Czech Technical University in Prague

Faculty of Electrical Engineering

Department of Computer Graphics and Interaction

Diploma Thesis

Bc. Pavel Salamon

Software Implementation of the Battlespace on Demand Concept

Study programme: **Open Informatics**

Specialisation: **Software Engineering**

Thesis supervisor: **Ing. Ondřej Vaněk, Ph.D.**

Prague, 2014

Affidavit

I declare that I wrote this thesis independently and on my own. No sources or aids have been used in the preparation of this thesis other than those mentioned in the attached list.

Prague, May 9, 2014 ....

Acknowledgement

I would like to take this opportunity to thank Ing. Ondřej Vaněk, Ph.D. for supervising my work. Also I would like to thank my family for their support.

Pavel Salamon

Abstract

The thesis deals with the problem of illegal drug interdiction in the Caribbean Sea and East Pacific. The U.S. naval forces have employed the Battlespace on Demand (BonD) concept to aid them in warfighting effectiveness. The BonD concept introduces tier-based architecture which allows transformation of data in lower tiers into informed decisions in higher tier. The concept is flexible and can be employed in any domain where decision-making takes place.

In this work we present a way to use the BonD concept in the maritime drug interdiction domain. We design and implement software architecture reflecting the BonD concept and use it to make informed decisions based on data collected from the domain. We introduce algorithms optimizing asset allocation and evaluate them on the scenarios extracted from the domain.

Abstrakt

Práce se zabývá problémem zadržování ilegálních drog v Karibském moři a Tichém oceánu. Námořní síly Spojených států amerických zapojily do svých rozhodovacích procesů koncept Battlespace on Demand (BonD). BonD koncept se skládá z vrstvené architektury, která umožňuje přetváření dat z nižších vrstev do informovaných rozhodnutí ve vyšších vrstvách. Koncept je flexibilní a lze nasadit v jakémkoli prostředí, kde je nutné rozhodování.

V této práci používáme koncept BonD v doméně zadržování drog na moři. Navrhujeme a implementujeme softwarovou architekturu reflektující koncept BonD a používáme ji k rozhodování na základě shromážděných dat z dané domény. Představujeme algoritmy pro optimální rozmisťování zdrojů a jejich výstupy hodnotíme na scénářích získaných z domény zadržování illegálních drog na moři.

Contents

List of Tables

List of Figures

LIST OF ACRONYMS AND ABBREVIATIONS

BonD	Battlespace on Demand
FNMOC	Fleet Numerical Meteorology and Oceanography Center
GODAE	Global Ocean Data Assimilation Experiment
GRIB	Gridded Information in Binary
HTTP	Hypertext Transfer Protocol
IED	Improvised Explosion Device
ILP	Integer Linear Programming
Java EE	Java Platform, Enterprise Edition
JAX-RS	Java API for RESTful Web Services
JIATF-S	Joint Interagency Task Force South
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
KML	Keyhole Markup Language
METOC	Meteorological and Oceanographic
MILP	Mixed Integer Linear Programming
REST	Representational State Transfer
RTM	Risk Terrain Modeling

Chapter 1

Introduction

Drug smuggling across the southern border of the United States has been a problem for many years. According to the 2013 International Narcotics Control Strategy Report[3] the majority of drugs smuggled across the southern border is produced in Peru and Colombia and shipped through Ecuador. Drug traffickers transport their shipments using fishing boats, submarines, "go-fast" boats, aircraft and container ships.

Various countermeasures have been put into effect in order to discourage the production, distribution, and consumption of illegal drugs. Joint Interagency Task Force South[4] conducts international detection and monitoring operations and facilitates the interdiction of illicit trafficking in Central and South America.

In the recent years, the U.S. naval forces have employed Battlespace on Demand (BonD) concept to aid them in decision making processes. Battlespace on Demand introduces tier-based system allowing flow of collected data from lower tiers and their transformation into informed decisions based on current conditions in higher tiers. One of the use cases for Battlespace on Demand is optimal asset allocation over areas the drug smugglers pass through. Battlespace on Demand is being used by the US Naval Oceanography programme to enhance warfighting effectiveness[5].

This thesis is directly motivated by the utilization of BonD for drug interdiction domain. We focus on design and implementation of software architecture that reflects the Battlespace on Demand concept and allows automatic decision making in multiple domains. Decision making process used in this work is optimal asset allocation over time in predefined area. Our goal is to deploy assets in time and place to best cover predicted areas where the drug smuggling vessels sail through based on meteorological data and knowledge about smuggler behaviour extracted from the domain.

Developed optimization processes are evaluated and compared on a set of scenarios extracted from maritime drug interdiction data. We describe the impact of scenario parameters, such as area size and time duration, on quality of solution and time required to find that solution.

1.1 Thesis Goals

1. **Study the concept of Battlespace on Demand (BonD).**

   To be able to design and implement the software architecture of the BonD concept, the concept itself needs to be studied. We need to understand how the concept operates, what are its inputs and outputs and what its use cases are. This is all described in Chapter 2.

2. **Study the problem of drug interdiction in the Caribbean sea and East Pacific.**

   To successfully employ the BonD concept in the drug interdiction domain, we need to research the domain itself. The actors in the domain need to be identified and their behaviour needs to be understood. Sets of available domain data must be described. The description of the drug interdiction domain is available in Chapter 3.

3. **Extract and describe a set of scenarios from the domain using the concept of BonD.**

   To model the use of the BonD concept in the drug interdiction domain, we need a set of accurate representations of the drug interdiction problem instances. Using the research from step 1 and 2 we extract a set of scenarios which will simulate real-world problem instances. Data sets used for extraction of the scenarios are described in section 3.4. The process of scenario creation is available in Chapter 4 and scenarios themselves are described in section 6.2.

4. **Design a software architecture enabling modeling of the tier-based architecture of BonD.**

   The software architecture able to model the BonD concept needs to reflect the tier-based architecture of the BonD. We propose an architecture consisting of four layers corresponding with the tiers of the BonD concept. Formalisation of this architecture is described in Chapter 4.

5. **Implement the system using data from the drug interdiction domain using any JVM compatible language.**

   The implementation of the proposed software architecture from step 4 is described in Chapter 5. We present a system of four independent components communicating with each other using URI-based requests. These four components together form a system capable of turning data into informed decisions using the BonD concept.

6. **Develop an optimization algorithm for optimal asset allocation at the decision layer.**

   The last step in the BonD concept is using the data from lower tiers to decide the best course of action. This is done at the Decision Layer and in this work the best course of action is represented by the optimal asset allocation.

   We propose two optimization algorithms. The first algorithm uses mixed integer linear programming to provide optimal solutions at the cost of lower scalability, while the second algorithm uses dynamic programming approach to provide suboptimal solutions and higher scalability. Formalisation of both solution approaches is available in section 4.4, where the Decision Layer is described.

7. **Integrate the algorithm into the system developed in step 5.**

   Both algorithms are integrated into the software component representing the Decision Layer of the BonD concept. The developed architecture allows an easy integration of multiple different solution algorithms. The integration of the two algorithms is described in section 5.2.5.

8. **Demonstrate algorithm's added value for scenarios from step 3.**

   As the last part of our work we focus on the evaluation of developed algorithms on the scenarios extracted from the drug interdiction domain. In Chapter 6, we present results of the algorithm's performance evaluation and the added values for extracted scenarios.

1.2 Thesis Structure

- **Chapter 2** addresses the first goal of the thesis. It presents the Battlespace on Demand concept and its use cases. It also provides an overview of Linear Programming and Multiple Asset Trajectory Optimization.

- **Chapter 3** addresses the second goal of the thesis. It describes the maritime drug interdiction domain, weather data sources used in this work and knowledge about smuggler behaviour.

- **Chapter 4** addresses the third and fourth goal of the thesis. Design of architecture reflecting layers of Battlespace on Demand is described and algorithm and data structures used in solving the asset allocation problem are presented in this chapter.

- **Chapter 5** addresses goals $5, 6$ and 7. It describes implementation of the designed software architecture as a web application and integration of solution approaches.
- **Chapter 6** addresses the final goal of the thesis. The performances of developed optimization algorithms are presented on scenarios from the drug interdiction domain.

Chapter 2

Related Work

This chapter provides an introduction to the Battlespace on Demand concept and presents an example of its use in various projects.

2.1 Battlespace on Demand

Battlespace on Demand (BonD) is an operational concept designed to enhance the warfighting effectiveness by providing decision superiority. Although the concept was originally designed to aid naval forces in warfighting, it can be applied to any decision making process[5]. The idea of BonD is to provide a way to use observation data and forecasts to make informed decisions.

Battlespace on Demand consists of four tiers. Each tier has specified inputs and outputs. Outputs of lower tiers form inputs of higher tiers which results in a flow of information from bottom (data) to top (decisions).

Tier 0 is the Data Layer, where observational data from various sources are collected and together can provide an accurate information about the current state of conditions in the area. Inputs of this tier can be measurements from buoys, satellites and other sensors. Responsibility of the Data layer is providing these measurements to higher tiers.

On top of the Data Layer lies tier 1, the Environment Layer, where the data is analysed and algorithms are executed to forecast the future state of the environment. We should note that since weather forecasting is a complex problem, a significant amount of computational power is usually employed in this tier.

In tier 2, the Performance Layer, the performance of assets is computed based on how the predicted environment will affect them. Result of these computations is the predicted performance of assets, e.g. detection and movement capabilities of the naval forces. In order to predict the assets' performances, the information about the impact of environment on the performances has to be provided, e.g. impact of weather conditions on detection capabilities must be defined in this tier.

Finally the tier 3, the Decision Layer, is where the decision-making processes take place. Processes in this layer and their outputs are heavily dependent on the nature of task. Various strategic and route planning algorithms can be employed, asset allocation or risk quantification can take place. The goal is to provide the best decisions based on information provided by lower tiers.

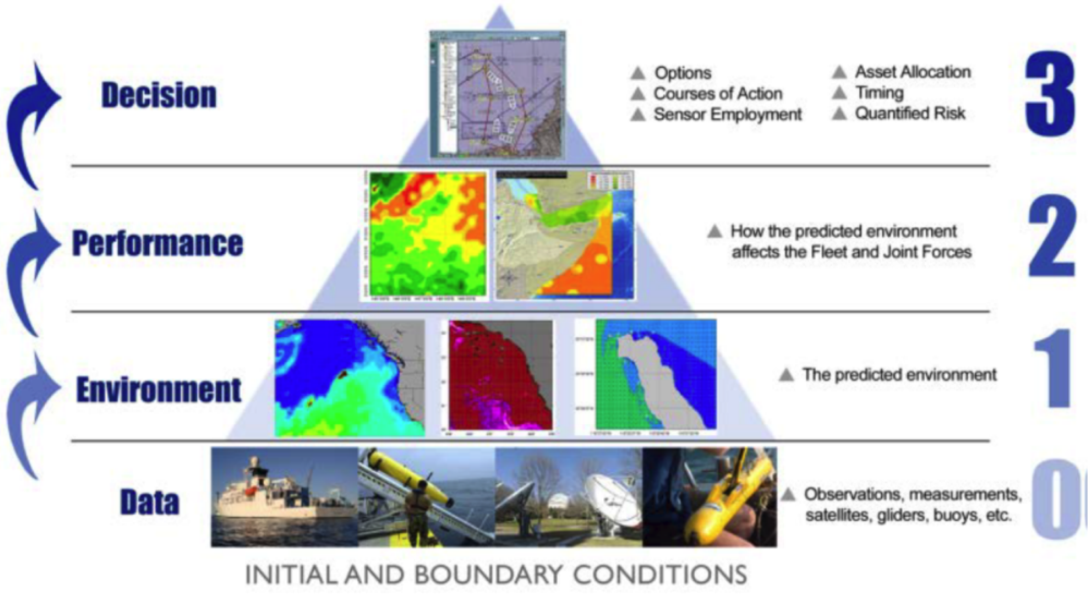

Decision
Options
Courses of Action
Sensor Employment
Asset Allocation
Timing
Quantified Risk
3
Performance
How the predicted environment affects the Fleet and Joint Forces
2
Environment
The predicted environment
1
Data
Observations, measurements, satellites, gliders, buoys, etc.
0
INITIAL AND BOUNDARY CONDITIONS

Figure 2.1: Battlespace on Demand: The 4 Tiers[1]. Each tier contains processes needed in turning data into decisions. Lower tiers are focused on data collection and assimilation while higher tiers use these data to predict future circumstances and make decisions based on the predicted conditions. Final outcome is the best course of action based on all available knowledge.

2.1.1 Use Cases

Chu et al.[6] used BonD in the analysis of the data with respect to the mine and improvised explosion devices (IED) drift scenarios. They utilized real-time climatological data in Data Layer to impact ocean-atmospheric model in the Environment Layer. In the Performance and Decision Layers the simulation of mine drift was executed and impact of tidal forcing and winds on trajectory prediction was evaluated.

Stoughton[7] presented the connection between forecasts and outcomes of Decision layer of the BonD concept in his study focusing on applying forecast uncertainty in planning. Stoughton presented specific decision context for an aircraft carrier ammunition offload and then studied the impact of different wind forecasts from the Environment

Layer on the outcomes in Decision Layer.

United States Navy uses Battlespace on Demand in their strategy by linking environmental data to informed decisions. EIDWS Study Guide[1] provides more detailed description of U.S. Navy's use of individual BonD tiers. Focus of the Data Layer is collection of data from observatory equipment:

> Tier 0 consists of data collected while observing the atmosphere and the ocean using a vast range of in situ sensors and remote sensors, including satellites, altimeters, gliders, buoys, and master clocks. This data is assimilated and fused to provide initial and boundary conditions that accurately describe the current ocean and atmosphere environment, as well as the celestial and temporal reference frames. The output is a collection of raw observation data on the state of the physical environment.

As EIDWS Study Guide[1] states, the Environment Layer contains processes for prediction of the future states of the environment using data from the Data Layer:

> In Tier 1, the Tier 0 data are analyzed, processed, and merged into databases and/or prediction systems or numerical models operated on High Performance Computing (HPC) systems to forecast the future state of the environment. The output is a set of predictions, in space and time, of the expected physical environment for whatever operation is under consideration. The output can also contain a 'confidence factor'.

In the Performance Layer, the predicted environment is used to determine how equipment and forces are going to perform. The ability to operate under given conditions is predicted and provided to higher tier:

> In Tier 2, the predicted environment is used in conjunction with information about the operational environment to predict how forces, sensors, weapons systems, and platforms will perform over time in a given operational situation. This information is analyzed to provide meaning with respect to implications for the operation, such as influences on planning, force structure, targeting, timing, maneuver, tactics, techniques and procedures. The output of this fusion of information about the predicted environment and the friendly and enemy situation is an impact assessment in terms the operator understands, again with a confidence factor if appropriate. Situational awareness is the desired outcome at this level.

In the Decision Layer, the best course of action is decided based on the predicted performance of assets applied to the situation at hand. Final decision is made based on the best understanding of current conditions:

> In Tier 3, the situational awareness gained in Tier 2 is applied to specific situations to quantify risk and opportunity at strategic, operational, and tactical levels. Here, actionable recommendations are made to the decision-maker regarding force allocation and employment that directly enhance safety and warfighting effectiveness. In Tier 3, the performance predictions made in Tier 2 are considered with alternative scenarios to develop optimal solutions, i.e., courses of action (COAs), and to understand probabilities of success and elements of risk. The intent is to make recommendations that take maximum advantage of asymmetric opportunities in the changing physical environment, to provide the most advantage to our forces, and the most disadvantages to the enemy. The output is a decision recommendation with compelling rationale, based on our best understanding of the physical environment. The decision-maker combines knowledge of the present and future situation with their judgement into situational understanding to facilitate superior decision-making.

2.1.1.1 Drug Interdiction Mapping

Let us demonstrate how the tiers of BonD can be mapped in the domain of maritime drug smuggling. We will use similar mapping in the subsequent formalisation and implementation of the drug interdiction problem in this work.

Tier 0 is used for collection of data. In general this data can be any set of observational data describing current state of the real-world conditions, or it can be any domain knowledge or representation of gained experience. The collected data is assimilated to form an accurate representation of current state of the area. This representation is provided to the higher tier. In this work we use two sets of tier 0 data. First set describes weather conditions and second set describes smuggler intelligence. Both of these data sets are described in section 3.4.

In tier 1 the representation of current state is used to predict future states of the environment. Typically the predictions are obtained as outputs of complex forecast algorithms executed on mainframes. In this work we use weather forecasts provided by GODAE[8] which impact predicted smuggler behaviour. Description of these forecasts is available in section 3.4 and more on the smuggler activity predictions can be found in section 4.1.3. The combination of predicted weather conditions and smuggler activity form the output of tier 1.

In tier 2 the performances of available assets are determined. The set of available assets depend on the domain. Generally the assets can represent any units used in achieving required goals. In the maritime domain the assets can represent various equipment such as radars, ships or aircraft. In this work the assets represent independent units of naval forces capable of flying or sailing on open sea. The predicted environment from lower tier impacts the computation of performances. The performance computation used in this work is affected by detection abilities of assets and weather forecast from lower tier. Detailed description of the computational process is available in section 4.3.

Finally in tier 3 the output of the whole system is formed. Depending on the nature of the problem, the output can be either the course of action with proper rationale, or a representation of output of the lower tier in readable format allowing people to make the informed decisions. In medical and engineering industries the output of the tier 3 could be the risk assessment related to current situation. In our work the output is course of action in the form of optimal placement of assets in place and time based on predicted assets' performances in given areas. The process of asset allocation is described in section 4.4.

2.1.1.2 Natural Mapping

BonD can be used in any domain where decision making processes take place. As an example let us demonstrate how the tiers of BonD would be mapped in case of natural event such as underwater earthquake, as presented in article by Rear Admiral Jonathan White[5].

When earthquake strikes, buoys and geological sensors in tier 0 detect the change in sea level as well as location and magnitude of the earthquake. These measurements are provided to real-time tsunami forecasting models in tier 1. Output of the forecasting models are tsunami wave characteristics which can be used to predict the impacts on local infrastructure in tier 2. Government and disaster management authorities (tier 3) can use these predictions as well as other information such as shelter locations to make informed decisions in order to prevent casualties.

2.2 Linear Programming

Linear programming is a method of mathematical optimization for achieving the best outcome in a mathematical model. It allows to define a problem using an objective function and a set of constraints. Decision variables are present in the objective function and constraints and the specification of decision variables forms solution of the problem.

2.2.1 Standard Form

The standard form for the linear programming problem looks like this:

$$
\begin{aligned}
&max && c_1x_1 + c_2x_2 + \cdots + c_nx_n && (2.1)\\
&s.t. && a_{11}x_1 + a_{12}x_2 + \ldots a_{1n}x_n \leq b_1 && (2.2)\\
& && a_{21}x_1 + a_{22}x_2 + \ldots a_{2n}x_n \leq b_2 && (2.3)\\
& && \vdots && (2.4)\\
& && a_{m1}x_1 + a_{m2}x_2 + \ldots a_{mn}x_n \leq b_m && (2.5)\\
& && x_i \geq 0 \qquad \forall i \in \{1, 2, ..., n\} && (2.6)
\end{aligned}
$$

where c, b and a are known input coefficients and x are the decision variables.

Function in (2.1) is the objective function which the linear program is meant to maximize. The inequalities are the constraints which are restricting the feasible region.

As stated before, the specification of the decision variables $(x_1, x_2, \ldots, x_n)$ is the solution of the linear program. If the solution violates one or more constraints, it is infeasible. Otherwise, the solution is feasible. It is possible for a problem to have no feasible solutions. Feasible solution which has the maximum value of the objective function of all the feasible solutions is called optimal solution.

2.2.2 Integer Linear Programming

Linear programming assumes that noninteger values are permitted in the decision variables. If this is not the case in some problem and the problem only allows integer decision variables, then the problem is defined as integer linear programming. If only some of the decision variables are restricted to integers while others are not, the problem is referred to as mixed integer linear programming. Special case of integer linear programming is binary integer programming, where variables can only take values of 0 or 1.

2.2.3 Solving Linear Programming Problems

Widely used algorithm for linear programming is the simplex algorithm[9]. In the case of integer linear programming, solvers often use Branch-and-Bound algorithms and cuts to provide solutions.

2.3 Multiple Asset Trajectory Optimization

Finding optimal trajectories for multiple cooperative agents has been described in [10]. Practical solution has been demonstrated using nonlinear programming.

Cap et al.[11] focused on path finding and collision avoidance. They stated that the state-of-the-art algorithms for cooperative pathfinding typically rely on heuristic forward-search pathfinding techniques where A* is often the algorithm of choice. They proposed a novel algorithm called MA-RRT* for multi-agent motion planning that builds upon a previously proposed sampling-based algorithm called RRT*. They demonstrated the use of the MA-RRT* algorithm on the case where the agents' mobility model is a discrete graph and evaluated its scalability with respect to the number of agents and the size of the environment.

2.4 Risk Terrain Modeling

Risk Terrain Modeling (RTM) is an approach to spatial risk analysis that utilizes a geographic information system to describe digitalized map locations using data extracted from the real-world. Risk terrain maps can describe conditions impacting the operations conducted in depicted areas in the future. One of the use cases of the risk terrain maps are localized crime statistics that provide an easy to understand overview of the city neighbourhoods. Risk assessments for crime and other hazards are especially important for tactical actions, resource allocations and planning problems[12].

Chapter 3

Drug Interdiction Problem

This chapter describes the problem of drug interdiction in the Caribbean Sea and East Pacific, presents knowledge about contemporary drug smuggling operations and describes what problems we will model in this work.

3.1 Domain Description

Despite many years of US government agencies' effort to limit drug smuggling across the southern border of the United States, drug smuggling still continues to be a challenge. In lesser extent the illicit drugs are transshipped to Europe too. Map of the region is depicted in Figure 3.1.

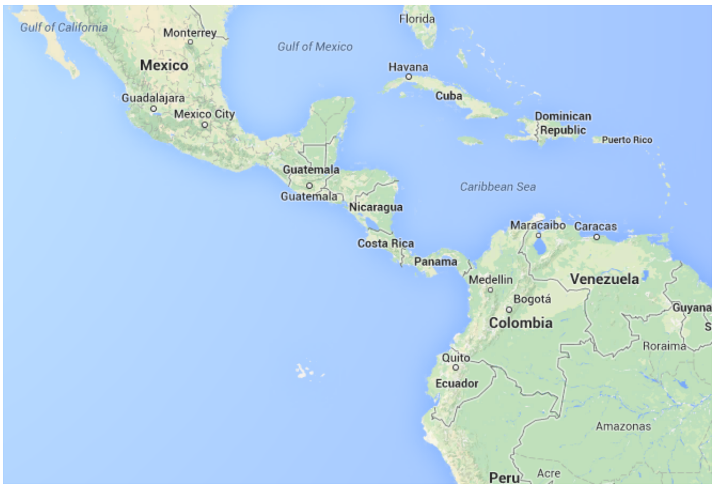


Figure 3.1: Geographical map of the domain region. Illegal drugs are produced mainly in Colombia and Peru. Destination of traffickers shipping their cargo across Eastern Pacific is the southern coast of Mexico. The transport methods include boats, aircraft and container ships.

Transit regions for the drugs produced in South America include Caribbean Sea and East Pacific Ocean. Islands in the Caribbean Sea serve as transit points in the traffic chain. Colombia remains number one cocaine supplier and Ecuador continues to be major transit point for cocaine, heroin and precursor chemicals. 2013 International Narcotics Control Strategy Report[3] states that:

> According to U.S. government estimates, up to 110 metric tons (MT) of cocaine transit Ecuador annually. This includes cocaine from Peru and Colombia, heroin from Colombia follows a similar pattern. Drug traffickers and movers of contraband transport shipments in various ways, including through small fishing boats, self-propelled semi-submersible and fully-submersible submarines, "go-fast" boats, non-commercial aircraft, human couriers, mail, and container ships.
>
> ...
>
> Drug traffickers continued to use bulk cargo and shipping containers to smuggle drugs out of Ecuador, and did so at an increased rate. Drug traffickers often conceal drugs in a variety of licit cargo.
>
> ...
>
> Additionally, traffickers continued to smuggle petroleum ether (also known as white gas), gasoline, and other precursor chemicals in large quantities from Ecuador to Colombia and Peru for cocaine processing.

The seven independent countries of Antigua and Barbuda, Barbados, Dominica, Grenada, St. Kitts and Nevis, St. Lucia, and St. Vincent and the Grenadines form an island chain used by traffickers to ship illicit drugs through the region. 2013 International Narcotics Control Strategy Report[3] states that:

> The region hosts abundant transshipment points for illicit narcotics primarily from Colombia and Venezuela destined for North American, European and domestic Caribbean markets. Traffickers are increasingly using yachts for drug transit, though "go-fast" boats, fishing trawlers, and freighters continue to serve as transit vessels.
>
> ...
>
> Drug traffickers, primarily from South America, use the region as a transit point to temporarily store drugs in the region's many uninhabited islands. Traffickers then move the cocaine up the island chain by "go-fast" or cargo vessels.
>
> ...

In 2012, the total volume of drugs seized in the Eastern Caribbean was approximately 724.75 kg of cocaine; 19.19 metric tons of marijuana; 1,526 cannabis cigarettes; 3,526,305 cannabis plants; 25,264 cannabis seedlings; and isolated seizures of “crack” cocaine.

3.2 Smuggler Behaviour

In East Pacific Ocean, the smugglers sail out mainly from the coast of Ecuador and Colombia, their destination is mainly the south coast of Mexico. They use cargo vessels and fishing trawlers. In Caribbean Sea the situation is similar but more complicated since islands can serve as transit points where smugglers load and unload illicit drugs which are then transported further using go-fast boats, cargo vessels or planes.

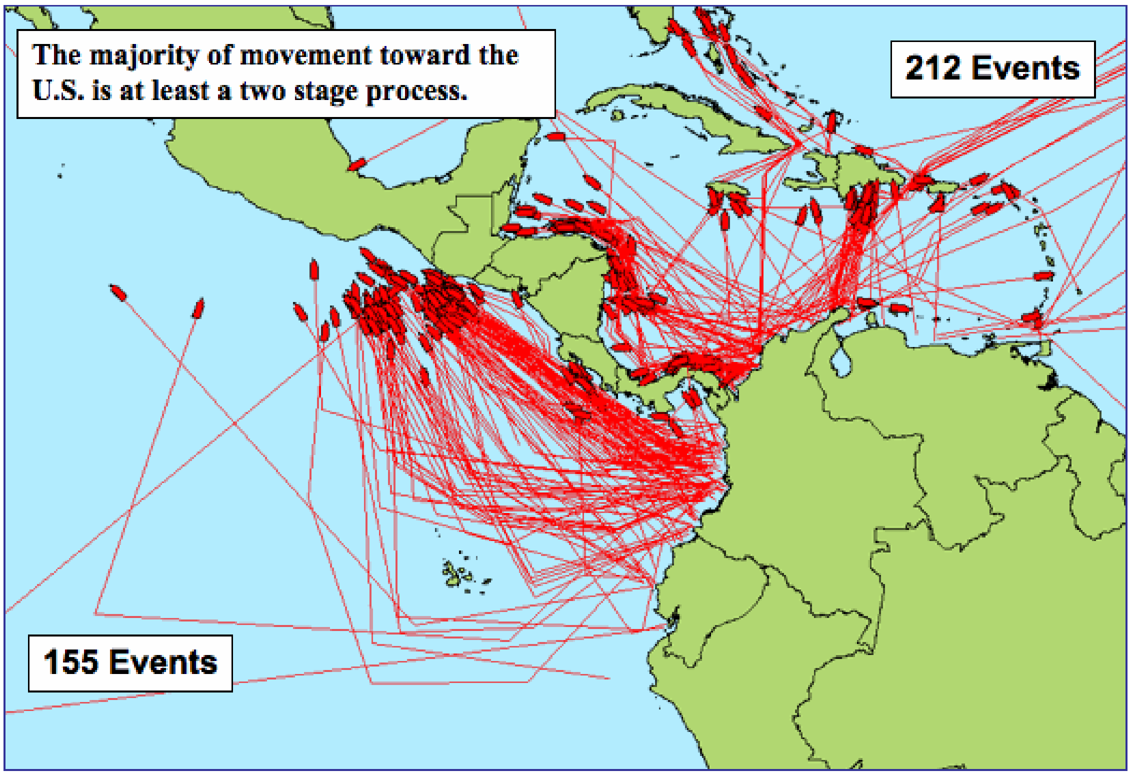

The majority of movement toward the U.S. is at least a two stage process.
212 Events
155 Events

Figure 3.2: Suspected maritime drug trafficking routes in 2007[2] according to the data collected by the Joint Interagency Task Force South. Data shows that the routes originate on the coasts of Colombia, Ecuador and Venezuela. In the Eastern Pacific roughly half of the smuggler vessels take direct route and the other half heads towards open sea and then turns towards the destination.

Figure 3.2 shows data collected by Joint Interagency Task Force South, it depicts regional suspected maritime drug trafficking routes in 2007. Map suggests that majority of suspected traffic originated from the coasts of Colombia.

3.3 U.S. Asset Allocation

In 1989, Joint Interagency Task Force South (JIATF-S) was appointed by the U.S. Department of Defense to conduct counter illicit trafficking operations. JIATF-S conducts interagency and international detection and monitoring operations as well as air and maritime drug interdiction operations in the Caribbean Sea, Gulf of Mexico, and the eastern Pacific. JIATF-S is subordinate command to United States Southern Command which conducts operations in support of the long-term campaign War on Drugs.

3.4 Data

For modeling of maritime drug interdiction we will use two datasets: smuggler intelligence and weather data. Smuggler intelligence dataset contains information about time and location of smuggler vessels. Since detailed information of this kind is not available to the public, we will generate this dataset by ourselves using unclassified data depicted in Figure 3.2. More on this dataset can be found in section 4.1.3.

The second dataset contains METOC (meteorological and oceanographic) data. Various oceanographic institutions provide unclassified data about current and predicted sea conditions. One of them is the Fleet Numerical Meteorology and Oceanography Center (FNMOC). Among various METOC datasets, FNMOC provides global and regional ocean wave prediction charts which are available on the FNMOC website[1] in KML format[13] or on the Global Ocean Data Assimilation Experiment (GODAE) website[2] in GRIB format[14]. In our model the significant wave height prediction charts are going to be sufficient for determining the sea conditions. Significant wave height is defined as mean wave height of the highest third of the waves. Datasets containing significant wave heights are part of the ocean wave prediction charts[3]. Example of significant wave height and direction data in visual form is in Figure 3.3.

[1]https://www.fnmoc.navy.mil/wxmap_cgi/index.html
[2]http://www.usgodae.org/ftp/outgoing/fnmoc/
[3]http://www.usgodae.org/ftp/outgoing/fnmoc/models/ww3/

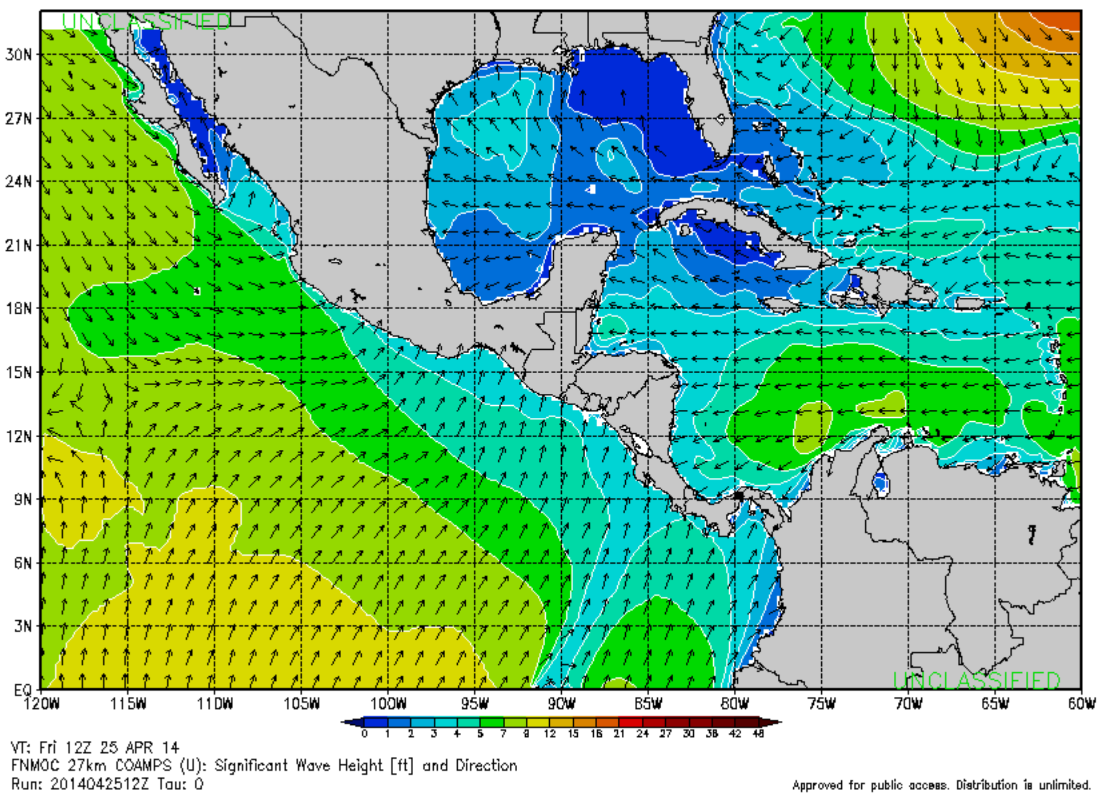


Figure 3.3: Example of significant wave height and direction chart from FNMOC. Data for Central America region measured on April 25, 2014

3.5 Modeling Challenge

In order to reflect the problem of maritime drug interdiction, our task is to model an asset allocation problem in naval environment. The modeled environment must reflect the collected knowledge about smuggler behaviour. By combining the smuggler movement knowledge and environmental weather data we will create maps of potential reward values the assets will gain when covering areas. Optimization algorithms will then be run to determine optimal asset allocation.

Chapter 4

Formal Model

In this chapter we describe a model of Battlespace on Demand layers, the problem described in 3.5 and propose approaches to solve multiple asset trajectory optimization.

4.1 Data Layer

At the bottom of the Battlespace on Demand lies the Data Layer consisting of providers of meteorological and oceanographic (METOC) and smuggler intelligence data.

Modeling the Data Layer of BonD requires a way of obtaining data and storing them in data structures. In section 3.4 we described the data we will use in modeling drug interdiction. Now we need to design an algorithm to generate intelligence data and develop data structures to hold the METOC and intelligence data.

4.1.1 Grid Representation

Standard METOC data representation in oceanography is Gridded Information in Binary (GRIB) format[14]. GRIB allows for description of real-world areas by discretizing them into rectangular 'cells'. METOC data we use in our model are formatted in a latitude-longitude grid version of GRIB. It is a two-dimensional array where each element represents area of predefined size and describes that area with a single floating point value. We can therefore store METOC data in our model using a two-dimensional array with metadata properties describing size of area represented by one element and coordinates placing the whole described area in real-world.

As described in section 3.4, METOC data used in our model are significant wave heights. To simplify use of these values by algorithms in higher tiers of BonD we transform these values to fit into interval between 0 and 1, where 0 means worst conditions possible and 1 means perfect conditions and visibility. This transformation is realized by the following formula:

$$metoc = max(0, 1 - \frac{wave\ height}{h}) \tag{4.1}$$

where h is the maximum wave height which allows for maritime operations. In our model this value is 48 feet. If there are any waves higher than this value in any area then the METOC conditions in that area are set to 0.

With this transformation we changed the original METOC function which provided significant wave height based on discretized longitude and latitude:

$$(X, Y) \rightarrow \mathbb{R} \tag{4.2}$$

into a function which provides METOC description between 0 and 1.

$$(X, Y) \rightarrow \langle 0, 1 \rangle \tag{4.3}$$

An example of extracted METOC data is depicted in Figure 4.1. The example data is layered over a geographical map of the domain region. The data shows METOC values using a heat map, cells describe METOC condition in the given area.

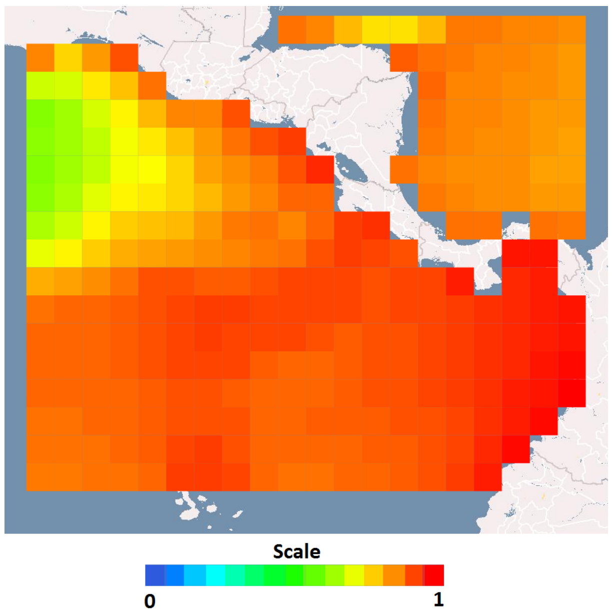


Figure 4.1: Heat map visualisation of the example METOC data. Cells describe the METOC conditions in their respective area by a value between 0 and 1. Value of 1 corresponds with the best possible conditions and value of 0 means the worst possible conditions.

4.1.2 Enhanced Grid Representation

Two-dimensional grid representation is not sufficient for representation of smuggler intelligence data. Since location of smugglers in areas change in time, we need to introduce a third dimension to account for time in our model. By discretizing time into time steps we can create a set of grids where each grid corresponds to a time step and describes its area in that time step. We can now consider this list of two-dimensional grids a three-dimensional data structure. This structure can serve to provide both smuggler intelligence and METOC conditions if given discretized longitude, latitude and time step. If the structure holds smuggler intelligence data, it provides real numbers:

$$(X, Y, T) \to \mathbb{R} \tag{4.4}$$

and if the structure holds METOC data, it provides real numbers between 0 and 1 as described in previous text:

$$(X, Y, T) \to \langle 0, 1 \rangle \tag{4.5}$$

4.1.3 Generating Smuggler Intelligence

To simplify the process of generating smuggler intelligence we will now focus just on the smuggler activity in Eastern Indian Ocean. Data in Figure 3.2 suggest that roughly half of the smuggler vessels choose to sail non-directly, presumably to avoid patrols near coasts. The non-direct trajectories first head west towards open sea, then change direction and continue directly towards their destination. We will model this behaviour in our generated smuggler trajectories. Half of the simulated smuggler vessels will choose a waypoint in open sea and after reaching it they will continue to sail towards the destination. The other half of the vessels will sail directly towards destination. Each simulated vessel will have its corresponding two-dimensional normal distribution which will follow the vessel's location and represent its detectability on open sea. In each simulated time step we will add smuggler vessels' detectabilities to the smuggler intelligence grid corresponding to that time step.

Pseudocode of algorithm for generating one smuggler vessel's trajectory is depicted in Algorihm 4.1.

Algorithm 4.1: Algorithm for generating smuggler trajectories

```
Input   : entryArea, exitArea
Output: vesselTrajectory

begin
    entryPoint ← generateEntryPoint(entryArea);
    exitPoint ← generateExitPoint(entryArea);
    vesselTrajectory.addWaypoint(entryPoint);
    if randomBetween$(0, 1) > 0.5$ then
        seaPoint ← generateSeaPoint();
        vesselTrajectory.addWaypoint(seaPoint);
    end
    vesselTrajectory.addWaypoint(exitPoint);
end
```

An example of the resulting smuggler intelligence grid is depicted in Figure 4.2. The example data shows one time step. Each cell represents chance of encountering a smuggler vessel in the respective area. The values of cells are relative with respect to all time steps.

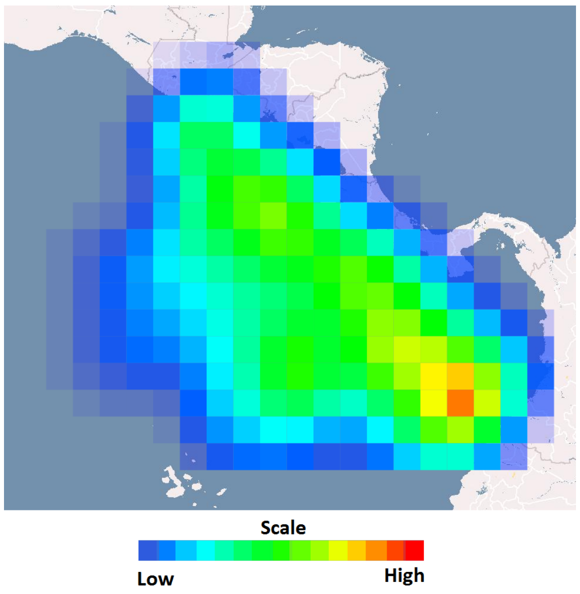


Figure 4.2: Heat map visualisation of one time step of the example smuggler intelligence data. Each cell describes the relative chance of an encounter with a smuggler vessel in its respective area.

4.2 Environment Layer

The Environment Layer receives both METOC and intelligence data and provides the forecast of the future conditions. Since we use forecasts from external sources we don't need to run weather forecast computations in our model. To reflect the impact of sea conditions on the chances of encountering smugglers in an area, we need to combine METOC forecast and smuggler intelligence data. A multiplication of METOC and intelligence data in corresponding areas results in a prediction of the environment. The data structure of the environment is the same three dimensional structure that was used for representation in Data Layer.

4.3 Performance Layer

In the Performance Layer the environment is used to determine performance of assets. A three dimensional structure representing each asset's performance is computed based on the asset's detection capabilities. The list of these data structures gives us a four dimensional structure we call **Performance map**.

To determine what assets' performances are going to be we need to have three pieces of information. First piece is the predicted environment, second is METOC data and third is information about quality of detection equipment of each asset. Environment and METOC predictions are provided by previous layers. The quality of detection equipment is going to be generated in this layer and to simplify the computations it will be described by values between 0 and 1 where higher values mean higher detection capabilities. To reflect the impact of sea conditions on detection equipment we introduce 'sensor performance' which will be used in the computation of assets' performances. To approximate the fact that sensors perform better in favourable weather conditions we use following function:

$$sp(x) = \frac{sin(\pi x - \frac{\pi}{2}) + 1}{2}. \tag{4.6}$$

Graph of the sensor performance function is depicted in Figure 4.3.

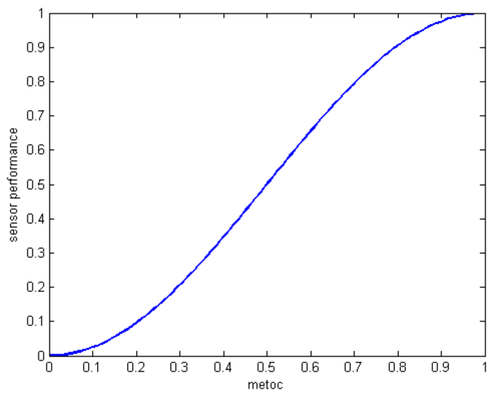


Figure 4.3: Graph of the sensor performance function. In bad METOC conditions the sensor performance is nearing zero and in good METOC conditions the sensors perform well.

Performance of asset i in time step t at location (x, y) is then computed by formula:

$$p^i = d^i \cdot sp({}^tM^{(x,y)}) \cdot {}^tE^{(x,y)} \tag{4.7}$$

where p^i is performance of asset i, d^i is detection equipment of asset i, $sp()$ is sensor performance function, ${}^tM^{(x,y)}$ is metoc value in time step t at location (x, y) and ${}^tE^{(x,y)}$ is environment value in time step t at location (x, y).

4.3.1 Performance Map Element Indexing

Elements in two dimensional matrix can be addressed linearly. By appending columns one after another we can create vector containing all elements of the original matrix. If we know the size of the original matrix we can also find the position of element in the original matrix using its position in the vector. Let us demonstrate on an example.

All positions in this example are indexed from 0. We have a matrix of size 5x3. Values of elements in this matrix are positions of those elements in vector created by linearizing the matrix.

	0	1	2
0	0	5	10
1	1	6	11
2	2	7	12
3	3	8	13
4	4	9	14

Table 4.1: Example of matrix linearization

Transposed linearized vector of this matrix looks like this:

$$(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14) \tag{4.8}$$

Given a position of element in linearized vector we can find the location the element was at in the original matrix. Obtaining row index of element is done by performing modulo operation on position of the element in linearized vector. We use the knowledge that the original matrix had 5 rows:

$$13 \bmod 5 = 3 \text{ (row index of element at position 13 was 3)} \tag{4.9}$$

Using integer division we can find the column index in the original matrix.

$$13/5 = 2 \text{ (column index of element at position 13 was 2)} \tag{4.10}$$

Linearization of matrices will help us in simplifying the formulations of the tasks ahead.

4.4 Decision Layer

At the Decision Layer the decision-making algorithms optimize asset trajectories using the data in Performance map.

4.4.1 Mixed Integer Linear Programming Model

Suppose we have an area map $H^{x \times y}$ which is composed of $h \in \mathbb{N}$ linearized positions $H_0, H_1, ..., H_{h-1}$. Let us define set P for indexing these positions.

$$P = \{0, 1, \dots, h-1\} \tag{4.11}$$

Every position of performance map specifies the reward we get for protecting that position and radius around it. Our task is to maximize the reward using assets provided to us.

In the case of placing a single asset a the optimization task would be:

$$max \sum_{p \in P} a_p \cdot H_p \tag{4.12}$$

$$s.t. \sum_{p \in P} a_p = 1 \tag{4.13}$$

$$a_p \in \mathbb{R} \qquad \forall p \in P \tag{4.14}$$

Value 0 of variable a_p means the asset will not be placed at position p. Value 1 means the asset will be placed at position p. Notice that in the solution vector exactly one variable will be set to 1. Constraint (4.13) states that the asset cannot be at more than one position. It is worth noting that this problem is not defined as integer linear programming program because none of its variables are restricted to be integers. It is defined as linear programming problem and as such it can be solved in polynomial time.

Now suppose we have $n \in \mathbb{N}$ assets we want to model. Let us define set I which will contain all the assets:

$$I = \{0, 1, \dots, n-1\} \tag{4.15}$$

The model of the problem with assets looks like this:

$$max \sum_{p \in P} z_p \tag{4.16}$$

$$s.t. \sum_{p \in P} a_p^i = 1 \qquad \forall i \in I \tag{4.17}$$

$$z_p \leq a_p^i \cdot H_p^i + (1 - a_p^i) \cdot M \qquad \forall i \in I, \forall p \in P \tag{4.18}$$

$$z_p \leq M \cdot \sum_{i=1}^{n} a_p^i \qquad \forall p \in P \tag{4.19}$$

$$z_p \in \mathbb{R} \qquad \forall p \in P \tag{4.20}$$

$$a_p^i \in \{0, 1\} \qquad \forall i \in I, \forall p \in P \tag{4.21}$$

Since we don't want to count values of performance map multiple times in cases where multiple assets are at the same position, we need to introduce variables z_p. Constraints (4.18) and (4.19) specify upper bounds for variables z_p. (4.18) pushes z_p down to reward value of assets at position p. M is number greater than maximum value in performance map H, its purpose is to disregard the inequality in cases where a_p^i is 0. (4.19) pushes z_p down to zero if no asset is at position p.

Since this model does not require assets to keep specified minimal distance from each other, simple greedy approach can provide optimal solution. Let us describe how the algorithm works. First, the positions in a part of performance map corresponding with one arbitrary asset is sorted by reward values from highest to lowest. Then, assets are sorted by their detection abilities d from highest to lowest. After that, sorted assets are assigned one after another to the first $|I|$ positions in the sorted part of the performance map.

Because the sensor performance function is monotonically increasing, it is not possible that asset with lower detection ability would gain more reward value in position already taken by asset with higher detection ability. If we assume that $|P| > |I|$ and sorting can be done in linearithmic time, then the time complexity of greedy algorithm is:

$$greedy = O(|P| \cdot log(|P|) + |I| \cdot log(|I|) + |I|) = O(|P| \cdot log(|P|)) \tag{4.22}$$

Now we need to introduce time to our model. Let us reduce the complexity of time modeling by discretizing continuous time to $q \in \mathbb{N}$ time steps. Set T will contain all time steps we want to model:

$$T = \{0, 1, \ldots, q - 1\} \tag{4.23}$$

Objective function taking time into account looks like this:

$$max \sum_{t \in T} \sum_{p \in P} {}^{t}z_p \tag{4.24}$$

Performances of assets change in time because their sensors are dependant on meteorological conditions. We reflect this fact by modeling rewards in time steps.

Now let's add constraints:

$$max \sum_{t \in T} \sum_{p \in P} {}^{t}z_p \tag{4.25}$$

$$s.t. \sum_{p \in P} {}^{t}a^i_p = 1 \qquad \forall i \in I, \forall t \in T \tag{4.26}$$

$${}^{t}z_p \leq {}^{t}a^i_p \cdot {}^{t}H^i_p + (1 - {}^{t}a^i_p) \cdot M \qquad \forall i \in I, \forall p \in P, \forall t \in T \tag{4.27}$$

$${}^{t}z_p \leq M \cdot \sum_{i=1}^{n} {}^{t}a^i_p \qquad \forall p \in P, \forall t \in T \tag{4.28}$$

$${}^{0}a^i_{s^i} = 1 \qquad \forall i \in I, s^i \in P \tag{4.29}$$

$${}^{q-1}a^i_{s^i} = 1 \qquad \forall i \in I, s^i \in P \tag{4.30}$$

$${}^{t}a^i_p \leq \sum_{p' \in neig(p)} {}^{t+1}a^i_{p'} \qquad \forall i \in I, \forall p \in P, \forall t \in T \tag{4.31}$$

$${}^{t}z_p \in \mathbb{R} \qquad \forall t \in T, \forall p \in P \tag{4.32}$$

$${}^{t}a^i_p \in \{0, 1\} \qquad \forall i \in I, \forall p \in P, \forall t \in T \tag{4.33}$$

Constraint (4.26) is a variation of (4.13). Constraints (4.29) and (4.30) specify the home position of asset a^i which is stored in constant s^i. Constraint (4.31) checks movement of assets. Function $neig(p)$ returns set of allowed positions the assets can move to from position p in one time step.

4.4.2 Alternative Approach

The complexity of MILP is too high for it to be used in real-world scenarios. A scenario reflecting real-world problem would need at least a million binary variables, which results in complexity beyond the point the state of the art MILP solvers can handle.

We propose alternative approach to solve presented problem. We relax the need for optimality which allows us to solve the problem in polynomial time.

4.4.2.1 Iterative Algorithm

Proposed algorithm finds trajectory iteratively one asset at a time. In each iteration it uses dynamic programming method to break down the problem to smaller subproblems. For each asset the algorithm iterates through time steps and stores two pieces of information: the maximum reward value that is possible to gain in each position and the position from which the asset moved to that position. Maximum reward value is computed by selecting the highest reward value from all neighbouring positions in previous time step.

After computing the highest reward values the trajectory is derived by going backwards through time steps and moving to the position in previous time step.

Algorithm provides optimal solution for each isolated asset, however in the general problem the assets are not isolated which results in suboptimal solution. Pseudocode of the algorithm is available in Algorithm 4.2. Depicted code uses sets I, P and T for assets, map positions and time steps respectively.

Algorithm 4.2: Iterative algorithm for suboptimal asset allocation

```
Input  : map[$asset$][$timestep$][$position$]              // performance map
         home[$asset$]                            // home positions of assets
Output: trajectory[$asset$][$timestep$]             // trajectories of assets
         reward[$asset$]                         // final rewards of assets
Variables: best[$asset$][$timestep$][$position$]      // best reachable rewards
           prev[$asset$][$timestep$][$position$]         // previous positions

begin
    fill(best,$-\infty$);                                // set all values to $-\infty$
    for $i \leftarrow 0$ to $|I|$ do
        best[$i$][0][home[$i$]] $\leftarrow$ map[$i$][0][home[$i$]];          // init first timestep

        for $t \leftarrow 1$ to $|T|$ do
            for $p \leftarrow 0$ to $|P|$ do
                foreach location $e$ in neighbours($p, map$) do
                    if best[$i$][$t-1$][$e$] + map[$i$][$t$][$p$] > best[$i$][$t$][$p$] then
                            // we found a way to get here with higher reward
                        best[$i$][$t$][$p$] = best[$i$][$t-1$][$e$] + map[$i$][$t$][$p$];
                        prev[$i$][$t$][$p$] = $e$;            // save the way we came here
                    end
                end
            end
        end

        prevPos $\leftarrow$ home[$i$];                 // set end of trajectory to home
        for $t \leftarrow |T|$ down to 0 do                           // backwards in time
            trajectory[$i$][$t$] $\leftarrow$ prevPos;
            prevPos $\leftarrow$ prev[$i$][$t$][prevPos];
        end
        trajectory[$i$][0] $\leftarrow$ home[$i$];           // set start of trajectory to home
        reward[$i$] $\leftarrow$ best[$i$][$|T|$][home[$i$]];
        clear(map, trajectory[$i$]); // set map values along trajectory to zero
    end
end
```

Chapter 5

Implementation

This chapter provides information about the software implementation of the BonD concept. The software architecture is described and the components of implemented BonD tiers are presented. At the end of the chapter a visualization tool and a front-end application are described.

5.1 Platform

The target for our implementation is Java Platform, Enterprise Edition. This platform was chosen because of the wide range of application servers supporting deployment of the developed web applications. It supports development of web applications that can be composed of independent components. This allows creation of a modular architecture where components can be changed depending on desired functionality of the whole system. Java EE servlet technology extends the request-response communication capabilities of servers that host applications.

Resulting software package containing a web application can be deployed on any application server implementing the Java EE specification. Permissions to write to files are required.

5.2 Layered Architecture

We reflect the layered nature of the BonD concept by implementing four independent components. Each of these components represents one BonD layer.

To allow for simple use of implemented layers, they all respond to URI-based requests using HTTP. The HTTP communication is implemented using Jersey framework[15]. The Jersey framework is an open source framework for developing RESTful web services in Java. Jersey serves as a reference implementation of JAX-RS, the Java API for RESTful

Web Services, which provides support in creating web services. Notable part of the JAX-RS are annotations that help in mapping of Java classes as web resources. We implement one Java class for each BonD tier and map it as a web resource using the JAX-RS annotations. This allows each implemented BonD layer to be available for HTTP requests via GET method.

From outside perspective every layer appears to encompass all the layers below itself. This can be used for simple reuse of the implemented layers. For example if we want to use the environment description provided by the Environment layer in our algorithms, we don't need to handle communication with Data layer because the Environment layer requests information from the Data layer by itself.

Layers communicate with each other using HTTP requests. Parameters of the requests are passed in HTTP headers. Responses contain representation of requested data in JSON format. Google Gson[16] is used to convert between JSON representations and Java objects. Description of all used HTTP headers is in Table 5.1, an overview of use of HTTP headers in layers is available in Table 5.2.

Header	*Units*	*Description*
topLeftX	degrees	Longitude of top left corner of desired area
topLeftY	degrees	Latitude of top left corner of desired area
bottomRightX	degrees	Longitude of bottom right corner of desired area
bottomRightY	degrees	Latitude of bottom right corner of desired area
cellEdgeSizeDeg	degrees	Size of atomic area, specification of granularity
year	integer	Specification of date and time
month	integer	Specification of date and time
day	integer	Specification of date and time
hour	integer	Specification of date and time
timestepsNum	integer	Number of time steps to simulate
timestepDurationHours	integer	Duration of one time step in hours
assetsNum	integer	Number of assets
radius	integer	Size of area covered by one asset
visualize	boolean	Generating KML files for visualisation

Table 5.1: Description of HTTP headers used by components representing the BonD tiers.

	Data	*Environment*	*Performance*	*Decision*
topLeftX	×	×	×	×
topLeftY	×	×	×	×
bottomRightX	×	×	×	×
bottomRightY	×	×	×	×
cellEdgeSizeDeg	×	×	×	×
year	×	×	×	×
month	×	×	×	×
day	×	×	×	×
hour	×	×	×	×
timestepsNum	×	×	×	×
timestepDurationHours	×	×	×	×
assetsNum			×	×
radius				×
visualize	×	×	×	×

Table 5.2: HTTP headers used in different layers.

5.2.1 Interaction Model

The interaction of the user and layers is depicted in Figure 5.1. The communication follows the request-response model. Responses contain JSON representations of layer outputs.

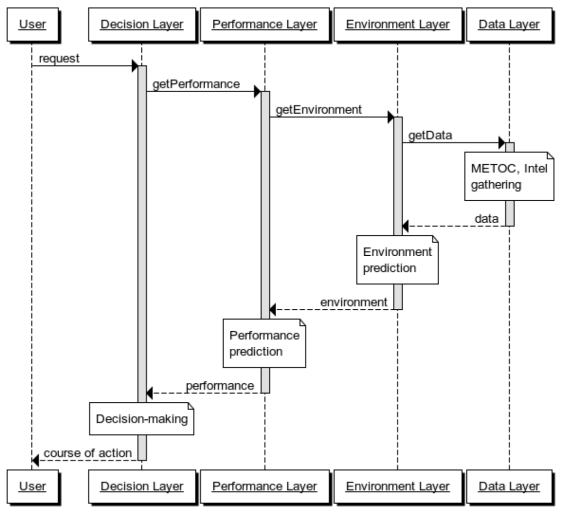


Figure 5.1: Sequence diagram of the user-layers interaction. User requests decision from the highest layer, each layer communicates with its neighbouring layers.

In the following sections each layer and its components are described. Every layer contains processes designated to it by the nature of the BonD concept. For a description of the BonD please refer to section 2.1.

5.2.2 Data Layer

Class diagram of the Data Layer is depicted in Figure 5.2. Components of the Data Layer and their interaction are described in the following text.

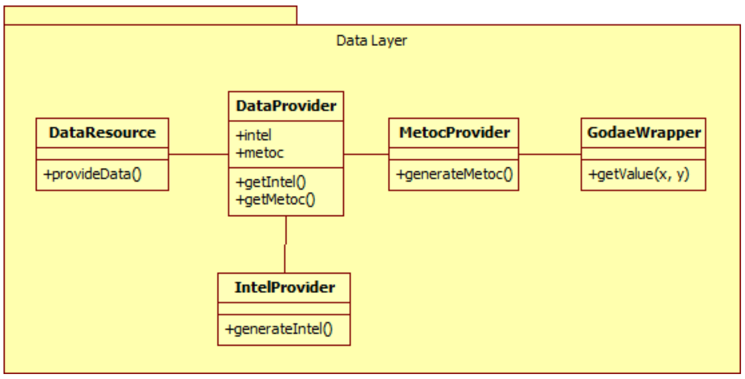


Figure 5.2: Class diagram of the Data Layer.

Responsibility of the Data layer is to provide description of METOC and smuggler intelligence data when requested. To be able to do that the Data layer contains two components called METOC Provider and Intel Provider. Both of these components provide their respective data based on parameters depicted in Table 5.3.

Parameter	*Units*	*Description*
Top left	degrees	Coordinates of top left corner of desired area
Bottom right	degrees	Coordinates of bottom right corner of desired area
Cell edge size	degrees	Size of atomic area, specification of granularity
Date	date	Specification of time and date
Time steps count	integer	Number of time steps
Time step duration	integer	Duration of one time step in hours

Table 5.3: Parameters passed to METOC and Intel Providers

Both METOC and Intel Provider provide their respective data in TimeGrid - an implementation of the data structure described in section 4.1.2.

5.2.2.1 METOC Provider

As described in section 3.4, GODAE project provides METOC data to the public in GRIB format. Responsibility of the METOC Provider is to provide data in more easily readable form when requested. Transformation from significant wave height data from GODAE into our METOC model is realized by implementation of formula (4.1).

Since our goal is to make informed decisions based on forecasts, freshness of the forecasts is an important issue. To tackle this fact we developed a component called Godae Wrapper which downloads the newest forecast files from the GODAE website and provides an interface to access data contained inside them.

Naming convention of the GODAE GRIB files describing significant wave height in Central America is following:

US058GOCN-GR1mdl.0113_0186_**<1>**00F0RL**<2>**_0001_000000-000000sig_wav_ht

where **<1>** is replaced by three digits specifying the forecast time in hours, and **<2>** is replaced by date string. For example the following name string represents 3-hour forecast from 1.1.2014:

US058GOCN-GR1mdl.0113_0186_**003**00F0RL**2014010100**_0001_000000-000000sig_wav_ht

5.2.2.2 Intel Provider

The Intel Provider implements functionality described in section 4.1.3. First, a set of smuggler vessels with their entry and exit points is generated, then the trajectories are simulated. Records of these simulated trajectories form the smuggler intelligence - the output of the Intel Provider.

Entry and exit areas for smuggler vessels we use in implementation are displayed in Figure 5.3.

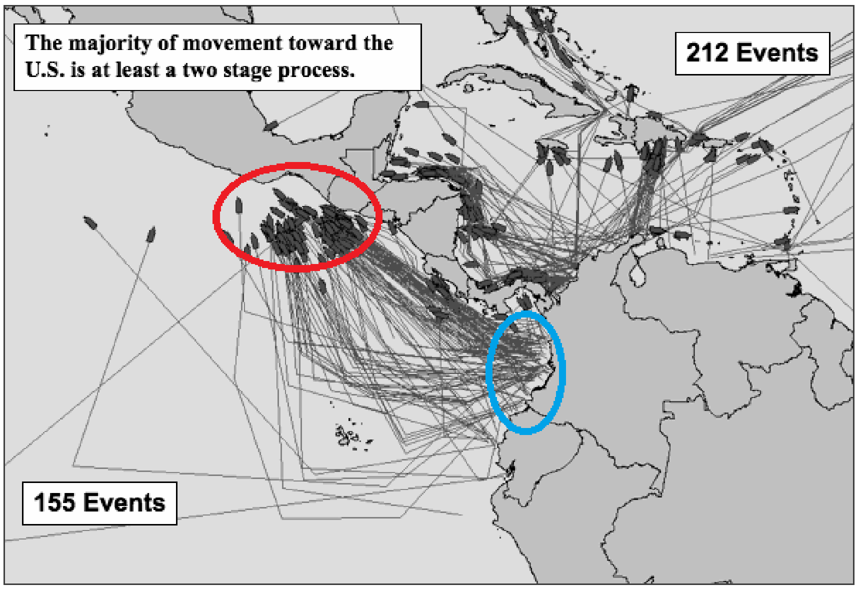

The majority of movement toward the U.S. is at least a two stage process.
212 Events
155 Events

Figure 5.3: Entry (blue) and exit (red) areas for smuggler vessels used in implementation. Background image depicts suspected maritime drug trafficking routes in 2007.

5.2.3 Environment Layer

In the Environment layer the data from the Data layer are combined into a representation of future environment in which the assets will act. This is done by the Environment Provider. As described in section 4.2, the combination of METOC and Intel data is realized by multiplication of values in corresponding areas. The Environment Provider also provides definition of sea and land areas for use in algorithms in the Decision layer.

Class diagram of the Environment Layer is depicted in Figure 5.4.

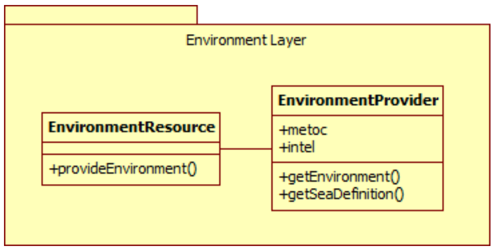


Figure 5.4: Class diagram of the Environment Layer.

5.2.4 Performance Layer

The Performance layer uses Performance Provider component to compute predicted assets' performances. As described in section 4.3, the performances depend on combination of every asset's detection ability and current METOC conditions. General sensor performances are computed from METOC conditions and subsequently used to impact assets' detection abilities in every area. Exact formulas can be found in equations (4.6) and (4.7).

Class diagram of the Performance Layer is depicted in Figure 5.5.

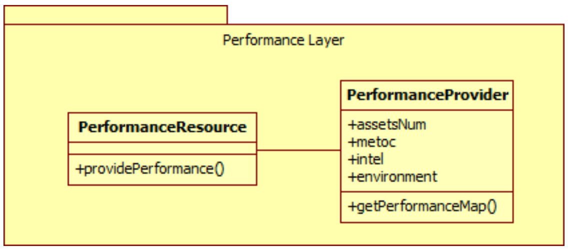


Figure 5.5: Class diagram of the Performance Layer.

Output of the Performance layer is implementation of the Performance map data structure introduced in section 4.3, definitions of assets' home positions are included.

5.2.5 Decision Layer

In the Decision layer the decision-making processes take place. In our case that means optimizing allocation of assets. To allow simple addition of different optimization algorithms in the future we designed abstract class Solver. Inheriting from this class allows an easy integration of different solvers.

Class diagram of the Decision Layer is available in Figure 5.6.

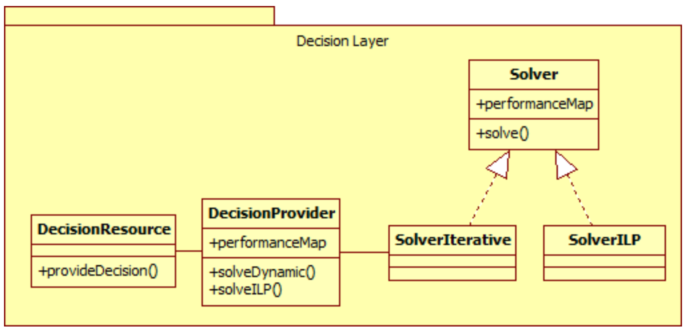


Figure 5.6: Class diagram of the Decision Layer.

In sections 4.4.1 and 4.4.2 we introduced two solution approaches for our decision-making problem. These two approaches are implemented by inheriting from the Solver class. Since the MILP approach is not scalable enough for real-world problem instances the iterative approach is default in our implementation. Both solvers are described in the following two sections.

5.2.5.1 MILP Solver

Variety of MILP solvers is available. Commercial solvers include Gurobi Optimizer[17] and ILOG CPLEX optimizer[18], well known free solver is GNU Linear Programming Kit (GLPK)[19].

After comparing the performance of GLPK and ILOG CPLEX optimizer on our MILP model defined in section 4.4.1, we chose to use the CPLEX optimizer to solve the model. GLPK offered worse performance on all instances of the model. The CPLEX optimizer was able to provide solution in a fraction of the time needed by the GLPK.

Integration of the CPLEX solver required use of its Java API. Since we wanted to allow easy integration of different MILP solvers we did not use the CPLEX Java API directly. Instead we used the Java ILP[20] library which allows use of uniform interface for communication with multiple different solvers. This way our model can be solved by different MILP solvers with minimal changes in the code.

5.2.5.2 Iterative Solver

The Iterative solver is an implementation of algorithm introduced in section 4.4.2. With the *radius* parameter mentioned in Table 5.1, we can control the minimal distance we want the assets to keep from each other. The algorithm also uses the sea and land definition provided by Environment layer to make sure assets only move on navigable waters.

5.3 Visualisation

Every implemented layer can be requested to create visualisation of its output in addition to its responsibilities given by the nature of the BonD concept. Since all the outputs contain data related to geographical areas it makes sense that the data should be visualised over these areas. We format the visualisation of the output data in KML format[13] which allows displaying them in Google Earth[21] or other geographical applications. We used Java API for KML[22] for transformation of our internal data structures into the KML object model described in the specification[13].

In each layer every time step of every dataset corresponds to its KML file. The KML files are available for download from the application server our application is deployed on. Locations for KML files containing data visualisation are in Table 5.4.

Dataset	*Location*
METOC	<host>/bond/kml/metoc/
Intel	<host>/bond/kml/intel/
Environment	<host>/bond/kml/environment/
Performance	<host>/bond/kml/performance/
Decision	<host>/bond/kml/decision/

Table 5.4: Locations of the KML visualisation files at runtime

To access visualisation of a time step from a dataset we need to append the number of time step and KML extension to the location from Table 5.4. For example the path to the visualisation file of first time step of METOC dataset is following:

$$\text{<host>/bond/kml/metoc/0.kml} \tag{5.1}$$

5.4 Front-end Application

In addition to the implementation of the BonD concept and integration of the asset allocation problem, a front-end was developed to allow easy interaction with the application. The front-end runs on client side in the web browser in JavaScript. It displays a dynamic map of the target geographical area and provides controls to enter parameters and run an instance of the drug interdiction problem. Screenshot of the front-end is displayed in Figure 5.7.

Wide range of JavaScript libraries such as Google Maps Javascript API[23] allow embedding dynamic geographical maps in web browsers. The Google Maps library was not suitable for our application because including layers from KML files to the map requires the KML files to be available for download by the Google servers. The URI for download must be passed to the Google servers via the JavaScript API, the servers then download the KML file and after that the file is forwarded to the client of our application.

Instead of using the Google Maps library, we used the OpenLayers library[24] which allows the client to download KML files containing map layers directly. The OpenLayers library keeps the displayed map in the Map object and provides programmatic ways to control the displayed information. We dynamically download KML files containing the information the user request to see. The information from KML files is loaded into the Layer object of the OpenLayers library and provided to the Map object. Depending on the user inputs we make appropriate Layers visible or invisible.

All datasets generated by the BonD layers can be displayed over the geographical map. User can go through the time steps back and forth and switch between the datasets. More detailed description of the work with the front-end is available in Appendix B: User Guide.

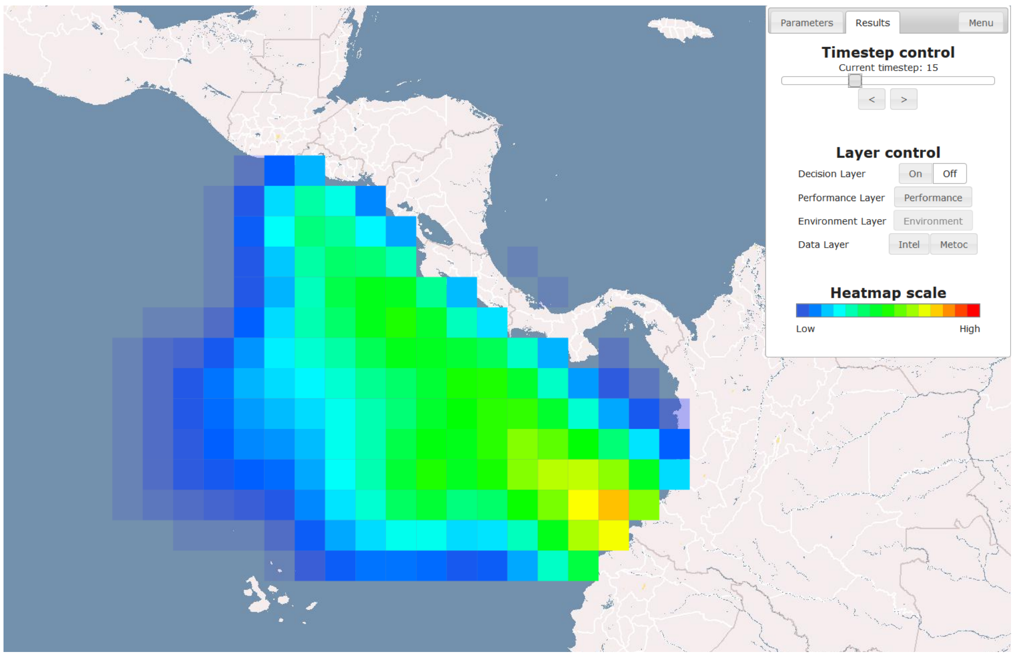


Figure 5.7: Front-end of the application. The geographical map of the region is displayed and menu for controlling time steps and displaying layer outputs is available.

We used jQuery library[25] to develop parts of the front-end. The jQuery JavaScript library simplifies developing JavaScript applications using an easy-to-use API working in wide range of browsers.

The menu and controls are implemented using the jQuery UI library[26] which provides a set of user interface interactions, effects, widgets, and themes built on top of the jQuery JavaScript library. This allowed us to make interaction with the application comfortable and easy to understand.

Chapter 6

Evaluation

This chapter provides an evaluation and comparison of the two solution approaches described in Chapter 4. In this evaluation we measure two algorithm properties. The first measured property is the time the solution was provided in and second is the reward value of the solution.

6.1 Parameters

Both measured properties are dependent on parameters of the problem instance. These include shape and size of the map, number of assets, number of time steps and home positions of the assets. We limit the shape of maps to squares and home positions to the middle of the maps to represent centralized army base. One time step in used scenarios represents time duration of 12 hours.

Please note that because of significant time requirements of the MILP solver the timeout for the solver was set to 300 seconds.

6.2 Scenarios

We prepared two sets of scenarios for evaluation purposes. First set contains scenarios representing problem instances of real-world drug interdiction events. The size of maps is represented by variable called **mapsize** which in this set ranges from 2^2 to 30^2 cells. One map cell represents area of 1×1 degree of latitude/longitude at equator which equals to 111.3 km $\times$ 111.3 km. Number of time steps is represented by variable **timesteps** which ranges from 2 to 40 and number of assets is described by variable **assets** with range between 1 and 5. This set was used for comparison of the two solution approaches and evaluation of the MILP approach.

Second set of scenarios was used to evaluate scalability of the iterative algorithm. It contains scenarios describing more complex instances of the drug interdiction problem. The **mapsize** ranges from 5^2 to 200^2 cells, **timesteps** variable ranges from 5 to 200 and **assets** is again between 1 and 5.

6.3 Execution Time

The first evaluated property of the two solution approaches is the time required to provide a solution. In this section we compare the two developed algorithms and then focus on their scalability.

6.3.1 Execution Time Comparison

In Figure 6.1 we can see graphs plotted on a logarithmic scale depicting the execution time comparison of the two algorithms on a set of scenarios with $1, 3$ and 5 assets, map size 14^2 and varying number of time steps. The data shows that with the increasing number of time steps, the MILP approach quickly reaches the time limit (dotted line) while the Iterative algorithm is not severely affected by the increasing problem complexity in scenarios of this size and provides solution in milliseconds.

This difference in execution times of the two algorithms suggests clear dominance of the Iterative algorithm in the execution time comparison. In the following two sections we focus on each algorithm independently.

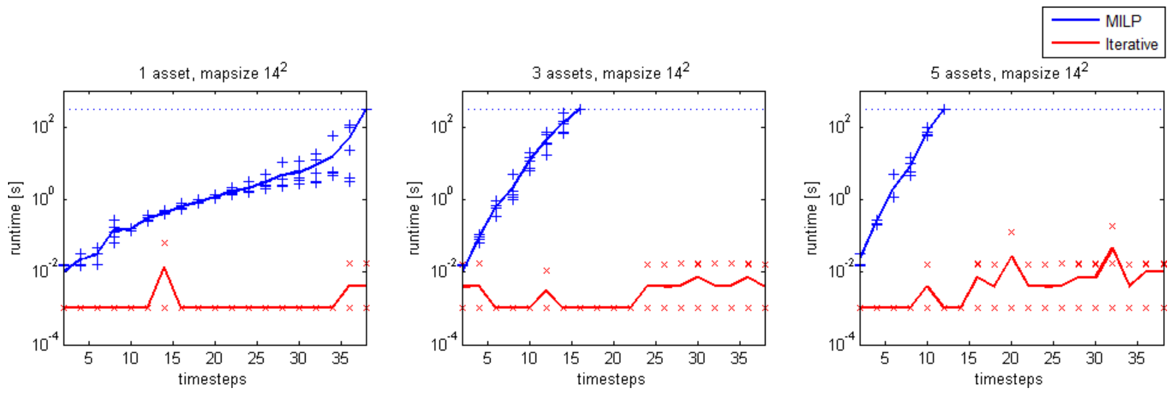


Figure 6.1: Graphs displaying the execution time comparison of the two algorithms on scenarios with 14^2 locations and $1, 3$ and 5 assets. The increase in complexity in scenarios with more time steps is visible and low scalability of the MILP algorithm is shown.

6.3.2 MILP Approach

Figure 6.2 shows execution time performance of the MILP approach on a diverse set of scenarios. The data plotted on a logarithmic scale suggests exponential dependency of the MILP algorithm on the number of time steps. Data also shows that scenarios with more assets need smaller number of time steps to reach the timeout, which shows that adding assets also increases complexity.

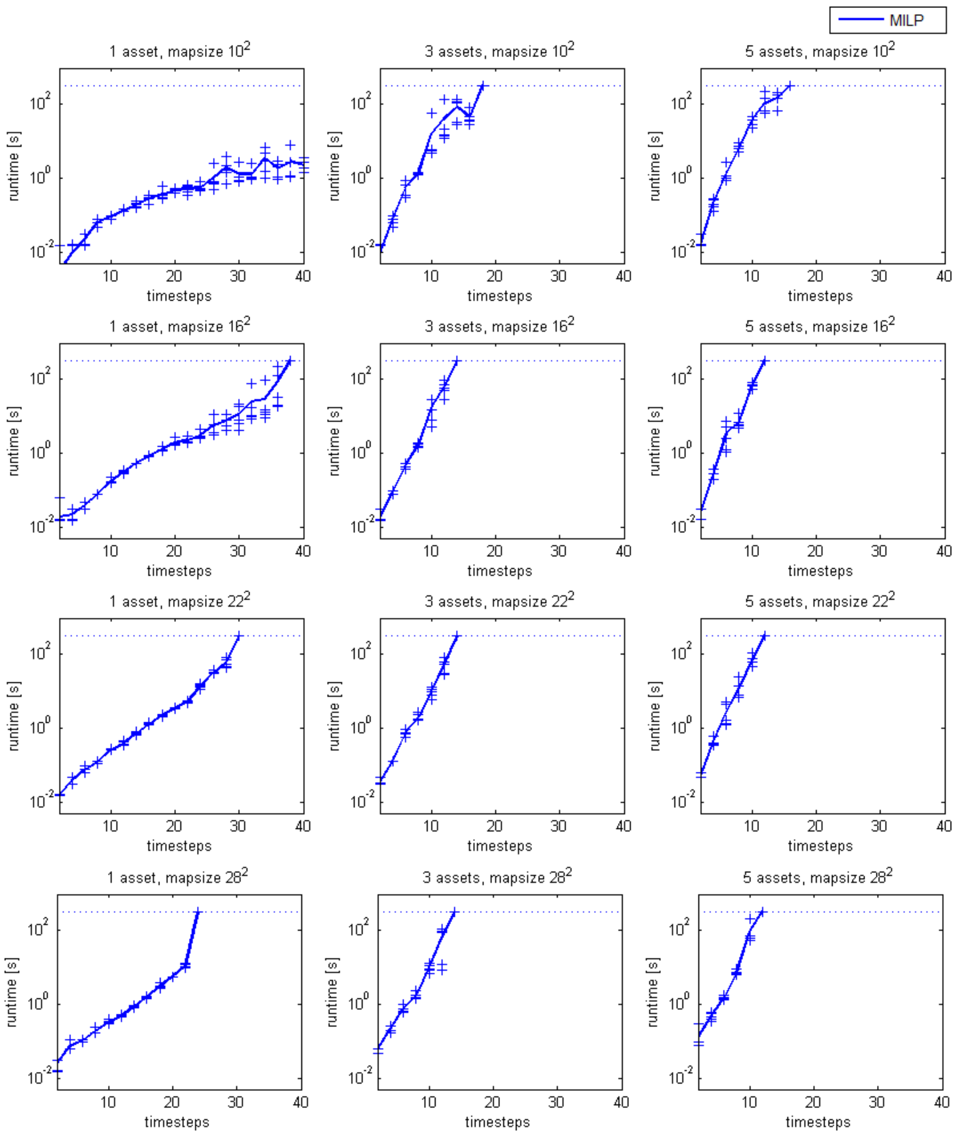


Figure 6.2: Graphs displaying the scalability of the MILP approach using varying map size, assets and time steps. Data plotted on logarithmic scale suggests exponential dependency of the algorithm on the number of time steps. The increase of the number of assets lowers the ability of algorithm to provide solutions before reaching the timeout (dotted line).

6.3.3 Iterative Approach

To better show the computational performance of the Iterative algorithm we recorded its performance on a set of more complex scenarios. Figures 6.3 and 6.4 show the execution time dependency on the number of time steps and the number of locations in map. The recorded data plotted on a linear scale suggests linear dependency of the execution time on both of these parameters.

The measured dependency is not strictly linear due to the garbage collection routines present in the Java platform. In larger scenarios the amount of memory required to represent performance maps triggers garbage collection which impacts the evaluation data.

Scenarios used in this part of the evaluation cover vast areas. One location in map corresponds to an area of 1×1 degree of latitude/longitude at equator which equals to 111.3 km $\times$ 111.3 km = 12387.69 km^2. Largest map in the scenarios contains 200×200 locations collectively representing 495 507 600 km^2 which is roughly the surface area of the Earth.

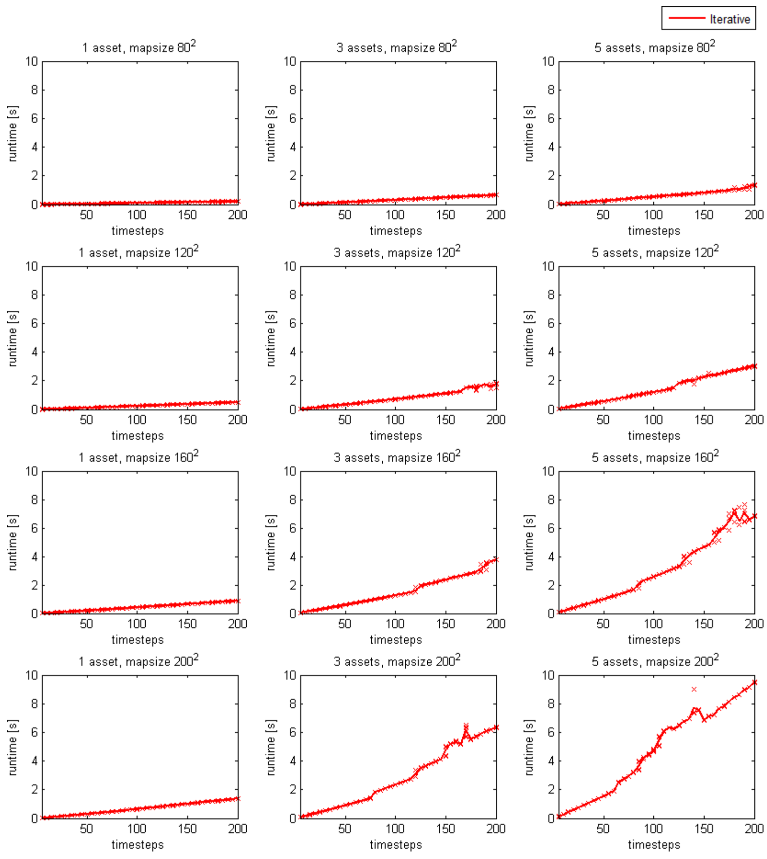


Figure 6.3: Graphs depicting the execution time dependency of the Iterative algorithm on the number of time steps in varying scenario parameters. Data plotted on linear scale suggests linear dependency. Irregularities are caused by the garbage collection present in the evaluation platform.

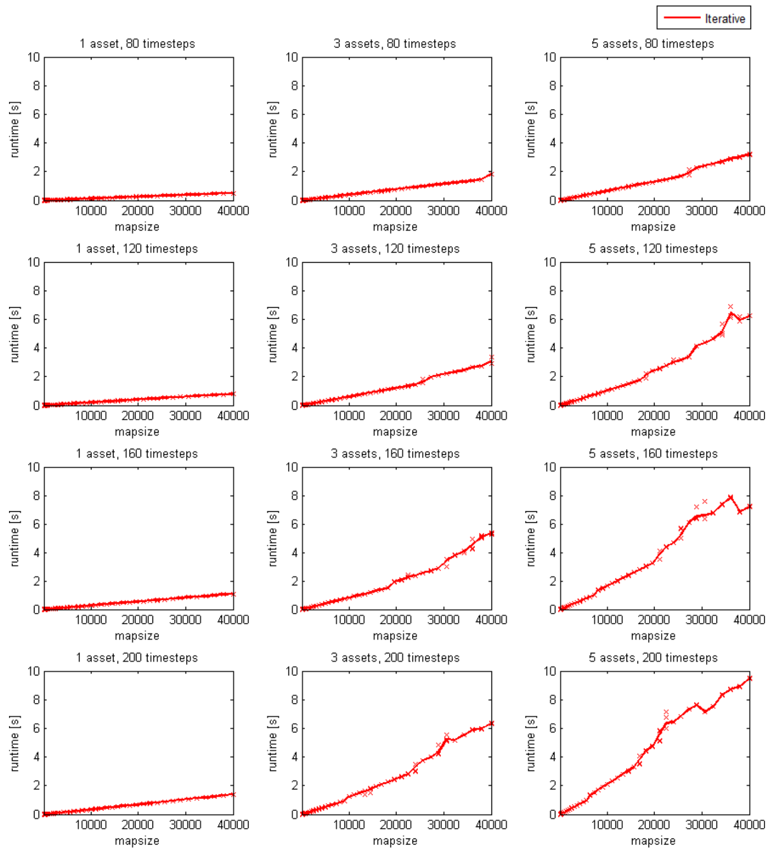


Figure 6.4: Graphs depicting the execution time dependency of the Iterative algorithm on the size of the scenario map in varying scenario parameters. Data again suggests linear dependency.

6.4 Reward Value Comparison

The second evaluated property of the two solution approaches are the reward values of the provided solutions. Since the MILP approach provides optimal solutions it can serve as a benchmark for the Iterative algorithm.

As seen in Figure 6.5, the Iterative algorithm provides solutions with same reward values as the MILP approach for scenarios with 1 asset. This confirms the optimality of the Iterative algorithm for single asset scenarios. With more assets the Iterative algorithm provides suboptimal solutions.

With the increasing number of assets the difference in quality of provided solutions grows. However, with the increasing number of time steps the percentage difference of reward values lowers. In the case of 5 asset scenarios the Iterative algorithm provides solutions with reward values equal to roughly 55% of the optimum value in the 4 time steps instances, while in the scenarios with 10 time steps it provides solutions reaching roughly

85% of the optimum value. Since real-world drug interdiction problems correspond to scenarios with 20 or more time steps, the Iterative algorithm can be expected to provide solutions with reward values nearing the optimum.

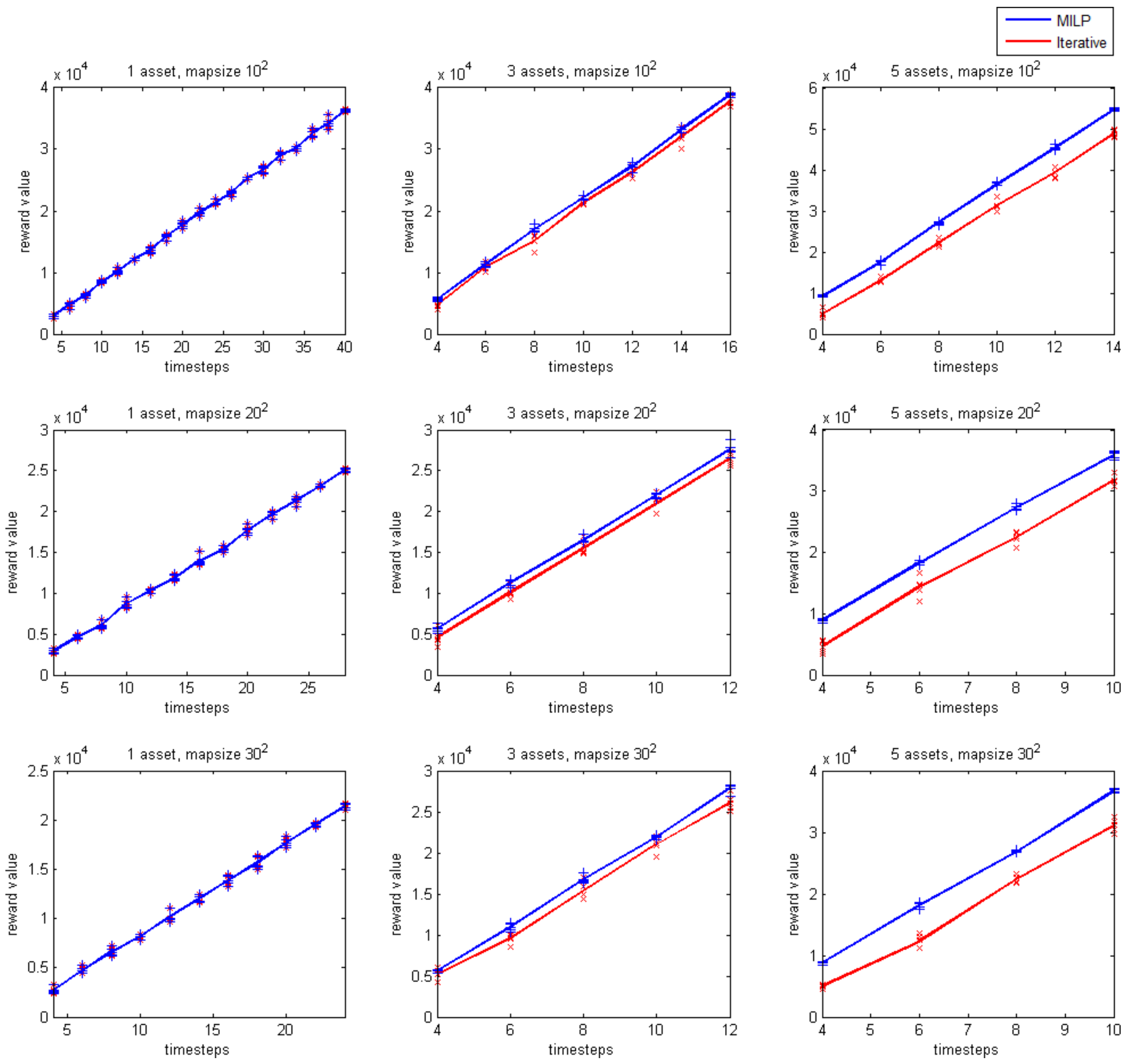


Figure 6.5: Comparison of the reward values of solutions provided by the two algorithms. In scenarios with one asset, both algorithms provide optimal solution. In scenarios with more assets the Iterative algorithm provides suboptimal solutions. Increase of the number of time steps lowers the percentage difference between the gained reward values.

Figure 6.6 shows the decrease of percentage difference between the optimal reward value and the reward value of the solution provided by the Iterative algorithm. Data from the scenarios with **mapsize** of 10^2 positions suggests that the Iterative algorithm can be expected to provide solutions with reward values distant less than 10% from the optimum for scenarios with high numbers of time steps.

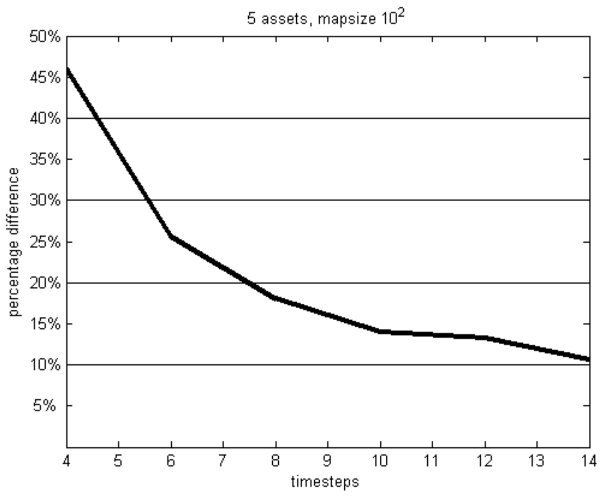


Figure 6.6: Graph of the relative error rate of the Iterative algorithm in scenarios with 5 assets and maps with 10^2 locations. With increasing number of time steps the relative error rate decreases and eventually reaches values of about 10%. Further decrease with more time steps can be expected.

Chapter 7

Conclusion

The goal of the work was to study the concept of Battlespace on Demand, gather information about contemporary drug smuggling operations in Caribbean sea and East Pacific, design and implement software architecture reflecting the Battlespace on Demand concept, develop an optimization algorithm for optimal asset allocation and evaluate performance of the algorithm on a set of scenarios extracted from the domain.

We described the Battlespace on Demand concept, studied data regarding the domain of illicit drug smuggling and presented knowledge about smuggler behaviour. We described the formal model of architecture reflecting the Battlespace on Demand concept and presented data structures and algorithms used in each layer. We proposed two approaches to use in solving the multiple asset allocation task. We executed experiments using scenarios extracted from the domain knowledge. On the results of experiments we demonstrated the impact of scenario parameter values on algorithm's computational performance and decision outcomes.

We demonstrated that in simple scenarios the mixed integer linear programming approach for decision-making is usable. However with increasing complexity of scenarios that approach requires too much time and a different approach is preferable. Implemented software architecture reflecting Battlespace on Demand concept provides an automated way to turn raw observatory data into informed decisions.

The implemented automated approach proved to be a viable option for decision-making in the maritime drug interdiction domain and could be used by organizations such as Joint Interagency Task Force South.

Bibliography

[1] United States Navy: CID EIDWS PQS Study Guide. 2011.
URL `http://www.eidws.org/114-metoc`

[2] Center for International Policy: Maps of detected trafficking routes. 2008.
URL `http://www.cipcol.org/?p=590`

[3] *2013 International Narcotics Control Strategy Report (INCSR).* Bureau of International Narcotics and Law Enforcement Affairs.

[4] United States Southern Command: Joint Interagency Task Force South.
URL `http://www.jiatfs.southcom.mil/index.aspx`

[5] Rear Admiral Jonathan White: Hydro International. 2011.
URL `http://www.hydro-international.com/issues/articles/id1275-Battlespace_on_Demand.html`

[6] Chu, P. C.; Williams, C.; Clem, T.; aj.: *Battlespace on Demand for Maritime Threats: Mine/IED Drift in the Strait of Hormuz and near Iraqi Oil Terminals.* Naval Postgraduate School, Monterey, California, USA, 2010.

[7] Stoughton, S.: *A Process for Applying Forecast Uncertainty in Planning for Underway Evolutions Along Intended Track.* Naval Postgraduate School, Monterey, California, USA, 2010.

[8] United States Navy: Global Ocean Data Assimilation Experiment.
URL `http://www.usgodae.org`

[9] Hillier, F. S.; Lieberman, G. J.: *Introduction to Operations Research, Ninth edition.* McGraw-Hill, 2010.

[10] Foderaro, G.; Ferrari, S.; Wettergren, T.: *Distributed Optimal Control for Multiagent Trajectory Optimization.* Duke University, Durham, NC, USA, 2012.

[11] Cap, M.; Novak, P.; Vokrinek, J.; aj.: *Multi-agent RRT*: Sampling-based Cooperative Pathfinding (Extended Abstract)*. 2013.

[12] Caplan, J. M.; Kennedy, L. W.: *Risk Terrain Modeling Manual: Theoretical Framework and Technical Steps of Spatial Risk Assessment for Crime Analysis*. Rutgers, 2010.

[13] Open Geospatial Consortium: KML format specification.
URL `http://www.opengeospatial.org/standards/kml`

[14] World Meteorological Organization: Guide to the WMO Table Driven Code Form Used for the Representation and Exchange of Regularly Spaced Data In Binary Form: FM 92 GRIB Edition 2. 2003.
URL `http://www.wmo.int/pages/prog/www/WMOCodes/Guides/GRIB/GRIB2_062006.pdf`

[15] Oracle Corporation: Jersey.
URL `http://jersey.java.net`

[16] Google: Google Gson.
URL `http://code.google.com/p/google-gson`

[17] Gurobi Optimization: Gurobi Optimizer.
URL `http://www.gurobi.com`

[18] IBM Corporation: IBM ILOG CPLEX Optimization Studio.
URL `http://www.ibm.com/software/products/en/ibmilogcpleoptistud`

[19] GNU project: GNU Linear Programming Kit.
URL `http://www.gnu.org/software/glpk`

[20] Martin Lukasiewycz: Java ILP.
URL `http://sourceforge.net/projects/javailp`

[21] Google: Google Earth.
URL `http://www.google.com/earth`

[22] Micromata Labs: Java API for KML.
URL `http://labs.micromata.de/projects/jak.html`

[23] Google: Google Maps Javascript API V3 Reference.
URL `http://developers.google.com/maps/documentation/javascript/reference`

[24] Open Source Geospatial Foundation: OpenLayers.
URL `http://openlayers.org`

[25] The jQuery Foundation: jQuery.
URL `http://jquery.com`

[26] The jQuery Foundation: jQuery UI.
URL `http://jqueryui.com`

[27] United States Navy: Fleet Numerical Meteorology and Oceanography Center.
URL `http://www.usno.navy.mil/FNMOC`

Appendices

CD Contents

Attached CD contains this work in PDF format its source code for LaTeXsystem. Source code of the developed application and war package for deployment are present. CD contains a prototype of the frontend which can be used to see the visualisations of the application's work. Description of this tool is in the User Guide.

The structure of CD is described in following table.

Directory	Description
`application/maven lib/`	third party libraries for maven repository
`application/source/`	source codes of the developed application
`application/war package/`	war package of the application
`frontend prototype/`	demonstration of the frontend of the application
`thesis text/source`	source files of the thesis text
`thesis text/thesis_diploma.pdf`	pdf file containing this thesis

Table 1: CD structure

User guide

Compilation

Attached CD contains built application in WAR format ready to be deployed on application server implementing the Java EE 7 specification such as JBoss Application Server.

If you wish to compile the application yourself, you can do using Apache Maven[1]. Since some of the used third party libraries are not available in public maven repositories, you will need to add them to your local repository or repository available to you under following artifacts:

- **javailp-1.2a.jar**
  `groupId: net.sf`
  `artifactId: javailp`
  `version: 1.2a`

- **jgrib.jar**
  `groupId: net.sourceforge`
  `artifactId: jgrib`
  `version: beta7`

- **edu_mines_jtk.jar**
  `groupId: edu.mines`
  `artifactId: jtk`
  `version: 1.1`

- **cplex.jar**
  `groupId: cplex`
  `artifactId: cplex`
  `version: 12.4`

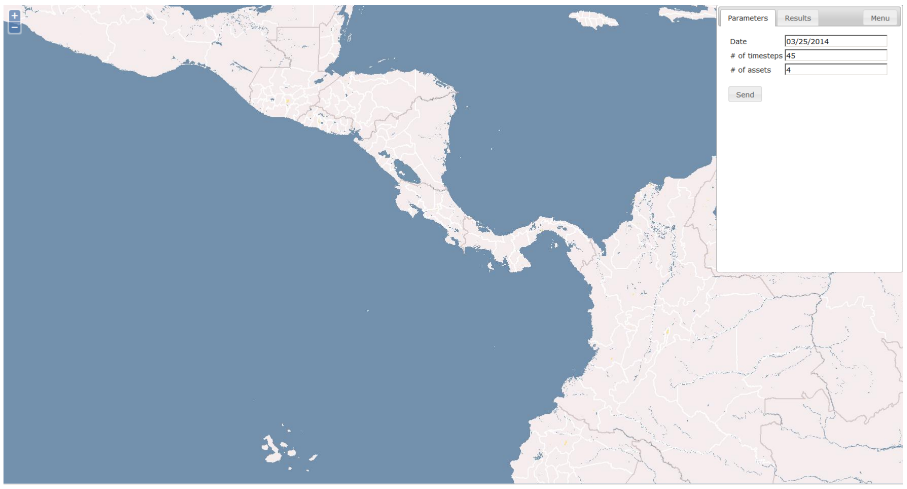
[1]http://maven.apache.org/

If you wish to use the CPLEX solver instead the default solver, the CPLEX native library version 12.4 must be present in the host server.

Application use

This section shows you how to use the deployed application with the front-end controls. If you wish to see an example of how the application works and do not want to deploy it on an application server, the attached CD contains a prototype of the front-end which works with precomputed data and allows you to see the application outputs on a predefined scenario.

At the start you will see geographical map of the region and a menu in the top right corner. In the menu you can define input parameters of the scenario you wish to run. These include the date, number of simulated time steps and number of assets. After you enter the parameters, you can click the Send button to run the scenario.



Date 03/25/2014
# of timesteps 45
# of assets 4

Figure 1: First look of the application. Geographical map of the region and menu for input parameters.

After you click the Send button, the application will run your scenario and prepare visualisation of its outputs. This may take several minutes depending on the complexity of the scenario. After the process is done, the outputs will show up on the screen as you can see in Figure 2.

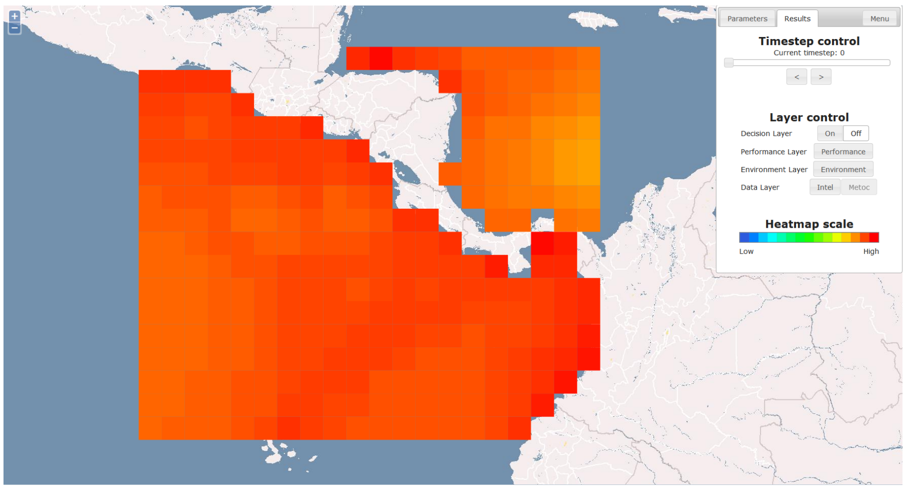


Figure 2: Displayed METOC data. User can change time steps and see how the METOC changes.

Default first visualised data is METOC from the Data Layer. In the Results tab of the menu, you can control what part of the scenario outputs you want to see by clicking on the appropriate button. Data Layer contains two data sets: METOC and Intel. Environment and Performance Layers contain each one data set.

Decision Layer visualisation shows allocation of the assets, it can be turned on or off. If turned on, it will be laid on top of any of the other displayed layer so you can see how the assets cover outputs of the other layers.

Every layer has its visualisation of all time steps so you can go through them and see for example how the METOC changes and how the assets move through the environment.

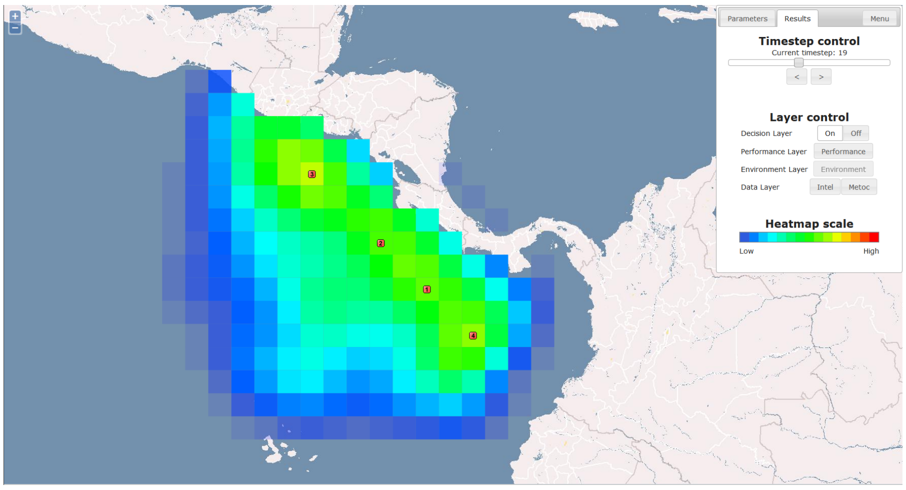


Figure 3: Displayed Environment Layer output and the Decision Layer output laid on top of it. We can see how the assets cover the simulated environment.