte a implementujte existující algoritmy pro sledování a trackování.
4. Navrhněte sledovací a trackovací algoritmy, které počítají s různými atributy patrolovacích agentů (tj. rychlost člunu a rychlost bezpilotních letadel).
5. Porovnejte kvalitu navržených algoritmů s těmi z bodu (3).

Seznam odborné literatury: Dodá vedoucí práce.

Vedoucí diplomové práce: Ing. Ondřej Vaněk

Platnost zadání: do konce zimního semestru 2013/2014

prof. Ing. Vladimír Mařík, DrSc.
vedoucí katedry

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická
L.S.

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 3. 6. 2013

Abstrakt

Ochrana kritických dopravních uzlů je životně důležitá pro mnoho zemí. Současný vývoj v oblasti bezpilotních prostředků poskytuje příležitost na využití těchto technologií k ochraně zmíněných uzlů. Přístav v Mombase, ležící na východním pobřeží Afriky poblíž pirátských vod, je jedním z míst, které by mohlo využít bezpilotní prostředky pro hlídkování nad oblastí anebo nad loděmi.

V této práci se ukazuje že oba problémy, hlídkování nad oblastí i hlídkování nad loděmi, mohou být redukovány na problém hlídkování nad grafem. Dále je pro hlídkování nad obastí navržen algoritmus s polynomiální výpočetní složitostí. Hlídkování nad loděmi je formulováno jako problém hlídačů na více trasách. Formulace je založena na problému více obchodních cestujících a je řešena za použití celočíselného lineární programu. Všechny formulace berou v potaz různorodé hlídkové jednotky. Pro otestování řešení byla vytvořena událostmi řízená multi-agentní simulace přístavu v Mombase. Výsledky ukazují robustnost použitého přístupu a škálovatelnost v rámci parametrů přístavu v Mombase.

Abstract

The problem of protection of critical transportation hubs is of a vital importance for many countries. The recent development in the field of autonomous vehicles technology provides an opportunity to utilize this technology in order to protect these hubs. The Mombasa harbour on the east cost of Africa close to pirate infested waters is one of such places which would greatly benefit from the utilization of the autonomous vehicles for surveillance and ship patrolling.

First, it is shown that problems of area surveillance and of multiple target patrolling can be reduced to patrolling problems on graphs. Second, an optimal algorithm for the area surveillance with polynomial computational complexity is presented. Third, for the multiple target patrolling problem, a formulation of the multiple patroller path problem is proposed. The formulation is based on multiple travelling salesman problem and it is solved using integer linear programming methods. All formulations account for heterogeneous patrolling units. Finally, a multi-agent event-based simulation of Mombasa harbour was created to evaluate presented algorithms. Results show robustness of the approach as well as scalability to real-world scenarios from the Mombasa harbour.

Acknowledgements

I would like to thank to my colleagues from Agents Technology Center who gave me lot of inspiration. Especially to my thesis supervisor and mentor Ing. Ondřej Vaněk for providing valuable advice and guidance.

Contents

List of Figures

Introduction

Protection of critical transportation hubs is of a vital importance for many countries. For example the pirate activity near the east cost of Africa affects security of merchant vessels in national waters. As the utilization of the unmanned aerial vehicles in security domain has been growing for the last years rapidly, this technology is becoming ready to be used in this task.

The goal of this thesis is to design algorithms to control group of unmanned aerial and surface vehicles in Mombasa harbour which is located near pirate infested waters and is an important trading center in the region. These vehicles should patrol the harbour area in a way to minimize the chance of robber attacking a commercial vessel.

The problem of surveillance by unmanned aerial vehicles (UAV) in urban areas is approached by Jakob et al. (2010). This work takes into account the occlusions created by buildings in environment. Solution is not suitable for this domain because it does not consider non-rectangular surveillance areas. Healey et al. (2007) uses the combination of UAVs and unmanned surface vehicles (USV) to increase the maritime awareness. However this work focus more on trajectory control and on localization of the units. Nigam and Kroo (2008) approach the problem of multiple UAVs surveillance. The proposed solutions are limited to the homogeneous units and to the simplified environment model.

Problems examined in this work are of two main types: area surveillance problem and target patrolling problem. However it is presented that area surveillance and target patrolling can be reduced to the same type of patrolling problem.

The area surveillance algorithm uses grid discretization of area and uses polynomial algorithm to find hamiltonian cycle in triangular grid which represents a hexagonal tiling. This type of grid tilling is proved to be the optimal one for the surveillance task. Furthermor a proof of hamiltonian cycle being optimal solution for patrolling on grid graph is presented. It is also proved that during the area surveillance, the different mobility capability between UAV and USV (air versus water) is not an issue – an optimal patrolling paths are circumscribed inside the surveillance area.

To solve patrolling problem on general graph with multiple patrollers, a novel

multiple patroller path problem (MPPP) was introduced. This problem is similar to multiple travelling salesman problem with additional constraints which take into account criterion for patrolling problem. All algorithms are extended to work with heterogeneous set of patrollers.

To evaluate the performance of the algorithms, a multi agent simulation of the Mombasa harbour has been developed. The event based simulation engine is selected to achieve better performance . The simulation contains all actors necessary – commercial vessels, robbers and patrollers which are modelled as independent agents.

The results show that MPPP is hardest for certain deployment-to-saturation ratio, which is ration between the number of patrollers and the number of targets. Furthermore it is presented that the algorithm converge at early stage of optimization. From evaluation in simulation it is apparent that the number of successful attacks is linear function of the time for that take to perform such attack.

Thesis Overview

Domain background described in chapter 1 studies the domain of city Mombasa and unmanned vehicles.

Chapter 2 contains overview of state of the art in patrolling domain and examines methods used further in this thesis.

The algorithms used to route patroller are approached in chapter 3. The patrolling problem terms are introduced informally and then described formally. The algorithms are first designed for homogeneous patrollers. Next these solutions are modified for heterogeneous patrollers. At the end of the chapter additional variants of patrolling problem are discussed.

Chapter 4 describes modelling of the Mombasa harbour simulation. First, the simulation design is described. Second, the information about Mombasa harbour model are presented. Third the agents are described.

The algorithms are evaluated in chapter 5. First part of the evaluation examines performance of an integer linear program formulation of patrolling problem. The evaluation in a simulation is then concluded.

Chapter 1

Domain background

The harbour security is an issue in multiple harbours around the world[1]. Vessels on anchor are being boarded from small boats. The robbers steal cargo and use their boat to escape. The coast guard is typically alerted after the vessel's crew detects, that the cargo was stolen. On of these harbours is Mombasa which is simulated in this work.

This chapter contains description of the harbour security domain. In section 1.1 the Mombasa harbour is described. Sections 1.2 and 1.3 contains description of unmanned aerial and surface vehicles respectively.

1.1 Mombasa Harbour

The Mombasa is the second largest city in Kenya. It is a trading center for the east cost of the African continent. The city contains several harbours the largest of which is Kilindini harbour. Closer geographical description is in section 4.2.

The location of Mombasa which is close to the pirate infested waters makes it easy target for illegal activities. During last years there were several reported robberies in the Kilindini harbour. Data were taken from AgentC[2] project internal database. Locations of these incidents are depicted in Figure 1.1.

The example of the robber attempt report from AgentC database gives an idea about nature of these attacks: "*Two robbers armed with knives boarded a container vessel moored to buoys. The onboard security men sighted the robbers on the forecastle deck and raised the alarm. Robbers managed to escape with stolen ship's stores. Incident reported to the local authorities. All crew safe*".

In order to improve security Mombasa could benefit from the employing unmanned surveillance units. These units, equipped with camera, could provide moni-

[1] See ICC Commercial Crime Services `http://icc-ccs.org/`.
[2] `http://agents.cz/projects/agentc/`

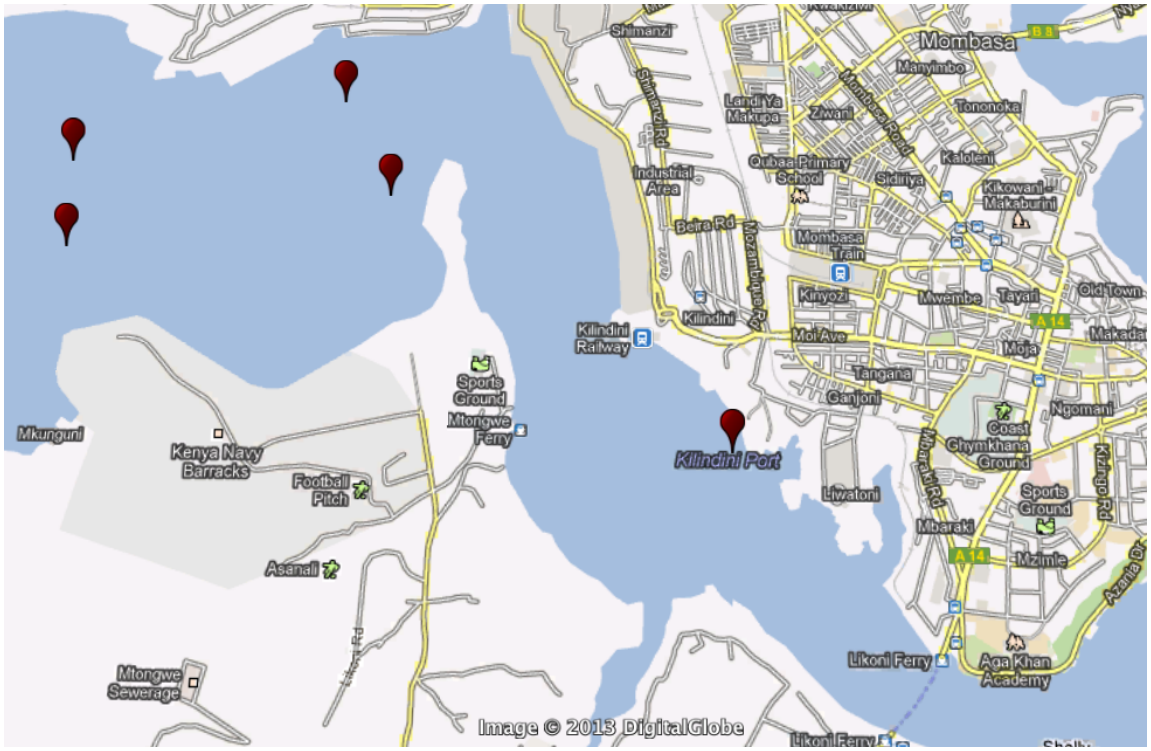

Figure 1.1: Reported incidents in Mombasa harbours

toring services for security authority. That way security authority could cover more ground with minimal requirements for manpower.

1.2 Unmanned Aerial Vehicle

An unmanned aerial vehicle (UAV) is an aerial vehicle controlled by a computer or operated by human remotely. Its size ranges from small hand-launched UAVs like RQ-11 Raven (AeroVironment, 2013) to the big UAVs operating like normal aircraft (Atomics, 2013).

The UAVs are already being deployed to provide critical infrastructure surveillance (aerosurveillance.com, 2013a). The UAVs deployed for these scenarios can be in form of small helicopters (rotary-wing) or in form of small aircraft (fixed-wing) like UAVs from Aero Surveillance Inc. (aerosurveillance.com, 2013b). They leverage third-party UAV platforms to provide a range of fixed- or rotary-wing solutions for small and medium tactical unmanned systems

Current state of the art is deployment of the autonomous UAVs operated by artificial intelligence. Selecký et al. (2013) presented successful deployment of two real UAVs in mixed-simulation environment. They also managed to run real life scenario with area surveillance.

Figure 1.2: The Arcturus T-20 fixed wing civilian mini-UAV Arcturus (2013)

1.3 Unmanned Surface Vehicle

The unmanned surface vehicle is an automated boat operated remotely by human or autonomously by computer.

The example of the USV is SeaFox USV developed at Naval Postgraduate School (Program Executive Officer, 2013). The SeaFox Unmanned Surface Vehicle (USV) is a UOES[3] vehicle that provides a remote, unmanned Intelligence, Surveillance, and Reconnaissance (ISR) capability, supporting multiple mission areas such as: Anti-Terrorism / Force Protection (AT/FP), Riverine Operations, Maritime Interdiction Operations (MIO), **Maritime Domain Awareness** (MDA), and **Port Security** (Program Executive Officer, 2013)

[3]User Operational Evaluation Systems

Figure 1.3: The SeaFox USV.

Chapter 2

Related Work

This chapter contains study of state of the art methods and other methods used in this thesis. In general the security related problems are very old. In this chapter only the most relevant to the thesis very picked. The section 2.1 contains overview of research done in patrolling domain. In section 2.2 various methods used in this thesis are described.

2.1 State of the Art Methods for Patrolling Problems

Bošanský et al. (2011) use game theory to formulate the problem of patrolling mobile targets and they solve the problem for a single patroller and a single attacker. They seek the solution in form of Markov Policy on a graph, taking into account movement of mobile targets. They use mixed integer programming techniques. They assume that the attacker observes the defender and seek for the defender's optimal randomized strategy in a Stackelberg equilibrium of the game.

Jakob et al. (2010) presents an agent-based coordination and planning method for aerial surveillance of multiple urban areas using a group of fixed-wing UAVs. The method differs from the existing work by explicit consideration of sensor occlusions that can occur due to high buildings and other obstacles in the target area. The solution employs a decomposition of the problem into two subproblems: the problem of single-area surveillance and the problem of allocation of UAVs to multiple areas. The work presents an algorithm based on generating a zig-zag pattern path through the surveilled rectangular area.

Faigl and Přeučil (2011) use self-organizing map to solve museum watchman problem; finding a shortest closed collision free path for a mobile robot operating in a planar environment represented by a polygonal map W. The requested path has

to visit a given set of areas where the robot takes measurements in order to find an object of interest.

Problem of patrolling in navy domain is approached by Healey et al. (2007). Authors presented work on deployment of the UAVs and USVs, where UAVs were used to feed data to the USVs. Their work is focused on low level control of trajectory of the UAV which uses camera to track the USV.

Chevaleyre (2004) describes surveillance (or patrolling) task on graph $G = (V, E)$ as a task to continuously visit all the graph nodes so as the time lag between two visits is minimized. He then proves that this problem for one agent is equal to finding travelling salesman problem solution. He defines the strategy of an agent as a function $\pi : \mathbb{N} \mapsto V$ such that $\pi(j)$ is the jth node visited by agent. A *multi-agent strategy* $\Pi = \{\pi_1, \ldots, \pi_k\}$ is simply defined as set of k single agents strategies. As a criterion to evaluate strategies *worst idleness* is used (Machado et al., 2003), which is maximal time lag between two visits of one node.

Nigam and Kroo (2008) present a semi-heuristic patrolling algorithm for a single UAV which is extended to the case of multiple UAVs using two methods. One is an extension of the semi-heuristic patrolling algorithm for a single UAV and the other involves allocation of sub-regions to individual UAVs for parallel exploration. The work furthermore focuses on creating the trajectory for UAVs with respect to aircraft dynamics. Presented algorithms are suitable for surveillance, however they are not optimal.

Jakob et al. (2012) created the system AgentC, which is a data-driven piracy-aware agent-based model of maritime transportation. Using AgentC, authors are able to test various counter-piracy measures including commercial vessels grouping, introduction of new transit corridors and *patrolling*.

Komenda et al. (2013) present Alite Tactical Package which is simulation of village environment with various assets usable in tactical missions, including patrolling units like UAVs or unmanned ground vehicles (UGV).

2.2 Solution Related Methods

The following text contains description of various methods used in presented solutions. In section 2.2.1 issue of tiling by triangular grid together with description of algorithm that allows to find hamiltonian cycle in this grid in polynomial time is described. This knowledge is leveraged in area surveillance described in section 3.3.1. Section 2.2.2 describes formulation of travelling salesman problem using integer linear programming upon which is based solution of attack detection with static targets in section 3.3.2. In section 2.2.3 is discussed problematic of multi-agent simulation which is tied with chapter 4 which describes modelling of Mombasa harbour.

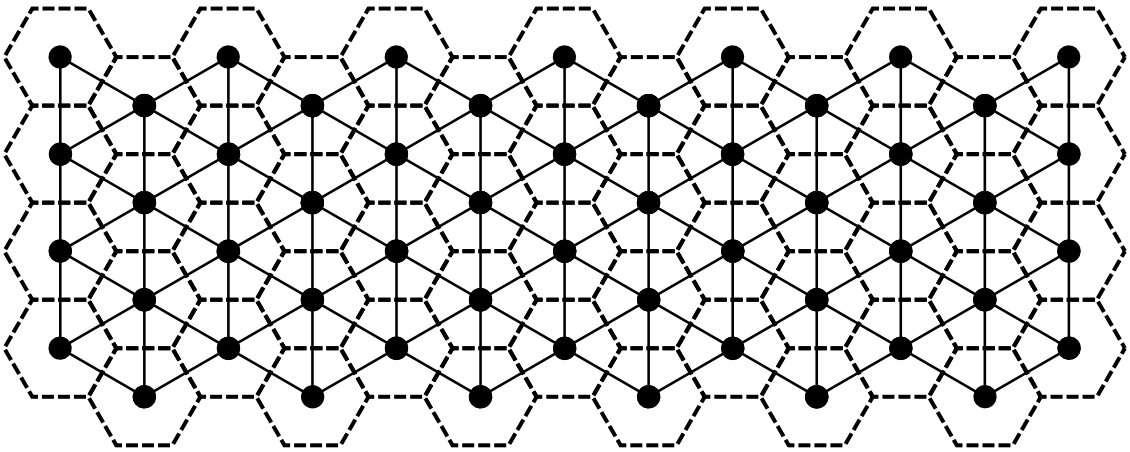

Figure 2.1: Example of triangular grid graph

2.2.1 Grid Tilling

Grids can be used to effectively represent area as a graph structure. In this thesis sampling by triangular grid graph $G = (V, E)$ is used. Polishchuk et al. (2006) define (finite) triangular grid graph as a set of nodes of a tiling of the plane with equilateral triangles. The example of triangular grid graph is depicted in Figure 2.1. The Figure also depicts another way to interpret triangular grid graph; as a set of nodes placed into centres of hexagons tiling the plane.

This thesis also utilizes the fact that it is possible to find a *hamiltonian cycle (HCP)* in polynomial time in triangular grid graph (with some properties) even though HCP is one of the basic six NP-complete problems (Garey and Johnson, 1979). This solution was presented by Arkin et al. (2009) in a work which gives systematic study of Hamiltonicity of grids. Arkin et al. (2009) present a polynomial algorithm for HCP for a finite triangular solid grid graph with one exception which is graph the Star of David (in Figure 2.2. *"A solid grid graph is a grid graph all of whose bounded faces have area one"* (Umans and Lenhart, 1997). Informally a solid grid graph is a grid graph without holes. The example of non-solid grid graph is in Figure 2.3 and an example of a solid grid graph is in Figure 2.4.

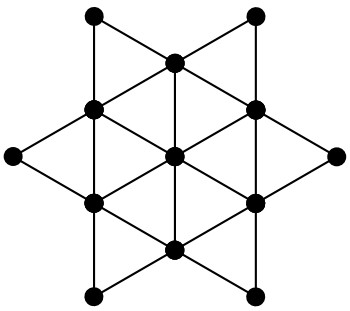

Figure 2.2: The Star of David graph.

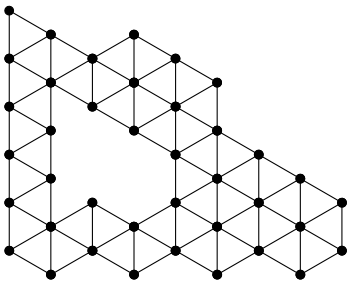

Figure 2.3: Non-solid grid graph example.

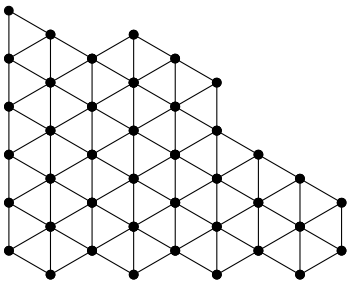

Figure 2.4: Solid grid graph example.

The algorithm presented by Arkin et al. (2009) is based on finding an initial cycle B going on the boundary of the graph G. The cycle B is then modified by adding one node at a time using so called *L-, V-* and *Z-modifications*. Each of these modifications adds one node which is not in B into B in a way that B remains simple cycle. At each step the algorithm tries to apply *L-modification* if it isn't applicable, it tries *V-modification* and if it also isn't applicable, it uses *Z-modification*. The algorithm ends when all nodes are in B which is then hamiltonian cycle or when any of these modifications cannot be applied which means that G is the Star of David.

2.2.2 Travelling Salesman Problem

Miller et al. (1960) defines the *travelling salesman problem (TSP)* that a salesman is required to visit each of n cities exactly once and he must end up in the starting city. The problem is to find such itinerary that the travelled distance is minimal. Miller et al. (1960) also describe how the problem can be expressed as *mixed integer linear program (ILP)*. This program contains $O(n^2)$ constraints with $O(n^2)$ variables where n is number of nodes in graph. Based on this solution, Bektas (2006) presented ILP formulation for *multiple travelling salesman problem (mTSP)* with the same complexity of variables and constraints:

$$\min \quad \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{2.1}$$

$$\text{s.t.} \quad \sum_{j=2}^{n} x_{1j} = m \tag{2.2}$$

$$\sum_{i=2}^{n} x_{i1} = m \tag{2.3}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j = 2,\dots,n \tag{2.4}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i = 2,\dots,n \tag{2.5}$$

$$u_i - u_j + px_{ij} \leq p - 1 \qquad 1 \leq i \neq j \leq n \tag{2.6}$$

$$x_{i,j} = \{0,1\} \qquad \forall (i,j) \in E \tag{2.7}$$

$$u_i \in \mathcal{R} \qquad \forall i = 1,\dots,n \tag{2.8}$$

$$\tag{2.9}$$

The variable x_{ij} represents that edge going from node v_i to node v_j if part of the solution if and only if $x_{ij} = 1$. The distance matrix $C = (c_{ij})$ represents distances between nodes $c_{ij} = dist(v_i, v_j)$. Constraints (2.2) and (2.3) ensure that each path from m paths leaves from the depot v_1 exactly once and returns there exactly once

as well. Constraints (2.4) and (2.5) together ensures that each node is in solution exactly once (with exception of node v_1).

The constraint (2.6) (presented by Miller et al. (1960)) is so called subtour elimination constraint (SEC) and is used to ensure, that each path will be exactly one cycle. Constant p is a maximal number of nodes on one path. The variable u_i is free variable that represents a potential for each node. For each path holds, that the potential is growing along the path, so that each node has a higher potential than the node before it, with the exception of the first node v_1. This ensures that only cycles in the solution start in the node v_0.

2.2.3 Multi-agent Simulation

Klügl (2009) presents three basic approaches to the simulation based on time advance paradigm (among others):

- continuous simulation
- discrete event simulation
- time step simulation

Continuous Simulation

The continuous simulation is basically connected to a model consisting of differential equations. This method would be too complex for multi-agent simulation used in this thesis and thus is out of scope.

Time step simulation

The time step simulation updates state periodically with given period Δt (Sislak et al., 2009). After each update, agents are given control. The value of Δt determines the resolution of the simulation. If Δt is too big, there is chance that important event in the simulation will be missed (e.g. two vessels will come into each other sensor ranges without noticing it). On the other hand having too small Δt results into lot of updates where there is no significant change of the model. The examples of the time step simulation are systems AgentC (Jakob et al., 2011) and AgentFly (Sislak et al., 2012).

Discrete Event Simulation

The discrete event simulation (DES) is based on the fact that the important state changes occur only in concrete discrete times. This change is called the event.

Between two consequent events the state variables remain the same. Therefore the simulation clock can jump directly to the time of the next event.

The fundamental element of DES is the event queue. The queue stores future events ordered by the time at which they are scheduled to occur. Each iteration loop of the DES one event is taken from the queue and it is executed. Part of the event execution might be adding the new events into the queue.

The process of creating of the new events is very important in DES. It determines which state variables are important and which are not. For example when vessel moves from point A to point B, the important events would be meeting another vessels on the journey and the arrival to point B. Therefore these events would be scheduled. Mechanism that resolves which events are created depends on the simulated domain and it creation is part of the simulation creation.

The advantage of the DES over time step simulation is that the simulation can run much faster because unimportant state changes are not computed. The disadvantage is that future outcome of every action must be defined when the action is invoked in order to put new events into the event queue.

The example of the DES simulation engine is Alite Tactical Package[1] which is running on Alite toolkit Komenda et al. (2013).

Alite Alite is a software toolkit simplifying implementation and construction of (not only) multi-agent simulations and multi-agent systems. It stands on technologies related to JVM7 ecosystem and it is mostly written in Java. The objectives of the toolkit are to provide a highly modular, flexible and open set of functionalities supporting rapid prototyping and fast implementation of multi-agent applications, mainly focusing on highly scalable and complex simulated environments. The guiding principles underlying the Alite design are i) modularity, so that the system does not commit a developer to a specific definition of concepts such as agent, environment, etc. and ii) composability, so that the various components of the toolkit can be put together in a rapid and flexible manner. In result, Alite can be seen as a collection of highly refined functional elements providing clear and simple APIs, allowing a programmer to put together relatively complex multi-agent simulation scenarios rapidly citepalite.

This work is built upon Alite toolkit. It benefits from the fact, that the Alite is written in Java and is written Scala, a blend of object-oriented and functional programming concepts in a statically typed language (Odersky et al., 2010).

[1] `http://jones.felk.cvut.cz/redmine/projects/tacticalenvironment/wiki`

Chapter 3

Formalization

This chapter formalizes patrolling problematic and describes algorithm to chosen set of problems. Section 3.1 informally introduces the key elements and actors together with basic patrolling problem parameters. In section 3.2 environment is described together with formalization of main actors in patrolling. Section 3.3 considers patrolling problems for homogeneous patrollers and describes solution algorithms to these problems. In section 3.4 modifications for heterogeneous patrollers are described. Section 3.5 discusses solved problems and shows how some other patrolling problems reduce to problems already solved in sections 3.3 and 3.4.

3.1 Breakdown and Taxonomy

This section informally describes patrolling terms used in this thesis.

In patrolling problems in this thesis three main types of actors are considered:

Target is an entity that is intended to be protected.

Attacker is trying to attack some target of getting physically close.

Patroller patrols the area in order to monitor it and/or to identify attackers.

There are several reasons to use surveillance. The following are considered: area wide information collection – area surveillance and detection of attacker actions, where attacker action can be either attack or movement or both.

The area surveillance can be used to obtain information about situation in area and to keep it up to date. Unlike the detection of attacker actions it is not limited to attacker actions and thus it gives more general information. However since it is not focused on the attacker, it is more likely that the attacker would succeed against this method. For example the area-wide surveillance would route the patroller even into

locations where the attacker has no reason to be and therefore the patroller will loose its time in such locations. During that time it is possible the attacker can attack.

The detection of attacker actions focuses specifically on the attacker only. The attack action is approach of attacker's boat to target vessel. The attack action is limited only to local approach in patroller sensor radius distance from the target vessel. The movement action on the other hand is defined as whole process of travelling from attackers origin location (point on shore) to the target vessel.

The targets protected by surveillance can be of two following types: static and dynamic. The static targets are targets that don't change their position. In this case they are represented by anchored vessels. When the vessel is moving, it is not considered as target (i.e. attacker is assumed not to be able attack moving vessel). The targets set can be changed in time (vessel anchors or lift anchors). The dynamic targets consists of the static targets together with targets that are moving (the attacker is considered able to attack moving target). The set of moving targets can also change in time when target vessel enter or leaves surveilled area.

3.2 Environment

All water units in the simulation (USVs, merchant vessels and attacker boats) operate within the *harbour water area*. The harbour water area is defined by solid polygon which in this thesis covers south bay in Mombasa so its edges represents the coastal lines.

3.2.1 Vehicles

In this thesis there are considered two types of vehicles (surface and aerial) that together represents all types of actors: patrollers, targets and attackers. Both types of vehicles are represented by the same type of model with one difference: the surface vehicle must contain their movement to water area(s) only whereas the aerial vehicles might fly over land as well.

The vehicle has following paramters:

1. current position
2. current speed
3. operating and maximal speed
4. sensor radius

Parameters 1 and 2 are time dependent whereas parameters 3 and 4 are constant.

For vehicle x these parameters are formalized: The current position of x at time t is defined by function $p(x,t)$, current speed at time t is $v(x,t)$, operating and maximal speed are defined by functions $v_{op}(x), v_{max}(x)$ respectively and sensor radius is defined by function $r(x)$. Sensor radius is maximal distance at which the vessel x can detect other vessels. The sensor can represent visual sensing by human, radar, sonar, etc.

The speed of each vehicle x when moving is usually $v(x,t) = v_{op}(x)$ but it can range[1] from $v(x,t) \in [0; v_{max}(x)]$.

Patrolling Units

The patrolling is provided by multiple patrolling units that are controlled by one patrolling agent. This layout is possible because the units operate in relatively small area and communication with central control agent is not an issue.

Each patrolling unit is equipped with sensor with radius $r(x)$. The sensor is considered ideal and thus it detects every other vessel that gets into its range. Once the attacker gets into sensor range, the patroller unit identifies it as attacker and disarms it.

Target Vessels

Target vessels represent merchant vessels. The target vessel is located in the harbour area for some time. During this time, it can either move around or be still which represents sailing in port or being anchored respectively. The target vessel has no means to prevent itself from the attacker – its protection is completely in hands of patrollers.

Attacker

Attacker means to attack one of targets. The attack takes form of getting to the same position as target and for some time τ to remain there undiscovered by patroller. This represents approaching the merchant vessel by a boat and spending time by getting on boat. The attacker has no knowledge of whereabouts of patrollers or about algorithm of their patrol. On the other hand it has information about position of all the targets. All targets hold the same value to the attacker – attacker does get the same reward for a successful attack on any of them. Given these assumptions, this leads to the optimal strategy for attacker which is: at random time attack at

[1] Speed at value 0 for aerial vehicles is managed by letting the vehicle fly in the minimum radius circle with minimum flight speed.

random target, where both random values are chosen with uniform distribution. This strategy is in Nash equilibrium (Tambe, 2011).

3.3 Patrolling Problems

This section examines several types of patrolling problems based on what is the objective of patrolling. The solutions to these problems are presented. First problem is the area surveillance problem (section 3.3.1) where optimal polynomial solution is shown. The next problem is attack detection with static targets (section 3.3.2) where new *multiple patroller path problem* is introduced and ILP formalization is presented. The last problem described in section 3.3.3 considers attack detection with dynamic targets and sub-optimal solution is shown.

Note that this section is limited to the patrollers with homogeneous parameters – patrollers with the same maximal and operational speed and with the same sensor range:

$$v_{max}(x_i) = v_{max}(x_j) \wedge v_{op}(x_i) = v_{op}(x_j) \wedge r(x_i) = r(x_j) \quad \forall i, j$$

The patrollers with heterogeneous parameters are addressed in section 3.4.

In all proposed solutions it is considered that each two patrollers can either have separate patrol paths or they share their path completely. Therefore they cannot share only part of their path which would result for example to the situation where they would alternate by visiting the shared part of the path. To solve this type of solution, the time variable and time horizon would have to be introduced which would lead to more complex problems.

3.3.1 Area Surveillance Problem

In the surveillance problem, the task is to monitor a given area with one or multiple patrollers in order keep information about situation in whole area as up-to-date as possible. This means that the information about each point in the area must be updated as often as possible. This objective leads to the minimization of the worst idleness for the whole area.

For this solution the surveilled area was discretized into grid $G = (V, E)$. Each node of the grid represents certain sub-area. As is described in 2.2.1 the triangular tiling is used in this thesis.

Note that in case of surveilled area is consisted of multiple sub-areas, the graph G is unconnected, where each component is created the same way as in case of single area/graph solution. Therefore the text bellow considers case with only one area.

The interpretation of the triangular grid graph as hexagonal tiling of the plane (in Figure 2.1) is very useful because the edge size for graph G corresponds with sensor radius of patroller $r(p_i)$. The weight (cost) of each edge $e_{j,k}(v_j, v_k) \in E$ represents the distance between v_j and v_k:

$$c(e_{j,k}) = 2\sqrt{3}r(p_i) \tag{3.1}$$

Equation (3.1) is illustrated in Figure 3.1. The hexagonal tiling is one of three "platonic" tilings (Chavey, 1989). The other tilings are tiling by triangles and tiling by squares. From these three, a hexagon is the polygon with the area closest to its inscribed circle, which represents patroller's sensor coverage, hence it is an obvious choice.

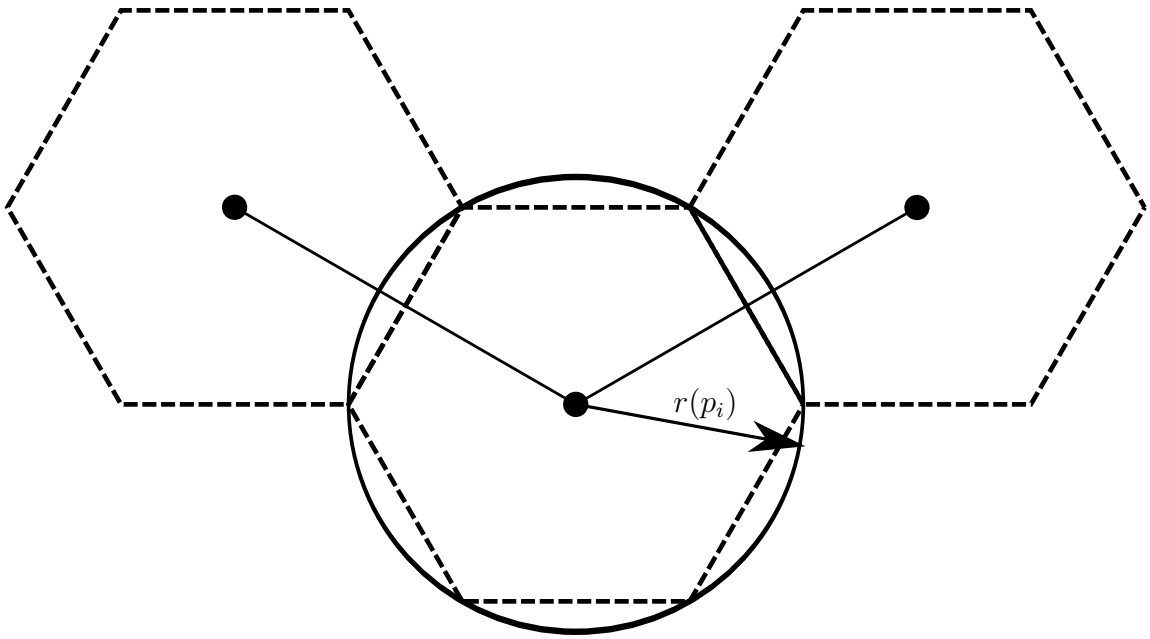

Figure 3.1: Illustration of relationship between edge cost and patroller sensor radius

For each node $v_i \in V$, function $f(v_i, t)$ representing idleness is defined:

$$f(v,t) = \begin{cases} 0 & \text{if } v \text{ is occupied by patroller} \\ t - \text{last}(v) & \text{otherwise} \end{cases} \tag{3.2}$$

The t represents current time and last(v) represents the last time when node v was visited (if the node was never visited, then last$(v) = 0$). The worst idleness for whole graph G (and therefore for the whole area) is then defined as function g:

$$g = \max_{v \in V} f(v,t) \quad \forall t \tag{3.3}$$

Since the objective is to minimize the worst idleness on a graph, this problem reduces to patrolling on graph described by Chevaleyre (2004). This problem for one

agent reduces to travelling salesman problem. For graph with equilateral edge length (or cost) the TSP is equal to the problem of finding the hamiltonian cycle (HCP) which is shown by the following Theorem 1.

Theorem 1. Let $G = (V, E)$ be a equilateral triangular grid graph and let $G_c = (V, E_c)$ be complete[2] graph such that G is factor[3] of G_c. Travelling salesman solution for the G_c is then equivalent to the hamiltonian cycle for the G.

To prove theorem 1 the following lemmas are used:

Lemma 1. *Set E is subset of E_c such as*

$$E = \{e \in E_c | c(e) = \min_{e \in E_c} c(e)\} \subseteq E_c$$

Lemma 2. *Every solution for HCP on factor graph G_{hc} is a feasible solution for TSP on its supergraph[4] G_{tsp}.*

The proof is based on showing that the both HCP and TSP solutions consist of the same set of edges with minimal weight (which is equal for all these edges) and thus they have the same value.

Proof. Let the $B_{tsp} \subseteq E_c$ be edges of the solution of the TSP problem for G_c, $B_{hcp} \subseteq E$ be edges of the solution of the HCP on G and *cost* be a function of cost of the path $cost(B) = \sum_{e \in B} c(e)$

Since B_{tsp} is optimal solution on G_c it holds that

$$cost(B_{tsp}) \leq cost(B_{hcp}). \tag{3.4}$$

The solution B_{hcp} is subset of E which by lemma 1 contains only minimal edges from E_c. This implies that

$$cost(B_{tsp}) \geq cost(B_{hcp}). \tag{3.5}$$

Equations (3.4) and (3.5) imply $cost(B_{tsp}) = cost(B_{hcp})$ and together with lemma 2 this implies that B_{hcp} is equivalent to B_{tsp} □

The HCP solution is optimal for one patroller. The value of the criterion function g is the time it takes for patroller p_i to go through all nodes in cycle B for graph $G = (V, E)$:

$$g^* = \frac{\sum_{e \in B} c(e)}{v(p_i)} = \frac{2\sqrt{3}r(p_i)|V|}{v(p_i)} \tag{3.6}$$

[2]A complete graph is a graph in which every pair of vertices is connected by an edge.
[3]A factor of G is a subgraph that has the same vertex set as G.
[4]A supergraph of graph G is a graph of which G is a subgraphs.

The first form of the equation expresses the g^* in relation to the cycle B. Since B is hamiltonian and thus contains all nodes and the distance between nodes in every path for grid graph is equal, it can be expressed in the second form of the equation where the edge size based upon the size of patrollers sensor radius is used together with number of nodes in graph.

The result of Theorem 1 is that the path is contained within the surveillance area and therefore it is unnecessary for patroller to leave. The path that would take patroller from the surveillance area would not even be optimal because it wouldn't be constructed of edges with minimal cost.

For multiple patrollers, the optimal solution is to let them share the path given by HCP. The patrollers are distributed along this path uniformly which minimizes maximal distance between them (along the path) and thus minimize maximal time between visits of the nodes.The optimum value of for this setting is the criterion g_m^*

$$g_m^* = \frac{g^*}{N}, \tag{3.7}$$

where g^* is optimal criterion value for one patroller and N is number of patrollers[5].

In this section the solution to area surveillance problem was presented. The solution is optimal for given discretization and algorithm works in polynomial time complexity even if the general form of this problem is NP-hard. This can be done because area surveillance is shown to be special case of patroll problem on triangular grid graph.

3.3.2 Attack Detection with Static Targets

In this case the goal is to detect attack on one of multiple stationary targets. In this thesis, merchant vessels on anchors are represented as stationary targets. These vessels are anchored on their position for relatively long period of time (days). At this state the attacker can easily approach them unnoticed, climb on board and steal some of their cargo.

The attacker is assumed to has no prior knowledge about the surveillance units. The attacker at random moment chooses random target and then proceeds to attack it. It is assumed that the attack takes time τ to complete. This represents approach to the target vessel, climbing on board, stealing cargo and getting away. If the attacker is spotted by the patrol during this action it is assumed that the authorities are able to prevent it from finishing it and/or from escaping.

The optimal solution to this problem would be to visit each vessel $m_i \in \mathcal{M}$ repeatedly in period shorter than τ. However this is often not feasible. In that case

[5]Parameters of these multiple patrollers are identical parameters of single patroller.

the goal to minimize the time that each target vessel was unvisited. This lead to the problem of minimizing worst idleness as defined by Chevaleyre (2004) (in section 2.1), where the nodes of graph $G = (V, E)$ are the targets positions and the graph is complete.

The criterion for this problem is the same as criterion for area surveillance problem (section 3.3.1) expressed in equation (3.3):

$$g = \max_{v \in V} f(v, t) \quad \forall t$$

But because the edges doesn't have the same weight the TSP problem cannot be reduced to search for HCP. The TSP can be solved optimally (for small instances) by integer linear programming (Miller et al., 1960).

For multiple agents, the TSP problem is a modification of the multiple travelling salesman problem (mTSP). Modified version of ILP solution to the TSP was introduced by Bektas (2006, page 213) to solve mTSP. This formalization requires to define a depot (or multiple depots) from which the salesmen will depart and where they will return. However in the problem of the attack detection this constraint of defined depots is undesirable. Hence this problem has to be reformulated.

The problem is defined on a complete graph $G = (V, E)$. V is the set of $n + 1$ nodes where $v_i, i \in [1, n]$ is node corresponds to the position of the i-th target and v_0 is a virtual node. The edges weights represents distances between the nodes in associated matrix $C = (c_{ij})$:

$$c_{ij} = \begin{cases} \text{dist}(v_i, v_j) & 1 \leq i \neq j \leq n \\ 0 & \text{otherwise} \end{cases}, \tag{3.8}$$

where $\text{dist}(v_i, v_j)$ represents distance between nodes v_i and v_j. If the edge leads from or into node v_0 the distance is then $c_{i0} = c_{0j} = 0 \; \forall i, j$, therefore the virtual node v_0 is always within zero distance from each node in graph.

The linear program to find paths for m salesmen which originate and end in node v_0 was created by modifying the program introduced by Bektas (2006) (see section 2.2.2):

$$\min \quad \sum_{i=0}^{n} \sum_{j=0}^{n} c_{ij} x_{ij} \tag{3.9}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} x_{0j} = m \tag{3.10}$$

$$\sum_{i=1}^{n} x_{i0} = m \tag{3.11}$$

$$x_{0j} \leq x_{j0} \qquad \forall j = 1, \ldots, n \tag{3.12}$$

$$\sum_{i=1}^{n} x_{ij} \leq 1 + Mx_{0j} \qquad \forall j = 1, \ldots, n \tag{3.13}$$

$$\sum_{i=1}^{n} x_{ij} \geq 1 - Mx_{0j} \qquad \forall j = 1, \ldots, n \tag{3.14}$$

$$\sum_{i=1}^{n} x_{ij} \leq 2 + M(1 - x_{0j}) \qquad \forall j = 1, \ldots, n \tag{3.15}$$

$$\sum_{j=1}^{n} x_{ij} \leq 1 + Mx_{i0} \qquad \forall i = 1, \ldots, n \tag{3.16}$$

$$\sum_{j=1}^{n} x_{ij} \geq 1 - Mx_{i0} \qquad \forall i = 1, \ldots, n \tag{3.17}$$

$$\sum_{j=1}^{n} x_{ij} \leq 2 + M(1 - x_{i0}) \qquad \forall i = 1, \ldots, n \tag{3.18}$$

$$u_i - u_j + px_{ij} \leq p - 1 + Mx_{0j} \qquad \forall i, j : 1 \leq i \neq j \leq n \tag{3.19}$$

$$x_{i,j} = \{0, 1\} \qquad \forall (i, j) \in E \tag{3.20}$$

$$u_i \in \mathcal{R} \qquad \forall i = 1, \ldots, n \tag{3.21}$$

$$\tag{3.22}$$

The optimized binary variable x represents set of edges for the mTSP. If $x_{ij} = 1$ then edge going from v_i to v_j is in the solution. Constraints (3.10) and (3.11) represents the fact that every path starts at node v_0 (3.10) and ends in there as well (3.11). The constrain (3.12) ensures that if path start by edge $e(0, j)$ then it ends by edge $e(j, 0)$. This would actually end up in producing solution, that is not a set of simple paths. Each path contains twice the node called *primary* which is node that is in each resulting path second after start node v_0. In Figure 3.2 these are the nodes v_1 and v_5. These primary nodes actually represent the depot nodes from original formulation of mTSP.

Constraints (3.13), (3.14) and (3.15) ensures that for each node enters the path exactly once if this node is not primary (3.13, 3.14) or if the node is primary then the path enters it at most twice at least once (3.15). Constraints (3.16), (3.17) and (3.18) work in similarly for edges leading from the node.

This program gives optimal results for multiple travelling salesman problem without given depot.However this solution is not optimal for criterion (3.3). For example in Figure 3.3 the optimal solution for two agents is one cycle through nodes $\{v_1, \ldots, v_9\}$ and second with only node v_{10}. This is because the optimal solution for this variation of mTSP is to find multiple cycles with minimal sum of their lengths. But to solve the patrolling problem, the task is to find multiple cycles with minimal sum of their lengths and with minimal difference between their lengths.

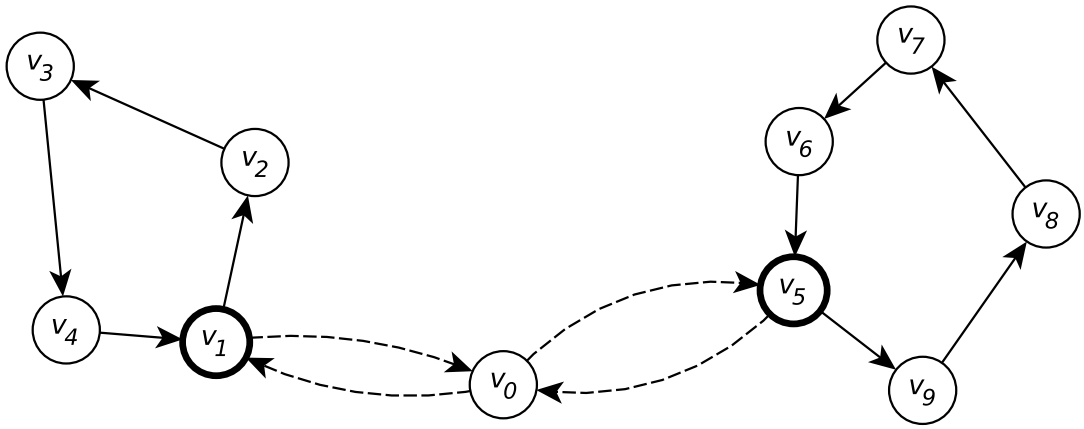

Figure 3.2: Example of multiple TSP problem solution. Nodes v_1 and v_5 are called primary.

Because of that the ILP formulation must be redefined to new *multiple patrollers path problem (MPPP)* formulation:

$$\min \quad s \tag{3.23}$$

$$\text{s.t.} \quad \sum_{i=0}^{n}\sum_{j=0}^{n} c_{ij}x_{ij}^{(k)} \leq s \qquad \forall k = 0,\dots,m \tag{3.24}$$

$$\sum_{j=1}^{n} x_{0j}^{(k)} = 1 \qquad \forall k = 0,\dots,m \tag{3.25}$$

$$\sum_{i=1}^{n} x_{i0}^{(k)} = 1 \qquad \forall k = 0,\dots,m \tag{3.26}$$

$$x_{0jk} \leq x_{j0}^{(k)} \qquad \forall j = 1,\dots,n, k = 0,\dots,m \tag{3.27}$$

$$\sum_{k=0}^{m}\sum_{i=1}^{n} x_{ij}^{(k)} \leq 1 + M\sum_{k=0}^{m} x_{0j}^{(k)} \qquad \forall j = 1,\dots,n \tag{3.28}$$

$$\sum_{k=0}^{m}\sum_{i=1}^{n} x_{ij}^{(k)} \geq 1 - M\sum_{k=0}^{m} x_{0j}^{(k)} \qquad \forall j = 1,\dots,n \tag{3.29}$$

$$\sum_{k=0}^{m}\sum_{i=1}^{n} x_{ij}^{(k)} \leq 2 + M(1 - \sum_{k=0}^{m} x_{0j}^{(k)}) \qquad \forall j = 1,\dots,n \tag{3.30}$$

$$\sum_{k=0}^{m}\sum_{j=1}^{n} x_{ij}^{(k)} \leq 1 + M\sum_{k=0}^{m} x_{i0}^{(k)} \qquad \forall i = 1,\dots,n \tag{3.31}$$

$$\sum_{k=0}^{m}\sum_{j=1}^{n} x_{ij}^{(k)} \geq 1 - M\sum_{k=0}^{m} x_{i0}^{(k)} \qquad \forall i = 1,\dots,n \tag{3.32}$$

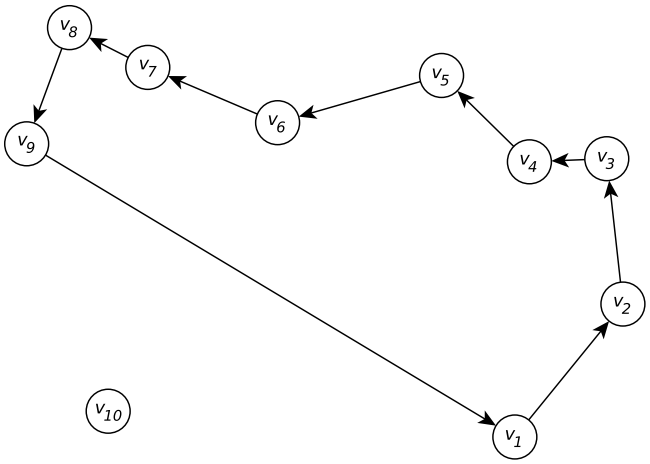

Figure 3.3: Example of wrong solution using mTSP solver.

$$\sum_{k=0}^{m}\sum_{j=1}^{n} x_{ij}^{(k)} \leq 2 + M(1 - \sum_{k=0}^{m} x_{i0}^{(k)}) \qquad \forall i = 1, \ldots, n \quad (3.33)$$

$$u_i - u_j + p x_{ij}^{(k)} \leq p - 1 + M x_{0j}^{(k)} \qquad \forall ij : 1 \leq i \neq j \leq n, k = 0, \ldots, m \quad (3.34)$$

$$x_{i,j}^{(k)} = \{0, 1\} \qquad \forall (i,j) \in E, k = 0, \ldots, m \quad (3.35)$$

$$u_i \in \mathcal{R} \qquad \forall i = 1, \ldots, n \quad (3.36)$$

$$M \in \mathcal{R} \qquad \text{Arbirtary big number} \quad (3.37)$$

The change from the previous definition is in the criterion, which is now formalized using variable s with constraint (3.24). This pair expresses criterion

$$\min \max_{k \in [0;m]} \sum_{i=0}^{n}\sum_{j=0}^{n} c_{ij} x_{ij}^{(k)}$$

which is not linear in this form.

The second change is that the individual paths are represented by index (k). The binary variable $x_{ij}^{(k)}$ then represents that k-th agent's path will go through edge $e(i,j)$ if $x_{ij}^{(k)} = 1$. All other constraints were changed accordingly. The total number of constraints for MPPP is $mn^2 - 2mn + kn + 6n + m + 2k - 6$.

Solution described above is optimal under the assumption that each patroller has its own path that is not shared with any other patroller. If this assumption is relaxed to the assumption that each patroller can has its path either completely separate or it share it entirely with another patroller, than this solution might not be optimal.

It can be demonstrated on the graph where nodes are placed in circle (Figure 3.4. The solution for two agents is depicted in Figure 3.5. However if the solution for one agent is computed (Figure 3.6) and both agents are put on this path with equal

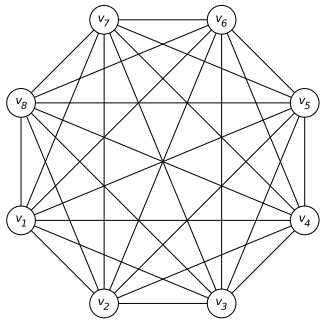

Figure 3.4: Circle complete graph.

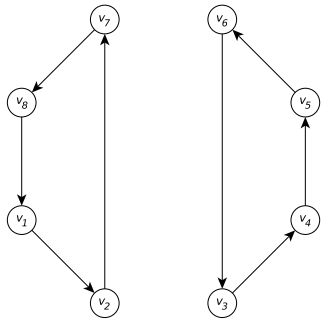

Figure 3.5: Solution for the circle graph for 2 paths.

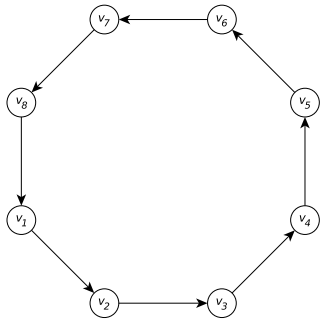

Figure 3.6: Solution for the circle graph for 1 path.

distance between themselves (on the opposite side of the circle in this case) then the solution outperforms the solution with two separate paths. Hence for some graphs instances it might be better to place more agents on one path with uniform distance.

To solve this problem for m agents it is necessary to examine all possible combinations of division of the agents into paths. Expression of all these combinations can be done using partition in from number theory (Andrews, 1998). (Integer) partition is a way to write positive integer n as a sum of positive integers. If two sums differ only in order of their summands they are considered the same. For example number 4 can be written in 5 ways:

$$
\begin{aligned}
4 &= 4 \\
4 &= 3 + 1 \\
4 &= 2 + 2 \\
4 &= 2 + 1 + 1 \\
4 &= 1 + 1 + 1 + 1
\end{aligned}
$$

The number of partitions is expressed by partition function $p : \mathcal{N}_0^+ \mapsto \mathcal{N}^+$ which grows very fast. For example[6] $p(5) = 7$, but $p(10) = 42$. This might result into huge number of computations for bigger number of agents, but because the number of commercial vessels anchoring in Mombasa harbour ranges from 20 to 30[7], more then 5 surveillance units for this task would be an excessive measure. In that case some suboptimal algorithm could be used because it is likely that the solution would resolve into a path, that would have a period lower than τ.

To incorporate partition the ILP problem formulation for MPPP must be

[6]Data were taken from The Online Encyclopedia of Integer Sequences (see `http://oeis.org/A000041/`)

[7]Based on observation done using website `http://www.marinetraffic.com/`.

changed. However the only change is in constraint (3.24) which would be changed into

$$\sum_{i=0}^{n}\sum_{j=0}^{n} c_{ij}x_{ij}^{(k)} \leq d_k s \quad k = 0, \ldots, m_d, \tag{3.38}$$

where $d = (d_1, \ldots, d_{m_d})$ is a distribution vector representing one of the partitions (e.g. $(2, 1, 1)$). The variable m in program is then change into $m_d = |d|$ in whole ILP representation.

The algorithm to find the optimal solution is written as algorithm 1. The algorithm goes through whole distribution of all partitions and selects best result. During the iteration the algorithm uses previous results as bound values for ILP formulation to quicken the computation.

Algorithm 1: Multi path patroller problem by ILP

```
Input: Targets positions positions, number of surveillance units n
Output: Optimal bestPaths, agent distribution bestDist

distributions ← getAgentDistributions(n)
bestS ← ∞
foreach dist ∈ distributions do
    paths, s ← runILP(positions, dist, bestS)
    if s < bestS then
        bestS ← s
        bestDist ← dist
        bestPaths ← paths
    end
end

return bestPaths, bestDist;
```

In this section problem of attack detection with static targets is addressed. It is shown, that this problem reduces to TSP for one patroller but for multiple patrollers new formulation of multiple patroller problem must be created. The problem for given agents distribution is formalized as ILP. The resulting algorithm searches over all possible distributions to find an optimal solution.

3.3.3 Attack Detection with Dynamic Targets

Attack detection with dynamic targets is very similar to attack detection with static targets (section 3.3.2). The only difference is that the target's speed can vary in time $v(m, t) \in [0; v_{max}(m)]$. The planned trajectory of the target m for a reasonable future is known to the surveillance authority but not to the attacker. The reasonable future in this case is that the plan is known to the point when merchant vessel stops to anchor for several days. In that event the target becomes stationary and until it

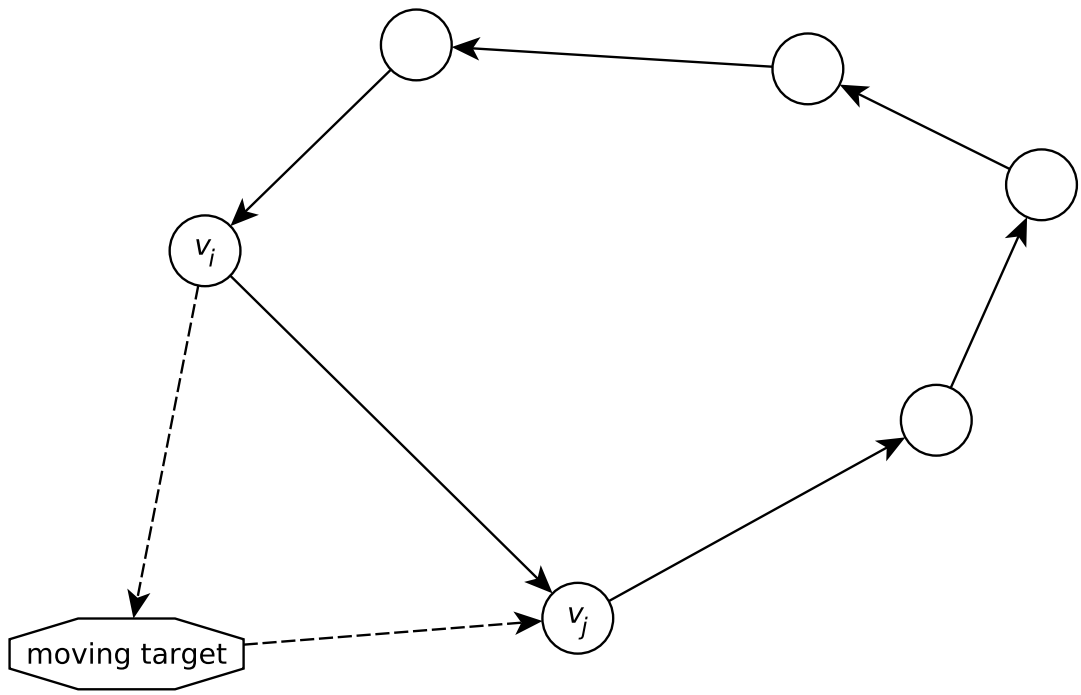

Figure 3.7: Example of modification to path with respect to the moving target

decides to move again. If the target leaves the harbour and sails to the open sea, it is not considered as a target any more.

The solution to this problem is again to minimize worst idleness as in sections before (3.3.1, 3.3.2) but unlike in case with static targets the edge weights $c(e)$ of the graph $G = (V, E)$ are function of time. In general case a optimal solution to this problem might be very difficult, but the Mombasa harbour traffic is very unique. The movement in harbour is very scarce and most of the time there is none at all. If some vessel is moving, it is typically only one or two vessels.

The proposed algorithm for this problem exploits the fact that for static targets optimal solution can be found, and adjust patrol plan to incorporate a moving target into.

The algorithm examines all possible modifications to planned paths for static targets. Modification in this case is determined by patroller x and consecutive points v_i and v_j on its path. The modification to the patroller's path has a form of adding the moving target as a waypoint into this path between points v_i and v_j. The example of the modification is depicted in Figure 3.7. Possible modification is also that the moving target would be ignored.

These modifications are sought on the time horizon Θ that is equal to maximum time that would take each agent to go around its path. For each of these modifications how the criterion would be changed is calculated and the modification which increase the criterion least is chosen. This is repeated each time after simulation time Θ.

During its path from point v_i to v_j through moving target's position the patroller x speeds up to its maximal speed[8] $v_{max}(x)$ in order to minimize delay from the situation when it would follow its path without visiting the moving target. If x shares its path with other patrollers, they slow down to keep uniform distance between each other along the path.

3.4 Modification for Heterogeneous Patrollers

This section contains modifications of algorithms described in section 3.3 to algorithms that work with heterogeneous patrollers. Heterogeneous patroller set is a set of m patrollers $x_1, \ldots, x_n$ for which holds that at least one pair of patrollers doesn't have equal operating parameters v_{op}, v_{max} or doesn't have equal mobility capability.

Mobility capability is capability of patroller x_i to move on different areas. In this thesis it is represented by unlimited movement of aerial vehicles versus limited to water only movement of surface vehicles.

Note that in this section the set of patrollers with different sensor radius r is not considered, since it is out of scope of this thesis.

3.4.1 Area surveillance Problem

In the area surveillance problem, the only obstacle to use heterogeneous patrollers is their speed capability difference. As was said in section 3.3.1 multiple (homogeneous) patrollers are during the surveillance distributed uniformly along the patroll path. However, with different speed, this would result in breaking the uniform distribution of patrollers along the path (faster agents would be overtaking the slower ones). On the other hand, the incapability of surface units to go over shore is not an issue, because area surveillance path is contained inside surveilled area.

The patrolling units are grouped by its speed capability into d groups, where d is number of unique operating speed properties from patrolling unit set. This forms the set of tuples $\mathcal{D} = \{(m_1, v_{op,1}), \ldots, (m_d, v_{op,d})\}$ where m_i is the number of units with operating speed $v_{op,i}$.

To solve this problem for d patrol groups the patrol grid G must be partitioned into d grids $G_1 = (V_1, E_1), \ldots, G_d = (V_d, E_d)$ that are its subgraphs $G_i \subseteq G$. These subraphs are induced by their vertex sets $G_i = G[V_i]$.

One property of this partitioning is given by the objective that criterion g_i^* for all subgrids must have the same value $g_1^* = \ldots = g_d^*$. This leads to partitioning

[8]Keeping maximal speed requires large fuel consumption, hence the patrollers can keep it only in rare occasions.

property:

$$|V_i| = \frac{m_i v_{op,i}}{\sum_{j=1}^{d} m_j v_{op,j}} |V| \tag{3.39}$$

The second property is that resulting graph must be solid (section 2.2.1). This can be done by using breath first search algorithm to generate the subgrids.

For each subgrid G_i, the path is calculated by the algorithm described in section 3.3.1 and it is assigned to i-th patrol group.

3.4.2 Attack Detection with Static Targets

In multiple patroller path problem both, properties of heterogeneous patroller set (speed difference and different mobility domain) form problem. The homogeneous formulation (section 3.3.2) doesn't consider speed at all and it incorporates the beeline distance.

Both of these issues can be solved by appropriate modification of distance matrix $C = (c_{ij})$ defined by equation (3.8). The matrix is now defined for each unit and $C_k = (c_{ij}^{(k)})$ represents matrix for k-th unit.

$$c_{ij}^{(k)} = \begin{cases} \text{time}(v_i, v_j, k) & 1 \leq i \neq j \leq n \\ 0 & \text{otherwise} \end{cases}$$

The function $\text{time}(v_i, v_j, k)$ returns time that would take journey of k-th unit from node v_i to v_j. The linear program formalization is then updated in its one constraint set originally described by equation (3.38):

$$\sum_{i=0}^{n} \sum_{j=0}^{n} c_{ij}^{(k)} x_{ij}^{(k)} \leq d_k s \quad k = 0, \ldots, m_d \tag{3.40}$$

3.5 Discussion and Summary

In this chapter solutions to three patrolling problems were discussed: area surveillance (section 3.3.1), attack detection with static targets (section 3.3.2) and attack detection with dynamic targets (3.3.3). Furthermore in section 3.4 modification to these algorithms to work with heterogeneous set of patrollers were discussed.

All solutions work under the assumption that every two patrollers can either share their path completely or their path are disjunct. This assumption was introduced to simplify the problem, because otherwise time variable and time horizon would have to be used which would expand state space significantly.

In area surveillance the surveilled area was discretized into a hexagonal grid. The solution is based on finding a hamiltonian cycle on this grid in polynomial time.

It was shown that in the area surveillance problem if the patroller(s) doesn't need to leave the surveilled area in optimal solution. This was leveraged in solution for heterogeneous patrollers where surveilled area was split into multiple areas and each was assigned to one group of patrollers.

For the attack detection with static targets, the novel multiple patroller path problem was introduced and formalized using ILP. It was shown that the problem can be expressed for heterogeneous patrollers with the same complexity. This solution was furthermore used in the attack detection with dynamic targets where simple algorithm was proposed. It was shown that in the harbour security, the number of moving targets is minimal (between 1 and 2 but in most time there is none moving target at all) and thus this problem is less relevant.

The authority in charge of harbour security might want to solve another types of patrolling problems as well. Following paragraphs explain that these problems actually reduce to problems already discussed.

Limited range attacker is an attacker that cannot operate in whole harbour area. In this case the the patrollers area of interest is simply trimmed by the availability area of the attacker.

Detection of attacker movement with static targets. In this problem an area from which the attacker can start its attack is considered (for example part of the coastal line). This area, together with the position of static targets, forms polygon (in Figure 3.8) in which the attacker may be located. Therefore the task is to monitor this area which reduces to area surveillance problem.

Detection of attacker movement with dynamic targets is the same as detection of attacker movement with static targets with difference that the surveilled area is generated from attacker origin area together with target movement plan as illustrated in Figure 3.9.

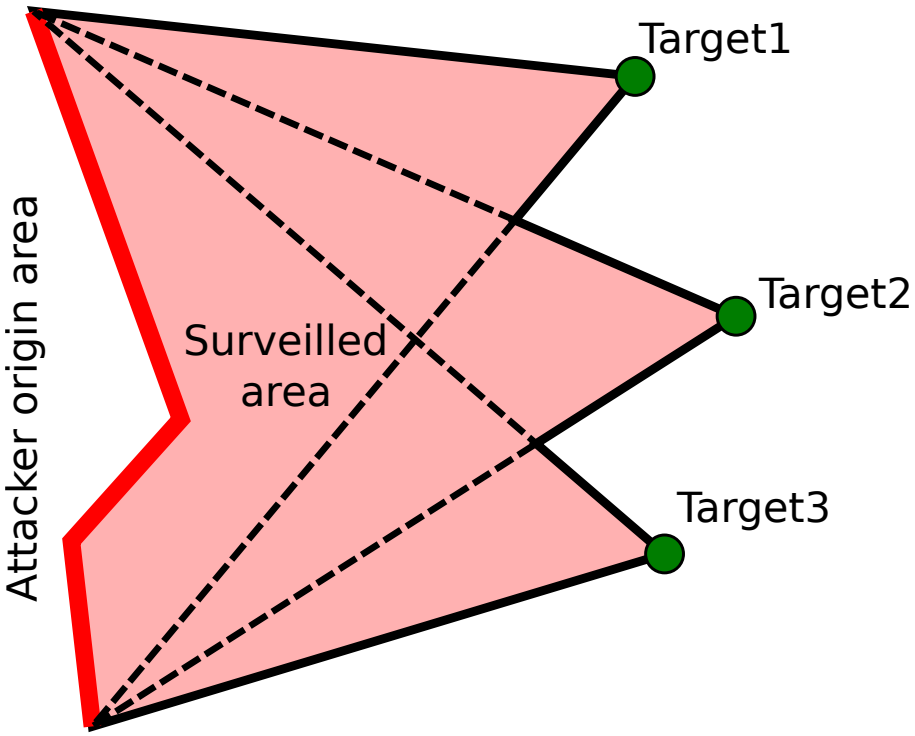

Figure 3.8: The situation for detection of attacker movement with static targets.

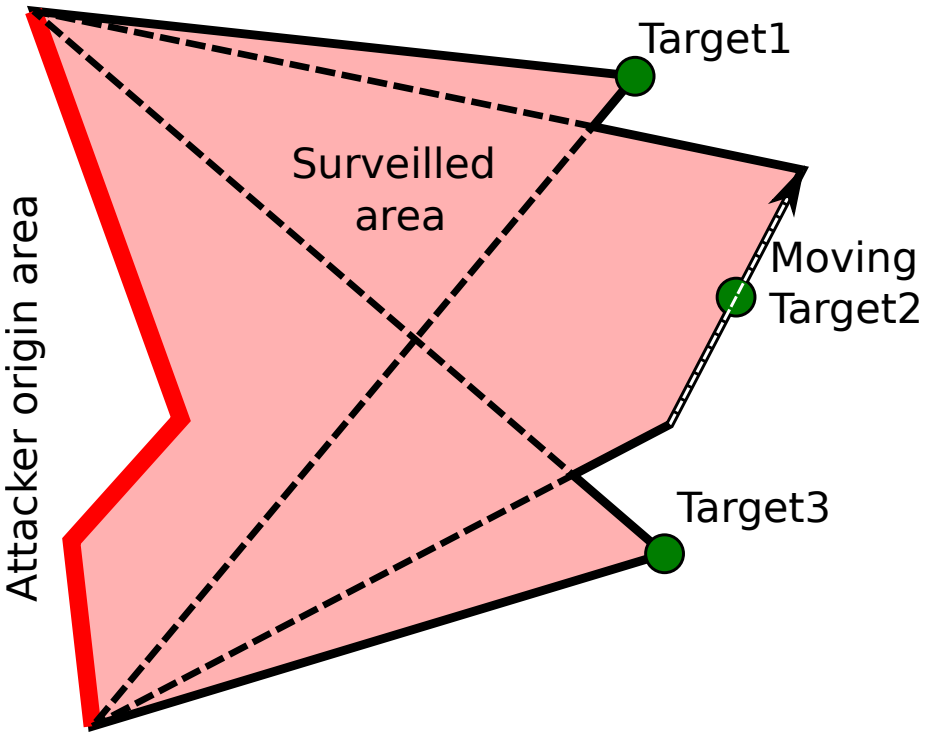

Figure 3.9: The situation for detection of attacker movement with dynamic targets.

Chapter 4

Modelling of Mombasa Harbour

Algorithms described in chapter 3 are developed for strict mathematical model. However the real world situation is more complex. To validate the algorithms in more complex environment the simulation of Mombasa harbour was developed. Even though the simulation is still imperfect it is more complex that the model upon which are the algorithms built and thus it gives better insight.

This chapter describes the modelling of the Mombasa harbour simulation. In section 4.1 the simulation engine structure is described. Section 4.2 covers modelling of Mombasa landscape. Section 4.3 describes the agent part of the simulation.

4.1 Simulation Engine

The simulation engine is a program that executes the simulation on given input data. This section describes the general building blocks of the engine:

- Storages
- Models
- Controllers
- Agents
- Mediator

The building blocks are depicted in Figure 4.1 together with their relations.

The storage is data structure responsible for keeping actual state variables of simulated entities. The state variables are those variables that describe the physical part of the virtual simulated world. In this thesis state variables are everything besides agent's mind state.

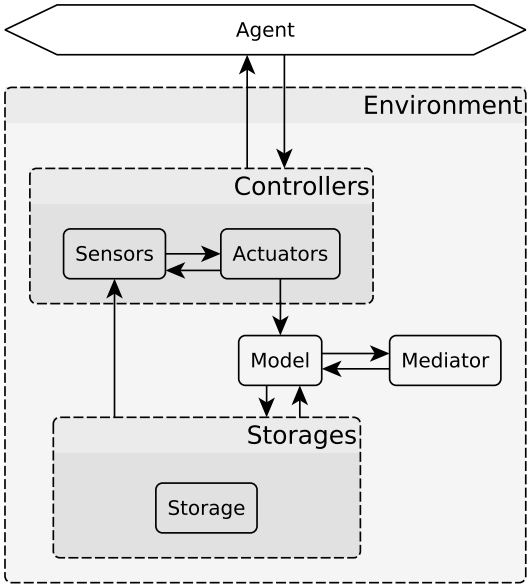

Figure 4.1: Simulation engine architecture

The model describes dynamic of the system. For a given action call and current state model produce a new state. As shown in Figure 4.1, the model is the only block that is capable of creating changes inside storage (arrow pointing to the storage).

The controller is consisted of two blocks: sensors and actuators. The sensors are used to perceive state variables from storages. Sensor is also responsible for filtering the information, so that its entity can obtain only information which it can perceive in the virtual world (for example that ship can only obtain position of other boats within its radar range). Sensor can also contain model of a real world sensor. In this thesis the sensor is ideal – it passes through informations that are consistent with storage data.

The actuators are responsible for translating action inputs from agent to model. Typically the agent create high-level actions, actuator (with data from sensor) translates them into low-level action for model. Actuator can then use data from sensor to create feedback control of the action.

The agent is an program that controls some entity (or entities) through controller. Informally the agent is the "mind" that controls the body represented by entity. It can obtain information about the environment through sensors and it can modify the environment through actuators.

The last block on the list is the mediator. The mediator is a special type of model that is focused on determining how action done by one model affects entities controlled by another models. In this thesis is implemented the cookie cutter mediator which is described in following section.

4.1.1 The Cookie Cutter Mediator

In modelled domain (harbour traffic) one of the most important issue with DES is how to represent the fact that one entity (e.g. vessel) gets into sensor range of another entity (vessel). To solve this problem there must be part of the program that resolves when (if even) these two vessel will get into their sensor ranges. This entity is called the mediator. In the case when sensor range is circle, it is called the cookie cutter mediator.

Buss and Sánchez (2005) present solution for one stationary sensor with range R and one vessel moving directly from point x at time $t = 0$ with constant velocity and direction give by vector v. The equation is based on solving quadratic equation. The times of vessel entering the range are

$$t_{1,2} = -\frac{x \cdot v}{|v|^2} \pm \frac{\sqrt{|v|^2(R^2 - |x|^2) + (x \cdot v)^2}}{|v|^2}. \tag{4.1}$$

There are 4 possible outcomes of this equation. Figure 4.2 depicts these outcomes based on the point of origin labeled by letters (A-D) (Buss and Sánchez, 2005):

Both roots positive (A) The sensor's range will be entered after a delay of the smaller root and exited after a delay of the larger root. In Figure 4.2, this corresponds to a vessel starting at point A heading through C.

One positive and one negative root (B). The vessel is already within the sensor's range and will exit after a delay of the positive root. In Figure 4.2, the vessel starts at B and proceeds through C. In case of equality of the roots, the vessel will be on a course tangent to the range ring.

Both roots negative (C). The vessel is outside the sensor's range and is moving away from the sensor. The vessel will never enter the sensor's range. In Figure 4.2, the vessel starts at point C and heads away from the sensor.

No real roots (D). The vessel will never enter the sensor range. In Figure 4.2, the vessel starts at point D and proceeds in a direction which completely misses the sensor's ring.

For valid t_1 and t_2 the *InRange* event and *OutOfRange* event are put into event queue respectively with the times. If either one of t_1 or t_2 is not valid the corresponding event is omitted. These events informs the sensor that vessel entered or leaved its sensor range. The sensor can then react on this information as it wants to (e.g. it can start sampling the vessel movement upon entry, which would result into usage of another mediator).

The modification for two moving vessels is simple. One vessel is considered stationary and vector of the second one is computed accordingly. In the coordinates

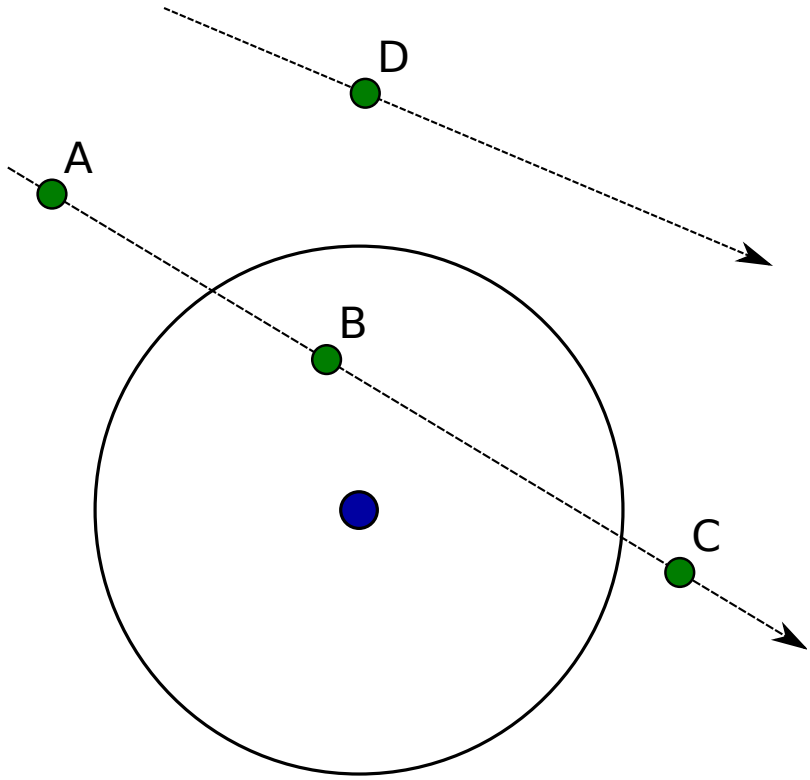

Figure 4.2: All possible solutions for the equation (4.1), picture based on the figure created by Buss and Sánchez (2005, Figure 2).

of the first vessel, the position x of the second one is (Buss and Sánchez, 2005):

$$x = (x_2 - x_1) + t(v_2 - v_1), \tag{4.2}$$

where x_1, x_2 is the starting point of the first and second vessel respectively, v_1, v_2 are their speed and t is current time. x is then the position of the second vessel relative to the first one.

However the problem occurs when one vessel during its movement changes its direction and/or velocity. In this case all future scheduled events related to this vessel need to be cancelled. This can either be done by removing these events from the event queue or by leaving the events in the queue and just disabling them. Because the Alite toolkit does not allow to remove events from the queue the second approach is used.

4.2 Mombasa Port

The largest part of Mombasa port is the The Kilindi Harbour located in a bay south of the Mombasa Island. The harbour water area was imported into the

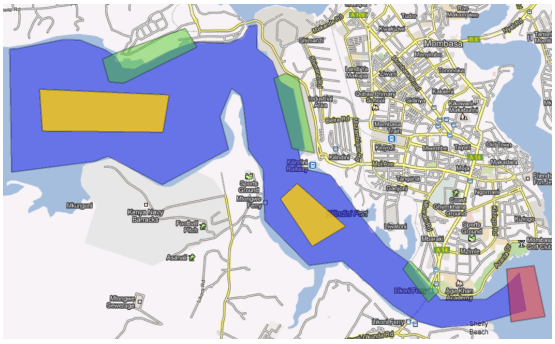

Figure 4.3: The harbour situation. Blue color represents the harbour area, red the entry area, green the port area and yellow the rest area.

Figure 4.4: Snapshot of anchored vessels from `http://www.marinetraffic.com/`.

simulation using kml[1] data format. The area (selected by hand) is depicted in Figure 4.3 by the blue color.

Part of the simulation environment are data about various types of areas in Mombasa port:

Port areas are located at bank at places where real vessels port.

Entry area is located on the east side of harbour water area and represents location where new commercial vessel agents are created representing arrival of the vessel into the harbour.

Rest areas are places in deep part of the harbour where vessels are anchored waiting for miscellaneous events (e.g. permission to leave, release of port ...)

These areas are also loaded into simulation using kml data format. They are depicted in Figure 4.3. Green color represents port area, yellow represents rest area and red area is entry area. These areas are trimmed by the harbour area (blue color) during the simulation initialization which is also the simulation area. For comparison the snapshot of positions of real world vessels is depicted in Figure 4.4.

4.3 Agents

As was said in section 4.1 the agent is responsible for controlling one or multiple entities in the simulation through controller(s). This section describes how agents in this thesis are implemented.

In the simulation the agent code is called only as a reaction to the event that is related to them, because all agent's actions are reactions to external events. This

[1]Keyhole Markup Language. `http://www.opengeospatial.org/standards/kml/`

holds even for a proactive agent, where the initial action that would start whole reasoning process can be interpreted as reaction to an external event which is introduction of this agent into the system. Then each action is a result of reasoning over agent's intentions and informations from sensors.

4.3.1 Activity Behaviour Model

The presented activity behaviour model (ABM) is based on the behaviour model represented using finite-state machines (FSM) used by Jakob et al. (2011) in AgentC platform. The activity model differs from FSM model in several ways.

The FSM model is designed to work in time step simulation. Each state has a callback method. This method is called on active state when the agent's time step is running. However in DES the simulation loop is not done by calling periodical giving control to all entities, but by giving them control only when an appropriate event happens. Therefore the ABM is designed that each activity[2] is registered to the type of events upon which it is designed to react on. The ABM manager than ensures that the active activity is invoked only when correct event occurs.

The second difference is that in the FSM the behaviour of each state was fixed. In ABM an activity can be either single activity or it can contain whole graph of another activities. This way the activity, like for example reaction to the external event, can be consisted of multiple activities which results into the possibility of creating more complex behaviour models.

4.3.2 Commercial Vessel Agent

The commercial agent represents all commercial ships visiting the harbour (cargo ships, tankers ...). The agent's life cycle represents its behaviour in harbour since arrival to exit:

1. enter harbour – the vessel enters the harbour from the open sea. In simulation this is represented by creation of the agent in the entry area.

2. (optional) wait at the rest area – the vessel stops at the rest area for random time.

3. port at the bank – the vessel port in the port area for random time

4. (optional) wait at the rest area

[2]Activity is equivalent to the state in FSM. It was renamed because it caused confusion with the state as state variable.

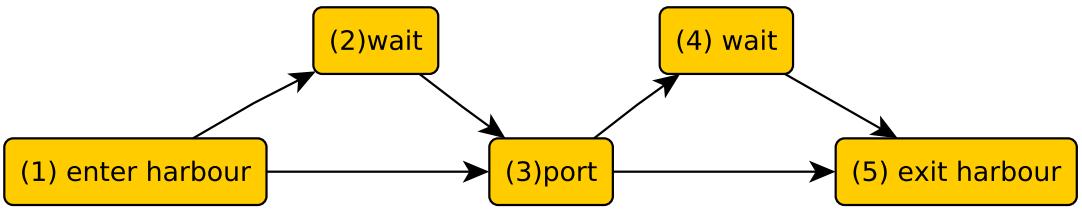

Figure 4.5: Commercial agent life cycle schema

5. exit harbour – the vessel leaves from the harbour to the open sea. This is represented by agent moving to the entry area where it is removed from the simulation.

The agent ABM graph is depicted in Figurefig:commercial. The number at the nodes label corresponds with the numbering of the activities above. The arrows represents possible activity changes.

4.3.3 Patrolling Agents

The patrolling agents patrolls along path calculated using algorithms described in chapter 3. Agents react only on approach to the robber agent. In this case, the robber agent is deleted from the simulation (representing arrest of the robber) and the event is recorded.

The patrolling agents are of two types: UAV and USV. The only differences between them is the speed (USV is considerably slower) and in their mobility capability – USV is limited to water areas only. As for the algorithm used, both of these agents are controlled by the same algorithms (see chapter 3).

4.3.4 Robber Agent

The robber agent represents adversary in the simulation. Robbers steal cargo from commercial vessels using boats. The targets are vessels at anchor.

The robbers life-cycle is follows

1. Wait random time at shore.
2. Choose random anchored target and start moving towards it.
3. (a) If target start moving go to 2.
   (b) When at target, board on it.
4. Go back to shore.

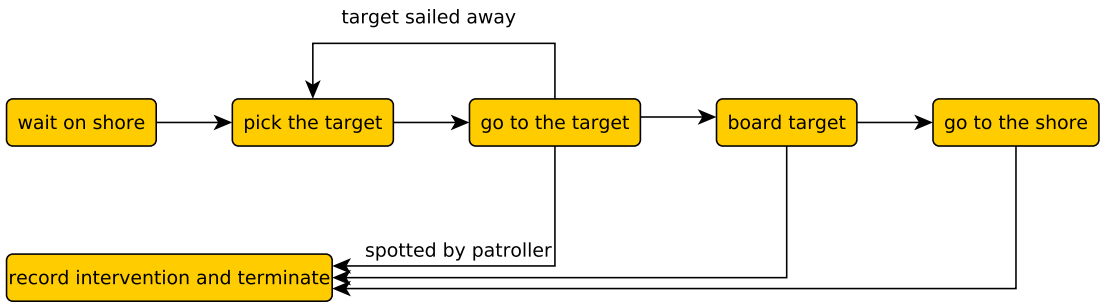

Figure 4.6: Robber agent life cycle schema

The agent's life cycle is depicted on schema in Figure 4.6. On schema is added activity transition for the situation when the robber is spotted by the patroller. In this case, the patroller disarms the robber which is represented by terminating the agent and removing it from the simulation.

Chapter 5

Evaluation

In this chapter the evaluation of presented algorithm is described. In section 5.1 the performance of ILP for MPPP is measured on random graph. Section 5.2 describes algorithm performance in simulation.

5.1 MPPP ILP Formulation Performance

This section focuses on performance of MPPP problem solved by ILP solver. All data were measured on random instances of targets position. The MPPP was tested with additional constrain that each patroller patrols its own path that is not shared with any other.

Experiments were run on AMD FX-8150 8-core processor with 16 GiB RAM. As ILP solver was used IBM Cplex version 12.4.

5.1.1 Deployment to Saturation Ratio

The performance of the MPPP problem is dependent on multiple parameters. Specifically: number and position of targets and number of the patroller. The question is what combination is computationally worst to solve, because it would be reasonable to do other tests on this hard instances to get worst-case scenario data.

Evaluation in this section is based on study of the Jain et al. (2012). They introduce the concept of the *deployment-to-saturation* $(d : s)$ ratio. The $(d : s)$ ratio is defined in terms of defender resources. Furthermore they show that in various Stackleberg security games (Tambe, 2011) the hardest case is with $(d : s) = 0.5$ – the computational pattern tends to be easy-hard-easy as the $(d : s)$ ratio increases from 0 to 1. Even though the MPPP is not Stackleberg security game it is reasonable to assume, that there would be similar pattern because it has similar characteristic which is defender (patroller) protecting multiple targets against attacker.

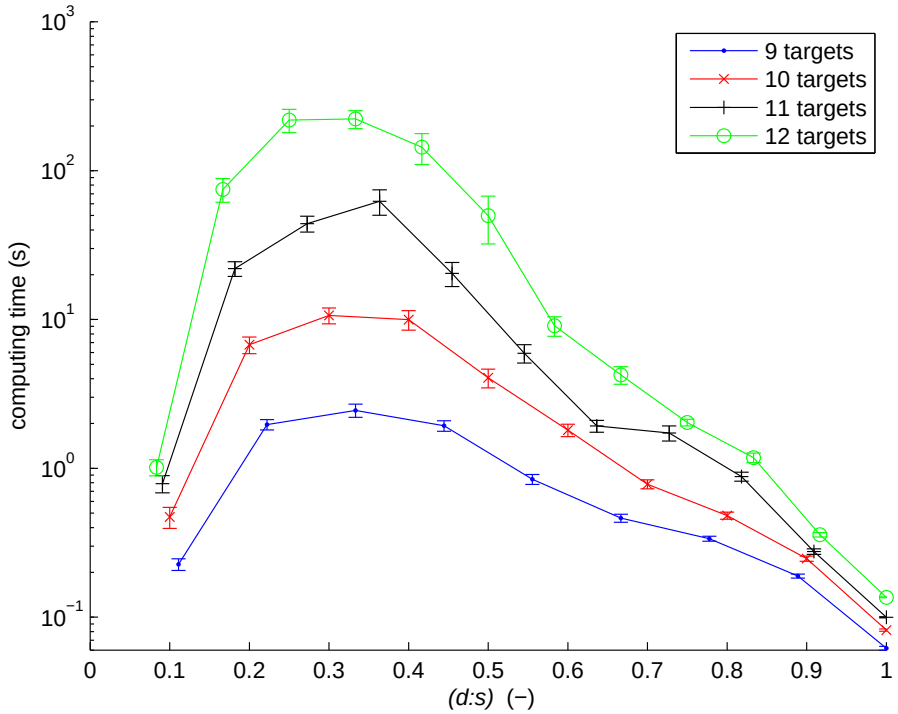

Figure 5.1: Deployment to saturation ratio performance for MPPP.

The experiment were conducted for scenarios with 9, 10, 11 and 12. For each scenario the test was run for number of patrollers in range from 1 to m, where m is the number of targets in scenario. The results, which were averaged over 50 instances, are depicted in Figure 5.1. The x-axis contains $(d : s)$ ratio, which is ratio between the number of patrollers versus number of targets, the y-axis contains runtime in seconds. Note that the y-axis is in logarithmic scale.

From the Figure 5.1 it is clear that the worst cases are these with $(d : s) \approx 0.35$. Cases with minimal $(d : s)$ are problems identical to the single TSP. On the other hand cases with $(d : s)$ are easily solved by setting one patroller for each target.

The measured maximal value $(d : s) = 0.35$ was used in following experiments (if possible).

5.1.2 Convergence Test

Another measured parameter is the convergence of the ILP solver towards the optimal solution in time. The experiment was done on instances with 13 targets and 4 patrollers. The results are depicted in Figure 5.2.

On the Figure 5.2 the x-axis represents time of simulation run, y-axis represents relative value. The lines above 1 are the optimum relative value. They represent multiple of the optimal solution. Under value 1 are values $1 - rGap$, where $rGap$ is

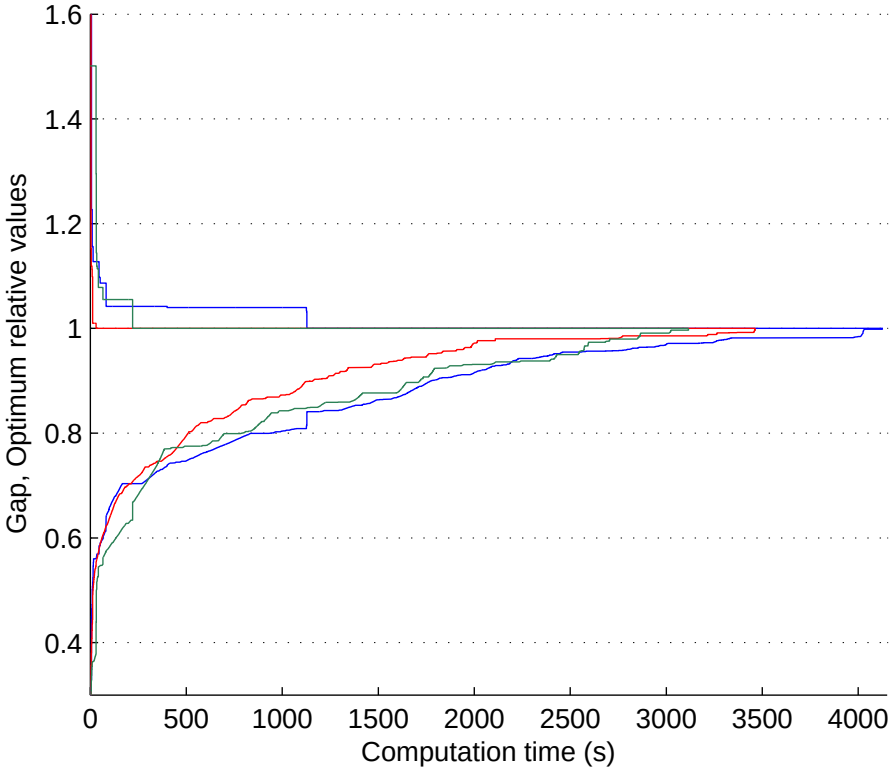

Figure 5.2: Convergence for MPPP.

relative gap used by Cplex solver. These values roughly means that the solver thinks that the current solution can be improved by maximally $1 + rGap$. This value is as bound used to determine when the computation should be stopped.

From the figure it is clear that the solver is able to find solution in early stage of the run but then it still searches through remaining state space. It is clear that the gap at this stage doesn't represent the actual state.

5.1.3 Dependence on Number of Targets

In this section, the performance dependence on the number of targets is measured. For this experiments the number of patrollers was fixed to 5 and the number of targets was changed in from 5 to 12. Each case was tested 50 times, resulting number was obtained by application of mean function. The results are depicted in Figure 5.3.

Note that in Figure 5.3 y-axis is in logarithmic scale. Therefore, the shape of the plot which is approximately linear means that the time complexity grows exponentially.

The dependence on number of agents is not explicitly measured since it describes the same values as deployment to saturation ratio from section 5.1.1.

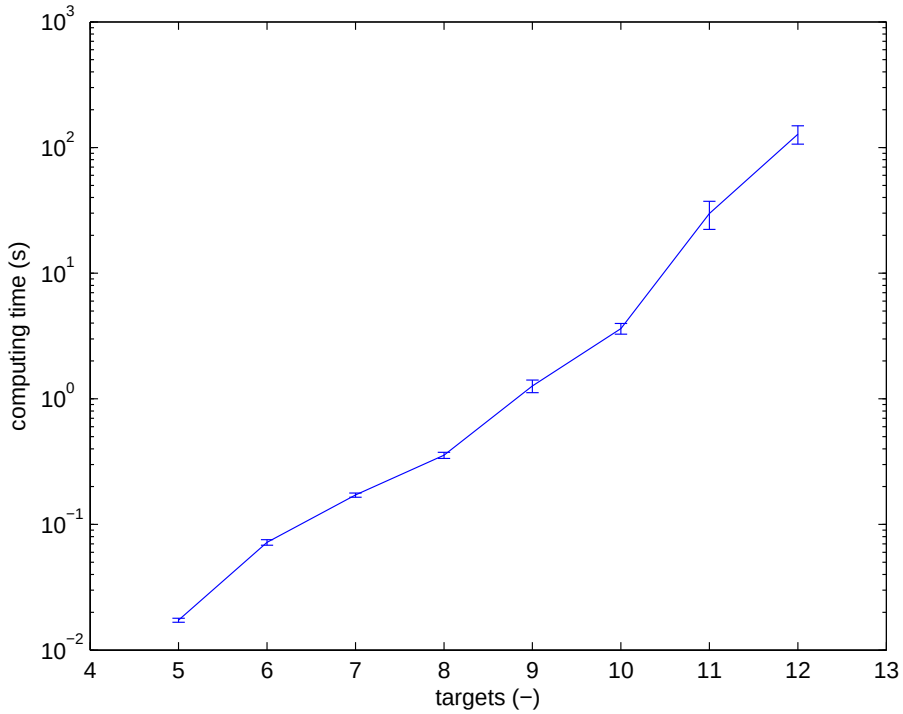

Figure 5.3: Performance dependence on number of targets.

5.2 Performance in Simulation

Although the simulation was created to model the real scenario, some adjustments against realism of the model were made in order to make the results more clear.

The most significant change is in the activity of robber agent. In real life conditions robber attempts one attack in months. However in simulation the robber agent is very active and it attempted to attack on daily basis.

To allow run of multiple simulations, the commercial vessel traffic was set such that the average number of commercial ships in harbour was 10. Therefore the average number of targets was also 10 which lead to ILP runs with average length under 1 second. In simulation were patrolling one UAV and two USVs. Each simulation was run for simulated 90 days.

To suppress influence of non-steady statem. the simulation data record was cut-off first few days

Worst idleness was measured with sampling frequency $t = 10\,\mathrm{s}$. The example of worst idleness result is depicted in Figure 5.4.

In the Figure 5.4 one can clearly observe situation when the layout of targets changed significantly and the worst idleness pattern shifted up or down.

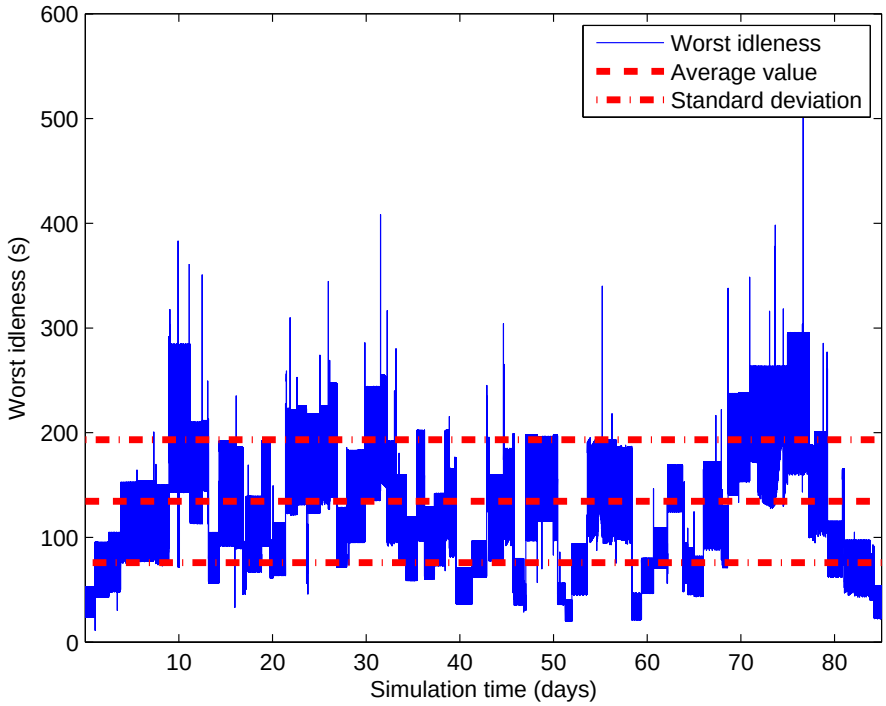

Figure 5.4: Worst idleness example.

Another measured value was dependence of successful attacks and patroller intervention on time τ which takes robber to rob a target. The intervention is a situation when patroller the disarms attacker. The time values were chosen with respect to the previous experiment. The results are depicted in Figure 5.5.

From Figure 5.5, it's clear that the number of successful attacks is a linear function of the attack time τ. The increasing trend of interventions versus decreasing trend of interventions is not a surprise – attackers who weren't caught were successful in their attack.

5.3 Discussion

This section examines the measured results and discusses the possible outcomes from these measurements.

In section 5.1.1 performance on the deployment to saturation ration was measured. It shows that the hardest case is with $(d : s) \approx 0.35$. This is interesting result in comparison with set of security games described by Jain et al. (2012), where the measured ratio was $(d : s) \approx 0.5$. Since this case is in some way similar to games described by Jain et al. (2012), future study of this phenomenon might give interesting insight into this phenomenon.

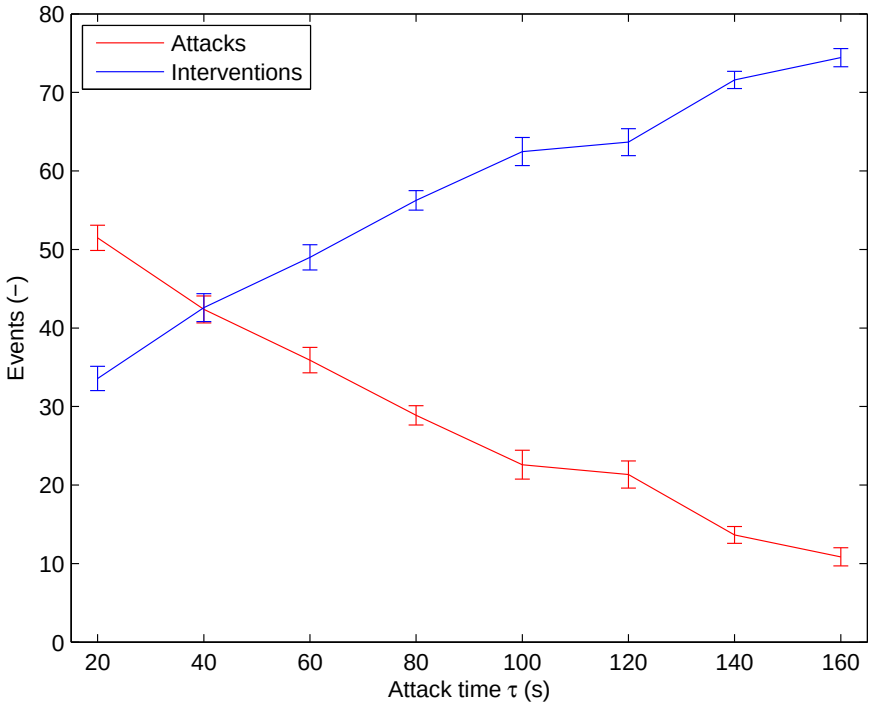

Figure 5.5: Dependence of attacks and intervention on attack time τ.

The convergence pattern presented in section 5.1.2 shows that even though the solution is find by algorithm quite quickly, the algorithm still "belives" it is possible to improve this solution for quite a long time. This suggests that the ILP solver could benefit for modification of constrains in ILP formulation.

One possibility to improve this performance might be to over-constrain the program. By adding redundant constrains it might be possible to input into solver more domain knowledge which could be used in computation.

Another possibility might be to modify program so it would calculate with undirected graph representation. In current solution, the ILP formulation contains representation of directed graph even though it is constructed from undirected one by replacing each undirected edge by two directed ones (one in each direction). This implies that each MPPP has at least two optimal solutions – one in each way, which are basically identical. By changing the graph representation, the state space would be reduced by half.

In section 5.1.3, the performance dependence on the number of targets was examined. It was shown that the complexity grows exponentially in the number of targets. To suppress this issue, it might be possible to leverage the fact that, lot of targets in real situations tend to be close together, so that the patroller could cover more of them at a time. The set of targets on the input could be preprocessed by clustering techniques in order to find clusters where the distance between each

member would be lower than patrollers sensor range. These clusters would be the new set of targets for the patroller. This would lead to problems with less targets and thus with lower time consumption.

Another improvement based on real life scenario could be in exploiting the fact that when a new target is introduced into problem, the solution from the previous problem is already known. This new target could be added to current solution by a greedy method and this upgraded solution (which is feasible) could be used as warm start[1] for ILP solver. It is reasonable to expect that this solution might be close to optimum or event might be optimal.

In section5.2, the experiments in simulation were performed. The dependence on attack time τ was measured. To set reasonable values of attack time for this experiment the worst idleness was measured first. It is shown that the number of attacks and interventions is a linear function of attack time τ.

[1]Warm start is an initial solution that the solver uses to start its search through the state space and to set bounds.

Conclusion

In this work algorithms for area surveillance and target patrolling were designed. These algorithms require patroller to visit targets in order to minimize maximal unvisited time of each target.

The problem of area surveillance is defined in this thesis as optimization problem where the space was decomposed into the cells using hexagonal tilling which leads to a triangular grid. The criterion of the optimization problem is expressed as minimization of maximal age of each cell. It was proved that this problem for grid-like graph is equal to the hamiltonian path problem. The area surveillance uses in this case a polynomial algorithm for finding hamiltonian path in triangular grid graph.

Algorithm solving the problem of attack detection with multiple static targets was formulated using the integer linear programming. To approach this issue, the novel multiple patroller path problem has been introduced. The MPPP is a modification of the multiple travelling salesman problem with constrains regarding length of individual paths and a relaxation of constrain regarding fixed depot for each salesman. Additionally, the MPPP considers all possible combinations of sharing individual patrolling paths between patrollers.

The algorithms were modified to work with set of heterogeneous agents, where agents differs in their movement capability (aerial, surface) and in their speed capability.

To evaluate these algorithms, a discrete event multi-agent simulation of the Mombasa harbour was created. The simulation contains necessary geospatial data in form of coastal lines, ports and anchorage areas. Agents deployed in the simulation represent commercial vessels, surveillance units and robbers. The surveillance units are of two types: UAV and USV agents which are controlled by surveillance algorithms.

Experiments show that MPPP solved by ILP are hardest to solve with deployment-to-saturation ratio $(d : s) \approx 0.35$. The time complexity for MPPP is exponential on the number of patrollers. From tests in the simulation it is apparent that the number of successful attacks is linear on the attack time.

Bibliography

aerosurveillance.com. Critical Infrastructure Surveillance. [online], 05 2013a. URL `http://www.aerosurveillance.com/node/11`.

aerosurveillance.com. Unmanned Aircraft Solutions. [online], 05 2013b. URL `http://www.aerosurveillance.com/node/16`.

AeroVironment. Raven RQ-11B datasheet. Technical report, AeroVironment, 2013.

G. E. Andrews. *The theory of partitions*, volume 2. Cambridge University Press, 1998.

Arcturus. UAV T-20 Spec Sheet. Technical report, Arcturus, 2013.

E. M. Arkin, S. P. Fekete, K. Islam, H. Meijer, J. S. Mitchell, Y. Núñez-Rodríguez, V. Polishchuk, D. Rappaport, and H. Xiao. Not being (super) thin or solid is hard: A study of grid Hamiltonicity. *Computational Geometry*, 42(6):582–605, 2009.

G. Atomics. MQ- Reaper/Predator B, 05 2013.

T. Bektas. The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, 34(3):209–219, 2006.

B. Bošanský, V. Lisý, M. Jakob, and M. Pechoucek. Computing time-dependent policies for patrolling games with mobile targets. In *Proceedings of AAMAS*, 2011.

A. H. Buss and P. J. Sánchez. Simple movement and detection in discrete event simulation. In *Simulation Conference, 2005 Proceedings of the Winter*, page 9–pp. IEEE, 2005.

D. Chavey. Tilings by regular polygons—ii: A catalog of tilings. *Computers & Mathematics with Applications*, 17(1):147–165, 1989.

Y. Chevaleyre. Theoretical analysis of the multi-agent patrolling problem. In *Intelligent Agent Technology, 2004.(IAT 2004). Proceedings. IEEE/WIC/ACM International Conference on*, page 302–308. IEEE, 2004.

J. Faigl and L. Přeučil. Self-organizing map for the multi-goal path planning with polygonal goals. In *Artificial Neural Networks and Machine Learning–ICANN 2011*, page 85–92. Springer, 2011.

M. R. Garey and D. S. Johnson. *Computers and intractability*, volume 174. freeman New York, 1979.

A. J. Healey, D. Horner, S. Kragelund, B. Wring, and A. Monarrez. Collaborative unmanned systems for maritime and port security operations. Technical report, DTIC Document, 2007.

M. Jain, K. Leyton-Brown, and M. Tambe. The deployment-to-saturation ratio in security games. *Target*, 1(5):5, 2012.

M. Jakob, E. Semsch, D. Pavlíček, and M. Pěchouček. Occlusion-aware multi-uav surveillance of multiple urban areas. In *6th Workshop on Agents in Traffic and Transportation (ATT 2010)*, page 59–66, 2010.

M. Jakob, O. Vanek, and M. Pechoucek. Using agents to improve international maritime transport security. *Intelligent Systems, IEEE*, 26(1):90–96, 2011.

M. Jakob, Vanek, O. Hrstka, and M. Pechoucek. Agents vs. Pirates: Multi-Agent Simulation and Optimization to Fight Maritime Piracy. 2012.

F. Klügl. *Agent-based Simulation Engineering*. PhD thesis, July 2009.

A. Komenda, J. Vokřínek, M. Čáp, and M. Pěchouček. Developing Multiagent Algorithms for Tactical Missions Using Simulation. *IEEE Intelligent Systems*, 28(1)(1): 42–49, Jan.-Feb. 2013. URL `http://doi.ieeecomputersociety.org/10.1109/MIS.2012.90`.

A. Machado, G. Ramalho, J.-D. Zucker, and A. Drogoul. Multi-agent patrolling: An empirical analysis of alternative architectures. In *Multi-Agent-Based Simulation II*, page 155–170. Springer, 2003.

C. E. Miller, A. W. Tucker, and R. A. Zemlin. Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)*, 7(4):326–329, 1960.

N. Nigam and I. Kroo. Persistent surveillance using multiple unmanned air vehicles. In *Aerospace Conference, 2008 IEEE*, page 1–14. IEEE, 2008.

M. Odersky, L. Spoon, and B. Venners. *Programming in Scala, second edition*. Artima Series. Artima, Incorporated, 2010. ISBN 9780981531649. URL `http://books.google.cz/books?id=ZNo8cAAACAAJ`.

V. Polishchuk, E. M. Arkin, and J. S. Mitchell. Hamiltonian cycles in triangular grids. In *Proceedings of the 18th Canadian Conference on Computational Geometry*, page 63–66, 2006.

L. . M. W. Program Executive Officer. Unmanned Maritime Systems Program Office (PMS406). [online], 05 2013. URL `http://auvac.org/uploads/publication_pdf/Pms406_PROGRAM.pdf`.

M. Selecký, M. Štolba, T. Meiser, M. čáp, A. Komenda, M. Rollo, J. Vokřínek, and M. Pěchouček. Deployment of Multi-agent Algorithms for Tactical Operations on UAV Hardware (Demonstration). 2013.

D. Sislak, P. Volf, M. Jakob, and M. Pechoucek. Distributed Platform for Large-Scale Agent-Based Simulations. page 16–32, 2009.

D. Sislak, P. Volf, D. Pavlicek, and M. Pechoucek. AGENTFLY: Multi-Agent Simulation of Air-Traffic Management. 2012.

M. Tambe. *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned*. Cambridge University Press, 2011. ISBN 978-1107096424.

C. Umans and W. Lenhart. Hamiltonian cycles in solid grid graphs. In *Foundations of Computer Science, 1997. Proceedings., 38th Annual Symposium on*, page 496–505. IEEE, 1997.

CD content

Attached CD contains source codes of the algorithms and diploma thesis in PDF format. The structure of the CD is described in the following table.

Directory	Description
`source`	source codes of the program
`thesis.pdf`	diploma thesis

Table 1: Directory structure of the CD