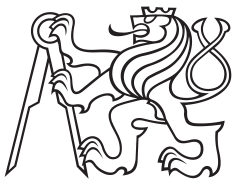
Bachelor's thesis



Czech Technical University in Prague

F3

Faculty of Electrical Engineering

Intelligent Algorithms for Monitoring of the Environment Around Oil Pipe Systems Using Unmanned Aerial Systems

Jakub Ondráček

May 2014

Supervisor: Ing. Ondřej Vaněk, Phd.

Acknowledgement / Declaration

I would like to thank my supervisor Ing. Ondřej Vaněk, Ph.D., for providing valuable advice and for his patient guidance.

Access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum, provided under the programme Projects of Large Infrastructure for Research, Development,and Innovations (LM2010005), is greatly appreciated.

This research was funded by the Office of Naval Research Global (grant no. N62909-11-1-7034).

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 23.5. 2014

.....

Abstrakt / Abstract

Efektivní monitorování ropovodních systémů je důležitým úkolem vzhledem k velmi vysokým škodám na životním prostředí způsobenými možnými nezjištěnými úniky ropy. Pro tento úkol se zaměřujeme na využití bezpilotních letounů. Formulovali jsme problém optimálního monitorování ropovodů pomocí několika mobilních agentů jako matematický program, který bere v potaz vlastnosti problému včetně sensitivity okolního prostředí, vlastností ropovodů a omezení pohybu bezpilotních letounů. Také jsme vytvořili tři algoritmické extenze, které zlepšují škálovatelnost našeho problému, a rozšířili jsme jej o problematiku optimálního rozmístění bází. Nakonec jsme ukázali, že i přes omezenou škálovatelnost jsme schopni najít optimální řešení pro reálně velké problémy a ukázali jsme nadějný způsob, jak přibližně řešit velké scénáře.

Klíčová slova: Optimalizace, Bezpilotní letouny, Matematické programovaní, Zabezpečení infrastruktur

Effective monitoring of oil pipeline systems is an important task due to a very high environmental cost of an undetected oil spill. For this task, we focus on the utilization of unmanned aerial vehicles (UAV) and we formalize the problem of optimal pipeline monitoring with multiple mobile agents as a mathematical program which captures properties of the problem, including environmental sensitivity, pipeline properties and motion constraints of the UAVs. We design three algorithmic extensions that push inherent scalability limits of this problem. We also extend the problem of area surveillance by problem of optimal bases placement. Finally, we show that even with limited scalability, we are able to find optimal solutions for real-word sized problems and show a promising way to approximately solve larger scenarios.

Keywords: Optimization, Unmanned Aerial Vehicles, Mathematical Programming, Infrastructure Security

Contents /

Tables / Figures

Chapter 1
Introduction

Oil pipeline systems belong to a set of critical infrastructures which are vital for contemporary society; however, by their inherent spread throughout a wide area and poisonous fluid transported, they pose a serious threat to the environment. Additionally, many oil pipeline infrastructures have obsolete equipment and underdeveloped monitoring systems. One of these is oil pipeline system in Nigeria where, between 1976 and 1996, authorities report nearly 5 thousand incidents of oil pipeline damage [1]. These incidents resulted into spills of over two millions barrels of crude oil (over 280 million liters) and over 84 percent of the cumulative spill was lost to the environment (According to unofficial sources, the amount of leaked oil was much higher). Oil leakages can cause some serious problems [2]; specifically loss of soil fertility, pollution of fishing waters - leading to loss of fish population and other aquatic organisms - pollution of drinking water, health problems on people or degradation of mangrove ecosystems, as it can be seen in Figure 1.1. In addition, it is expected that the problem will be more and more significant, due to the global problem of oil pipelines aging. Oil spills have a large negative impact not only on the environment but also on the companies engaged in the oil exploration itself of the three main reasons: firstly, companies lose profit by oil spills, secondly oil spills have a negative effect on the public opinion of the companies and thirdly companies are threatened by large fines for oil spills. For example, the Nigerian Maritime Administration and Safety Agency (NIMASA) and the National Oil Spill Response and Emergency Agency (NOSREA) have ordered Shell to pay $11.5 billion as fines and compensation for the 2011 Bonga oil spill incident [1]).

Figure 1.1. The oil-polluted waters of Bodo Creek in Nigeria. Photo Jane Hahn,The New York Times.

[1]) `http://royaldutchshellplc.com/2014/01/30/nigeria-bonga-oil-spill-fg-fines-shell-11-5-billion/`

Due to the increased interest and exploitation of UAS (Unmanned Aerial Systems) in commercial use, we decided to use UAS, which have been applied in many cases of infrastructure monitoring to solve this problem.

The goal of this thesis is to create a set of algorithms, which are able to solve the problem of pipeline system monitoring in a form of a strategy on a graph representing the pipeline infrastructure. A number of works that deal with problem of surveillance were proposed, however, these solutions are mostly based on problem of finding optimal cycle in a graph. We use a different approach where we do not attempt to find one circle, but we look for the optimal route for given time horizon. This decision is based on the fact that pipeline systems are so large that, with regard to the physical limitations of UAS, they cannot be patrolled through one cycle.

In our work, we created a basic patrolling model that optimally solves the problem of patrolling the pipeline system. We have also proposed a set of extensions considering real restrictions of UAS such as endurance of UAS or recharge time. We evaluate the proposed algorithms both on synthetic data on which we measure the influence of single parameters on scalability and quality of the obtained solutions. We also create a set of scenarios that are based on real-world pipeline systems. Due to the fact that solving the surveillance problem is computationally hard, we eventually developed a set of speed-up techniques with which we are able to solve problems of real world size.

During our research our paper Monitoring Oil Pipeline Infrastructures with Multiple Unmanned Aerial Vehicles [3] was published on Conference on Practical Applications of Agents and Multi-Agent Systems(PAAMS)[1]). This work is based on this paper and extends it.

Goals of the Thesis

Our work has a following goals

- **Study the problem of environmental protection in the vicinity of oil pipes.** We study the problem of environmental protection in the vicinity of oil pipelines in chapter 3. First, we explain the structure of pipeline transport system. Then we show causes of oil pipelines failures together with possibilities how they can be prevented. We also propose an overview of current methods of oil pipelines monitoring. In the second part of this chapter, we discuss the impact of oil spills together with techniques by which the sensitivity of environment on oil spills can be expressed.
- **Study the utilization of unmanned aerial systems (UAS) for infrastructure monitoring.**

  In chapter 2, we propose an overview of related work dealing with a problem of area surveillance or infrastructure monitoring. In chapter 4, we introduce current possibilities of UAS in commercial sector and we evaluate benefits and limitations of using UAS for pipeline system monitoring.
- **Design and implement a set of algorithms able to plan trajectories for a set of UAS monitoring the pipelines.**

  We create a basic algorithm which finds a plan for optimal surveillance of given pipeline system on the given time horizon as well as a set of extensions which reflect real world restrictions of UAS such as endurance or recharge time. We propose two versions of these algorithms: node-oriented and edge-oriented. Because finding an optimal route is computationally hard, we create a set of three speed-up techniques namely: iterative algorithm with guaranteed quality solutions, optimal area

[1]) `http://www.paams.net/`

decomposition algorithm and sub-optimal time decomposition algorithm. We also propose three different area discretizations on which the problem can be solved and two different time discretizations.

- **Create a set of scenarios for oil-pipe patrolling.**
  We create two real world scenarios. The first one is pipeline system in Rostock's harbor and the second one is pipeline system of oil field in Russia. See Section 8.2 in Chapter 8.
- **Evaluate the designed algorithms on created scenarios.**
  We evaluate proposed algorithms both on synthetic data, where we show the influence of single parameters and techniques on scalability and quality of the solution, as well as on real world scenarios where we show possibility of using proposed algorithm on real world size problems. See Chapter 8.

Structure of the Thesis

Chapter 2 provides an overview of published work related to our topic. The chapter is divided into three subsections where the first deals with history of linear programming (LP) and of integer linear programming (ILP), the second deals with patrolling pipelines and the third deals with utilizing UAS for area surveillance.

Chapter 3 focuses on oil pipelines as well as current methods of their surveillance. The problem of oil spills on the environment, type of pipelines' damage and factors affecting their emergence is also described in this chapter. In addition, this chapter provides an overview of current methods of monitoring pipelines together with their advantages and disadvantages.

Chapter 4 focuses on the overview of UAS application in infrastructure monitoring as well as on currently available commercial solution of UAS. The chapter ends with discussions with the advantages and limitations of UAVs and their potential use for pipelines patrolling.

Chapter 5 explains what the LP and ILP are as well as the algorithms used to solve ILP problems. Later in this section, the concept of the submodular set function is explained.

Chapter 6 explains the discretization of the oil pipeline system into a graph. Time discretization of the given time horizon is also described. Further, in this chapter, we formally define an agent's movement over the pipeline and utility functions.

Chapter 7 describes different algorithms solving defined problem. We additionally describe the issue of the bases placement.

Chapter 8 is divided into 2 parts; the first part is devoted to the evaluation of the impact of individual representations, discretization algorithms, parameters, and speed-up techniques on scalability and quality of the solution. The second part then provides results of evaluation on real-world scenarios that focus mainly on speed up techniques and spatial discretization.

Chapter 2
State of art

2.1 History of LP and MIP

This summary is based on work of R.E. Bixby [4] in which it can be read more about the history of linear programming (LP) and integer liner programming (ILP). Work of George Dantzig can be considered as the beginning of linear programming. Dantzig viewed LP as a method, which can be used for solving specific real world problems. He proposed a general algorithm for solving LP problems called Simplex Algorithm [5] in 1948. (Klee and Minty [6] proved that Dantzig's version of Simplex Algorithm has combinational time complexity in the worst case). The Simplex Algorithm is one of the most important algorithms for solving LP and ILP problems to this day. It was also marked as one of the top ten most important algorithms of 20th century. The first use of Simplex Algorithm is dated to year 1947 when Laderman solved 21 constraints, 77 variable problem instance of Stinger's Diet Problem with the total computational time equals to 120 man-days.

The first computer implementation of simplex algorithm was made in year 1953 on SEAC computer at National Bureau of Standards. The algorithm was re-formulated to IBM 701 computer in years 1954-1955. This implementation was able to solve LP instance up to 255 constraints. The version for IBM 7090 was released in years 1961-1962, which was able to solve LP instance up to 1024 constraints. LP 90/94 (LP code for IBM 7090 and 7094) was a milestone of LP, because it was the first commercial code using Branch-and-Bound technique for solving ILP. Thus, it was able to solve real world problems, which were unsolvable optimally until then, for example:

- The selection of ships and transport aircraft to support deployment of UK military assets
- Refinery infrastructure investments by British Petroleum
- Re-location of factories in Europe by Philips Petroleum

It was demonstrated for the first time that ILP based on Branch-and-Bound could be used for solving real world problems. It should be noted that at the end of 60's LP started to be used in oil Industry [7] as well.

In 70's and 80's, few of important improvements of LP were proposed such as implicit treatment of bounds, which reduced the number of explicit constraints in the mode, the use of LU-factorizations. Dual simplex algorithm proposed by Lemke [8] in 1954 was used for optimization within ILP branch-and-bound trees. Therefore, the ILP was developed into more powerful tool used for applications which were more extensive in practice. Khachiyan [9] firstly proved that LP could be solved in polynomial time in 1979. However, applicability of this computation was limited thus using of Khachiyan's algorithm was quickly abandoned. In 1984 Karmarkar [10], used projective transformations to show polynomial time bound LP, which was a bit better than Khachian's algorithm but also corresponded to computation approach that was applicable in practice. This algorithm is known as Interior Point Method or Barrier method. Karmak's

work renewed interest in the LP. The most important advances occurred during this time were:

- The emergence of the dual simplex algorithm as a general-purpose solver
- The development of dual steepest-edge algorithm
- Cholesky Improved factorization method for barrier algorithms and introduction of parallelism in this algorithm
- Vastly improved linear algebra in the application of simplex algorithms for large, sparse models

For ILP, the 70's was a milestone, with introduction of several algorithms using what were then state of art implementations of simplex algorithms tightly integrated with LP based branch-and-bound algorithm and combined with a wide variety of simple, but effective heuristics for overall search improving. Crowder at al. [11] published paper about using cutting-plane techniques for solving 0-1 IP in 1983, where they extended the general approach of Crowed and Padberg [12]. Roy and Wolsey [13] demonstrated effectively of cutting-plane techniques and ILP presolve reduction in solution of real world ILP problems, which were unsolvable to optimally until then in 1987. MINTO code was created in 1991; it was a first code, which systematically used cutting-plane techniques, moreover it was a research code, thus provided important tests for the effectiveness of these methods, however, it has never been used in practice.

The first version of CPLEX was released in 1988. CPLEX was dominant solver in early 90's, which contained implementation both primal and dual simplex method as well as barrier algorithms. You can see more about CPLEX in Section 5.3.1.

2.2 Oil Pipelines and Their Protection

Allen at al. [14] published a paper where they reviewed the utilization of UAS in operational oil spill surveillance, monitoring and assessment. They analyzed both the advantages and the disadvantages of current UAS and the possibility of their use in oil industry as well as ways to improve the possibility of using UAS. The result of the paper is that technology of UAS should be incorporated into spill response tactics and strategies.

Gundlach at al. [15] first described the Environmental Sensitivity Index (ESI) concept of using a scale of 1 to 10 to indicate vulnerability of costal environments to oil spill impact. Where one is the least vulnerable belong for exposed rocky headlines, where the cleaning is not necessary and 10 is the most vulnerable this evaluation have, for example, mangrove forests. The principle of ESI map was later extended also for the inland, and we used it in our work to evaluate the sensitivity of environment around pipeline systems.

Petersen at al. [16] proposed a guideline for creating ESI which provides an overview of all the properties that reflect the ESI maps as well as an overview of the formation of ESI maps.

Agbakwuru [17] proposed paper with summary of current pipeline potential leak detection technologies and assess their applicability in the Niger Delta, where one of the mentioned technology are remotely piloted aerial patrol drones.

Hopkins at al. [18] proposed work dealing with structural integrity of oil pipelines. In the first part, they provide an overview of pipelines like their types and errors on pipelines and the possibility of their detection. The second part of this work deals with the structural integrity and risk management of oil pipelines. We use insights from this

work in our work to formulate the properties of oil pipelines determining susceptibility to damage.

Dawotola at al. [19] proposed a Multi Criteria Decision Analysis (MCDA) framework for risk management of oil and gas pipelines, utilizing an Analytic Hierarchy Process (AHP) to prioritize oil and gas pipelines for design, construction, inspection and maintenance. They created a case study application on pipelines in Nigeria to demonstrate the proposed methodology. Unlike to our work it is not a direct search strategy for monitoring but a complex system combining several different methods of protecting pipelines and maintenance to avoid the chance of failure. Dey [20] proposed a similar work, in which he described proposed a risk-based decision support system (DSS) that reduces the amount time spent on inspection, utilizing an AHP as well as Dawolta used in his work.

Jawhar [21] proposed a framework for pipeline infrastructure monitoring using wireless sensor networks. In his work, he discuss the issues and challenges of using wireless sensors and proposed an architectural model including an overview of networking and protocols, which can be used to provide monitoring a control function.

2.3 UAS Surveillance

Krause at al. [22] publish an approach dealing with near-optimal sensor placement for water distribution networks. The biggest improvement of this work is that they show that the problem is submodular and formulate a greedy algorithm with proved theoretical lower bound about 63%. In our work, we use the fact that even our problem is submodular and we use a greedy algorithm as one of our speed up technique. Additionally, we extend greedy algorithm about a problem with non-homogenous set of agents, where we are not able to choose the best agent easily.

Jakob at al. [23] describes problem of multiple aerial surveillance of urban areas by UAVs with explicit consideration offsensor occlusions. As well as in our work their goal was to minimize the age of information however, they do not consider the importance of each area. They solve the problem by dividing it into two sub-problems. The first sub-problem is the division of patrolling multiple areas into a single area and the second one is a problem of single area surveillance. This is similar to our area decomposition speed up technique. Unlike to our work they do not solve the joint plan of agents but primarily solve single UAV surveillance, which extends into multiple UAV surveillance using three different suboptimal techniques. The result of the work is that the best solution of patrolling with respect to offsensor occlusions is using the zig-zag algorithm, which is however unusable in our work because of the size of oil pipeline systems.

Chevaleyre [24] proposed an analysis of the multi-agent patrolling problem. He considered two different patrolling strategies. The first one is a cycling strategy, which is based on traveling salesman problem. He proved that solving patrolling problem as traveling salesman problem is optimal for single UAV patrolling and also proposed that for multiple UAV can be solved near optimally with time complexity $O(n^3)$. He also shows, that for graphs with long `tunnels` it is better to use a second so-called Partition-based strategy, where graph is divided into r disjoint regions and each region is patrolled by a single UAV. The work is similar to ours - we also solve the problem as a cycling strategy, however Chavaleyre does not consider agent's endurance nor time horizon as we do. For this reason we cannot take this strategy, because we may not be able to find a solution of TSP, whatever the reason of endurance or given time horizon. We use Partition-based strategy but only so that the resulting strategy is optimal.

Nigam at al. [25] proposed sub-optimal semi-heuristic patrolling algorithm for single UAV. They also proposed two extensions for multiple UAV patrolling problem. The first extension is based on an extension of the algorithm itself and the second one is based on a similar strategy as the Partition-based strategy proposed by Chevaleyre, where given area is decomposed into sub-regions which are allocated to individual UAVs for parallel exploration. The works differs from the existing work by creating the trajectory for UAVs with respect to the aircraft dynamic.

Pasqualetti at al. [26] described an approach dealing with a cooperative patrolling from robotics point of view via Weighted Tours, where identical holonomic agents are moving over minimum spanning tree through the viewpoints. Because solving cooperative patrolling is computationally hard, Pasqualetti solves the problem with sub-optimal but bounded algorithm, which is based on work of Chevaleyre. They also proposed two communication scenarios. Work is similar to ours in that it also considers different priorities of individual parts of the graph, however, it is not optimal and it does not consider heterogeneous set of agents.

Hrstka [27] deals with a problem of harbor security surveillance using UAS. He solves the problem as a search of Hamiltonian path for each agent in polynomial time over the triangular graph, which must be solid; otherwise, it is not possible to find a Hamiltonian path in polynomial time. As well as Hrstka, we use a triangular solid graph as one of our discretization; however, as it turns out, this discretization is not the best for our work. The reason why we cannot take this solution is that he simplifies the problem with assumption that either every two patrollers can share their paths completely or their paths are disjunctive.

Our work deals only with finding the optimal route, thus for the real-world usage it is needed to create trajectory based on planted path for single UAVs. This problem can be solved for example work of Selecký at al. [28], in which they introduce a technique based on accelerated A*, which allows to take the wind effects into account and generate trajectory reachable by UAV.

Chapter 3
Pipelines Infrastructure

In this chapter, we propose a basic overview of oil pipelines. At first, we explain a structure of pipelines, here should be noted that in our work we deal only with onshore or inland pipeline systems. Then we focus on an emergence of damage of oil pipelines and the factors that affect the probability of their emergence, we also propose and discus current methods of pipelines monitoring. In the second part of this chapter, we discuss the influence of oil spills to environment and possibilities how to determine environmental sensitivity of a given area.

3.1 Pipeline System

Pipeline systems consist of two parts: pipelines themselves and components that operate together to move products from location to location. In general, pipelines can be classified in three or five[1])categories depending on purpose:

- **Gathering pipelines** are short distance lines with small diameter usually in range from 50mm to 305mm, which gather an oil from mining towers in oil fields and move them to processing facilities or a storage.
- **Feeder pipelines** are lines with a diameter up to 508mm which move oil from processing facilities or storage to the main transmission lines.
- **Transmission pipelines** are the main long pipes (USA's pipeline system is over 250,000 km in length), which can have a very large diameter (up to 1422mm), moving oil between cities, countries and even continents. These pipelines include several pump stations and often are buried in land, thus we don't mainly focus on them in our work.
- **Product pipelines** are lines similar to feeder pipelines, which move oil from the main transmission lines to distribution centers.
- **Distribution pipelines** are low pressure pipes which are used for local distribution from a distribution centers. They can have a large diameter, but most of them are under 152mm diameter.

Components of oil systems are:

- **Initial injection station** is the beginning of the system, where the oil is injected into the pipeline.
- **Pump station** is s facility including pumps and equipment for pumping through the pipeline. Its location along pipelines is determined especially by elevation of the pipelines.
- **Partial delivery station** is a facility which allows to deliver part of the oil being transported.

[1]) In some works Gathering and Feeder pipelines and Product and Distribution pipelines pipelines are not distinguished.

- **Block valve station** is a valve station by which the operator can isolate some segment of the line for maintenance work or isolate a rupture or a leak. Block valve stations are usually located every 32 to 48 km depending on the design of pipeline system. Even though it is not a design rule, it is just a usual practice, for example one of the problems of pipeline systems in the Niger Delta is a large spacing of block valve station.
- **Regulator station** is a special type of valve station, where the operator can release some of the pressure from the line.
- **Final delivery station** is a place where the product is distributed to the consumer. It could be a tank terminal or connection to a distribution network.

3.2 Pipeline Failures

Causes of Oil Pipeline Failures can be divided into 5 types: mechanical failures, 3rd party activity, corrosion, operational error and natural hazards. Their proportional distribution is different for each pipeline system. Table 3.1 shows proportional distribution for Niger Delta System.

Causes	Distribution (%)
Mechanical failures	42
3rd party activity	24
Corrosion	18
Operational error	10
Natural hazards	6

Table 3.1. Causes of Oil Pipeline Failures [29].

Natural hazards represent failures caused by earthquake, floods, landslide etc. These are failures that we cannot prevent in any way; however, we can assume the probability of their occurrence by observing the properties such as stability of the subsoil or rainfalls in the given area. Operational errors are caused mostly by human error or sometimes intentionally when the regulations are not strict followed. These failures cannot be even estimated.

Each pipeline system slowly corrodes regardless whether it is buried in land, under the sea or placed overground. The corrosion is the main reason of pipeline failure in USA. It is very often the initial cause of mechanical failures of pipelines, thus a percentage distribution of mechanical failure is so high at the expense of corrosion (see Table 3.1). We distinguish two types of corrosion: internal corrosion and external corrosion. Today there are many ways how to prevent corrosion using a wide range of corrosion protection. Other ways to prevent damage due to corrosion is early detection, before there is a leakage from oil pipelines. This can be accomplished by means of pigging or by laser or ultrasonic inspection (see Section 3.3). The priority of monitoring single segments of the system can be determined by results of these inspections and according to the applied corrosion protection. Two main reasons of pipelines failures belonging to 3rd party activities are sabotage and stealing. It is possible to use an UAS in proactive way to prevent them. The game theory deals with this issue however, our work does not take this issue into account.

One of the biggest current problem of pipelines industry is fact that pipeline infrastructure is ageing, which has an important influence on pipeline reliability. How you

can see on the table 3.2 a reliability of pipelines older than 30 years is half beside with pipelines younger than 20 years. For example:

- In 2010 40% of pipelines in Nigeria were older then 30 years.
- More than 50% of the 1,000,000 km USA oil and gas pipeline system is over 40 years.
- Over the next 15 years 50% of Russia's oil and gas system will be at the end of its design life.

Age (Years)	Reliability (%)
< 20	46
20-30	29
> 30	25

Table 3.2. Age and reliability [29]

Table 3.3 shows a number of incidents on 1000km of pipeline during one year for western world pipelines. The numbers look relatively small, but if we consider the fact that only Transmission pipelines in USA's pipeline system is over 250,000 km in length we get 1,000 incidents that require a repair, 150 incidents that result into loss of oil and 40 high cost incidence per a single year. Thus more than three failures emerge on the USA's pipeline system every day.

Incident	Frequency (incidents/1000km year)
Incident Requiring Repair	4
Failure (loss of product)	0.6
Failure (casualties and/or high costs)	0.16

Table 3.3. Benchmark Incident Rates for Western World Pipelines [18].

3.3 Current Methods of Pipeline Monitoring

According to [17] the current methods of pipeline monitoring can be divided into 9 main groups, how it can be seen together with theirs application in Table 3.3. In practice many of another techniques of pipeline monitoring can be found, however, they are just a sophisticated solution of methods mentioned in table 3.4 or theirs combination. The methods of pipeline monitoring can be divided by three different ways. The first dividing is to off-line and online methods. The off-line methods are used for periodic no real-time checks of oil states. The online methods are used for real-time control of pipelines and prompt detection of incurred errors. The second possibility how methods can be divided is dividing into proactive and reactive methods. The proactive methods are used to avoid the failure of oil pipelines findings degraded parts of both deformation and rust even before leaks. The reactive methods are not able to prevent failures and thus their aim is to detect the fault as quickly and accurately as possible. The last possibility how to divide methods is dividing to inner of outer monitoring, on whether an inspection is from inside or outside the pipeline, where internal monitoring is performed by pigging.

Method	Examples of tool/systems that use the method
Laser Scanning	Laser scanning, Buckle detectors, etc.
Ultrasonic	Intelligent pigging, Automatic ultrasonic tester, TOFD, Ultrasonic probe testers etc.
Acoustic	Acoustic Leak detector, hydrophones, Electromagnetic Acoustic Transducers, piezoelectric meter etc.
Fibre Optics	Optical sensors (for leak, strain, fatigue and ground movement detection), etc
Visual Inspection	Inspection light, Robotic crawlers etc.
Magnetic flux leakage method	Intelligent pigging, Eddy current, Magnetic particle inspection etc.
Inventory accounting (pressure differentials, mass flow-rates etc)	Negative pressure wave detectors
Flourometry Leak detection sensors	Fluorescence and Hydrochemical detectors
Temperature based sensors	Thermal spray technology etc.

Table 3.4. Categorisation of leak detection methods [17].

3.4 Environment Sensitivity Ranking

One of the problems, which must be solved during monitoring an oil pipelines system, is how to determine priority of monitoring for each single segment. Formal definition of the utility function is proposed in Section 6.5, where one of the items taken into consideration is sensitivity of the environment around the pipeline i.e., how much the segment is prized from a biological point of view. One of the best-known ways of area ranking is Environment Sensitivity Index (ESI) proposed by Gundlach at al. [15] in 1798. It describes sensitivity of area to oil spills on a scale from one to 10 where one represents the lowest sensitivity and 10 represents the highest sensitivity. The advantage of this ranking is a fact, that it was developed especially for measuring a sensitivity of environment to oil spills; however, it has also some disadvantages. The main disadvantage is that it was developed for ranking of costal environment and thus it is more vital for a problem of offshore pipeline protection. It should be noted, that another possibilities to determine the sensitivity of the Environment also exit for example some parts of Environmental Vulnerability Index [1]) or Environmental Sustainability Index [2]) can be used.

[1]) `http://www.vulnerabilityindex.net/`
[2]) `http://www.yale.edu/esi/`

Chapter 4
UAVs and Their Utilization

UAV (Unmanned Aerial Vehicle) also called a drone is an unmanned aircraft that can be controlled remotely or fly separately using pre-programmed flight plans using more complex dynamic or autonomous systems. UAV concept dates back to 1849, when Austria attacked Venice using balloons carrying explosives, but the boom of usage UAS (Unmanned Aerial System) in civil use has occurred since the last few years.

Currently, many different solutions using UAVs for surveillance are available, whether in the form of individual drone with control software [1]), or complete surveillance systems using UAS as one of its components. Companies Aero Surveillance and AeroVironment can be mentioned for example.

Allen at al. [14] in their work published below list of the possible useage of UAS in civil sector.

- Homeland Security - border patrol, coastal marine transportation, drug interdiction, high risk facilities (terrorist targets) and major event security surveillance.
- Law Enforcement - criminal activity surveillance, traffic control monitoring, emergency evacuation assessment, hostage rescue, crime scene investigation.
- Natural Disaster - early warning assessment and post-impact emergency response to hurricanes/typhoons, tsunamis, earthquakes, floods, and wildfires.
- Marine Casualty and response surveillance and monitoring.
- Oil & Gas Industry surveillance and monitoring of terminals, refineries, offshore E&P platforms and pipelines.
- Oil and Hazardous Substance - spill surveillance, detection, monitoring both pre- and post-incident, including pre-impact assessment and natural resources damage assessment.
- Wild Fire/Forest Fire - monitoring & assessment.
- Search and Rescue - marine and terrestrial.
- Wildlife - monitoring of the migration, population count and assessment, rescue and rehabilitation.
- Agriculture - high valued crop surveillance, detection and monitoring.

UAS have several properties that determine their usefulness. The most 4 important properties are: endurance, maximal payload, maximal speed(or cruising speed) and range of signal. In Table 4.1 we introduce values of these properties for above mentioned UAV models together with the values of Predator B which is currently the best non-military UAV used by NASA under name 'Ikhana'[2]).

[1]) There is a large range of offered drones from micro drone specifically mentioned for example eBee system proposed by Parrot company to drone approaching the quality of military unmanned aircraft with flight time in tens of hours and a range of hundreds of kilometers like Penguin B is

[2]) `http://www.nasa.gov/centers/armstrong/news/FactSheets/FS-097-DFRC.html`

Model	Endurance	Payload	Max. speed	Signal range
eBee	50 min	< 0.5 kg	57 km/h	3 km
Penguin B	24+ hours	10 kg	136 km/h	150 km
Predator B	24+ hours	1,700 kg	482 km/h	Global

Table 4.1. Examples of commercial UAV

Benefits and Limitations of UAS

The use of UAS for pipelines system monitoring has indisputable advantages, however, some limitations and problems that remain to be solved. Let us start with the negative side of using UAS: perhaps the most important problem of UAS usage is the fact, that globally no protected frequencies are available for commercial use. The next limitation is range of UAS, as it can be seen in table 4.1, UAS with radius of hundreds of kilometers exists, however, most of commercial solutions of UAS are focused on the micro UAV with endurance in rage around maximally 2 hours and small speed, thus the range is very small relative to the size of the pipeline systems. Of course, commercial UAS without this problem for example above-mentioned Penguin B are already available, however, their price is significantly higher. The last disadvantage is that UAS is not suitable for the 24/7 monitoring which can be useful in very exposed areas of systems where the risk of significant damage exits.

As a main advantages of using UAS can mention the following: The first advantage is that it requires no input into the existing system, unlike some current methods that must be included already in the development of the system. Thus, it is advantageous to use UAS in places where the replacement of the system or its major modernization is either not possible or is not financially cost-effective. The next advantage of UAS is that it can be used unlike other methods for protection against 3rd party activities in a proactive way. But their biggest advantage is its versatility. On one hand it is not static therefore it can be used to iterative monitoring of more pipeline system according to their current significance. It can also be used for multiple purposes, for example in the case of the oil spills, detection it can switch from pipeline system monitoring into area monitoring and monitor the movement of oil spills and thereby streamlining the process of rescue work.

In spite of some usage problems of UAS which, however, are along with the popularization of UAS usage in recent years rapidly removing, utilization of UAS for pipelines monitoring is an interesting technique which has its uses especially when gathering and products lines are patrolled, where monitoring can be associated with monitoring connected complex network. Simultaneously, these pipelines are mostly over-ground and it is a very dense network in a relatively small area.

In Table 4.2 you can see a summary of benefits and limitations of using UAS for oil pipeline systems monitoring.

Advantages	Disadvantages
Don't require intervention in the pipeline system	Applicable only for overground pipelines
Can be used for protection against 3rd p.a.	Limited range when micro UAVs are used
Aren't static and may periodically move	The issue of costs*
Upon detection can be used for other purposes	Not suitable for 24/7 monitoring of a one segment
	Problem with frequencies for UAV ussage

Table 4.2. Summary of benefits and limitations of UAS

Chapter 5
Technical Background

In this chapter we introduce a basic general background of the Integer linear programming (ILP), which we use for solving our problem. Firstly, we describe what ILP and LP are and how they can be represented. Then we show a proof, that ILP is NP-complete. We also explain the three most used techniques for solving ILP, namely Branch-and-Bound algorithm, cutting plane technique and Branch-and-Cut algorithm. In the end of this chapter, we describe what the submodular set function is and explain a iterative algorithm for maximizing of submodular problem with lower bound about 63%.

5.1 Overview of LP and ILP

Linear programming is a technique for optimizing linear objective function according to restrictions defined by a set of linear equations or inequations. The set of feasible solution is polyhedron that is defined by an intersection of the finite number half spaces where each half space is defined by a linear restriction, thus linear programming finds a maximal or minimal point of polyhedron. More formally the polyhedron P is $P(A, b) = \{x \in R_n : Ax \leq b\}$, where A is $m \times n$-matrix and $b \in R_n$ is a vector. The polyhedron P is called a rational polyhedron if A and b can be chosen to be rational. If a polyhedron is bounded then it is called polytope. Definition of integer linear programming is similar, however, the variable need not to be continuous. In the case where all variables need to be integer, the problem is called a pure integer linear programming problem. If all variables need to be 0 or 1 the problem is called 0 - 1 linear programming problem. The polyhedron $P \subset R_n$ is formulation of ILP for set $X \subset Z^p \times R^{n-p}$, if $X = P \cap (Z^p \times R^{n-p})$. As you can see in Figure 5.1, there is infinite number of formulations for a set X, so we want to know how `good` the given formulation is. Consider that we have a given set S and two formulation P_1 and P_2, we say that P_1 is better than P_2 , if $P_1 \subset P_2$

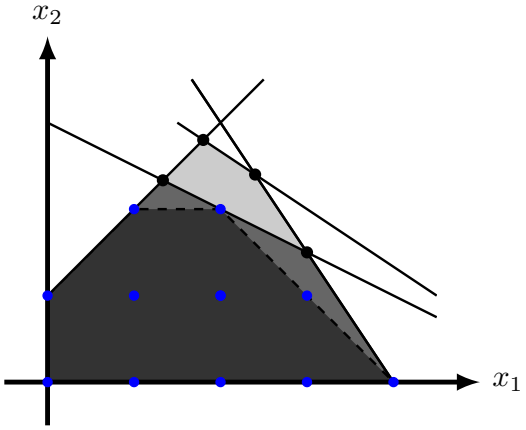


Figure 5.1. Example of different formulation of ILP.

The formulation P for set X is called optimal if $P = conv(x)$, in this case we can solve a ILP problem just by solving LP relaxation. As we show below, finding of optimal formulation is one of the keys of solving ILP.

The formulation P for set X is called optimal if $P = conv(x)$, in this case we can solve a ILP problem just by solving LP relaxation. As we show below, finding of optimal formulation is one of the keys of solving ILP.

There are two main possibilities how to represent linear programming problem. The first possibility is represent a problem in the standard form (It should be noted that names differs in different books. By standard maximum form, we mean the form that Dantzig called Canonical form). Standard form (5.2) representation has three conventions. The first one is that we maximizing the objective function. The second one is that all constraints are inequalities in form $Ax \leq b$. The last on is that all variables are nonnegative. Other possibility is dictionary form (5.1) with the following conventions: objective function is minimize, all constraints are equalities and all variables are nonnegative. Each LP problem can be converted both to standard form and to dictionary form by using simple rules.

$$\begin{aligned} &\max c^t x \\ &Ax = b \\ &x \geq 0 \\ &x \in R \end{aligned} \tag{5.1}$$

$$\begin{aligned} &\max c^t x \\ &Ax \leq b \\ &x \geq 0 \\ &x \in R \end{aligned} \tag{5.2}$$

5.1.1 Complexity of Integer Programming

The 0-1, mixed integer or pure integer linear programming problems are NP-complete. Let's start with 0-1 problem:

$$\max c^t x \tag{5.3}$$
$$Ax \leq b \tag{5.4}$$
$$x \in \{0;1\}^n \tag{5.5}$$

If we want to prove that the problem mentioned above is NP-Complete we have to show 2 things:

- The problem is NP.
- There is NP-Complete problem Π such that Π is reducible to the problem in polynomial time.

Suppose that the instance (A,b,c,k) has a feasible solution x^* with $x \geq k$. The encoding size of $x^* \in \{0;1\}$ is n. We can check in polynomial time that x^* is feasible solution by checking all the constraints and the value of the objective function, this give us an certificate that (A,b,c,k) is `yes-instance`.

To prove completeness of 0-1 Integer Programming we reduce satisfiability problem to 0-1 problem. Suppose to have an instance $\{X_1, X_2, \ldots, X_n, C_1, C_2, \ldots, C_m\}$ of satisfiability problem. For C_i we denote by C_i^- and C_i^+ the index sets positive and negative literals in C_i . We formulate the following 0-1 problem:

$$\max 0 \tag{5.6}$$

$$\sum_{j \in C_i^+} + \sum_{j \in C_i^-} (1 - x_j) \geq 1 \tag{5.7}$$

$$x_j \in \{0; 1\} \tag{5.8}$$

0-1 problem can be set up in polynomial time given the instance of satisfiability problem. Let us consider an instance C_i. If we have a truth assignment to the variables in C_i that satisfies C_i, then the corresponding setting of binary values to the variables in the 0-1 problem will satisfy the constraint (5.7) and vice versa, thus the 0-1 problem has a feasible solution if and only if C_i is satisfiable. Integers and Mixed Integers Programming are generalization of 0-1 Integer Programming; however, the proof that they are NP-Complete is harder to show. It can be found in [30].

5.2 Algorithms

As we have shown above, solving the integer linear programming (ILP) is a hard problem. In practice, ILP problems are solved with using relaxation to LP problems. This solution will be a vertex of the convex polytope consisting of all feasible points, which usually not integer and how you can see in Figure 5.2, the mere rounding of the result is suboptimal thus some more sophisticated algorithm has to be used. We show here the most important algorithm for solving ILP the Branch-and-Cut algorithm as well as the part forms that it is composted: Branch-and-Bound algorithm and Cutting-plane method.

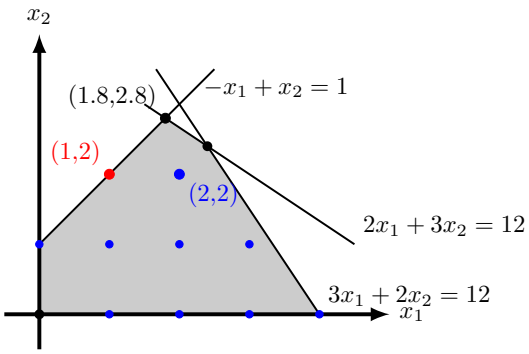


Figure 5.2. Example of LP relaxation routing sub-optimality. The result of LP relaxation (1.8,2.8) is rounded into (1,2) instead of optimal value of ILP (2,2).

5.2.1 Branch and Bound Algorithm

The Branch and Bound algorithm (B&B) is a general algorithm for finding optimal solutions of various optimization problems. The algorithm was proposed by A. H. Land and A. G. Doig [31] in 1960 and it is based on the `divide and conquer` idea.

The Algorithm consists of three steps. The first one is called branching in which problem is decomposed into a set of sub-problems. The second step is called bounding. In this step lower and upper bounds for given sub-problem are computed. The last step is called pruning during which some of sub-problems are discarded form the search.

Suppose that we would like to solve the following problem:

$$z = \max\{c^t x : x \in S\}.$$

This problem is hard to solve, but we can decompose it into set of sub-problems S. If $S = S_1 \cup S_2 \cup \ldots \cup S_n$ and $z^i = \max\{c^t x : x \in S_i\}$ for $i = 1, \ldots, n$ then $z = \max\{z^i\}, i = 1, \ldots, n$. where S_i can be recursively divided in enumeration tree. This is how branching part works; it is easy to see that number of leaf of enumeration tree can be enormous for big problems, so we must cut useless parts of tree. This can be done with using bound and pruning steps. Suppose that we have a lower bound and an upper bound for all sub-problems given by bound step $\underline{z_i} \leq z_i \leq \overline{z_i}, i \in 1, \ldots, n$. Then for $z = \max\{c^t x : x \in S\}$ it is true that $\max\{\underline{z_i}, i = 1, \ldots, n\} \leq z \leq \max\{\overline{z_i}, i = 1, \ldots, n\}$. We can prune the tree in three cases. The first one is pruning by optimality: if $\underline{z_i} = \overline{z_i}$, then an optimal value z_i is known, thus it is not necessary to subdivide it to smaller pieces. This is shown in Figure 5.4.

The second one is pruning by bound. If $z_i < \underline{z_j}, i \neq j$ then optimal value z_i never can be an optimal value of the whole problem, thus there is no reason to subdivide z_i. This is shown in Figure 5.3.

The last one is pruning by infeasibility. If $z_i = \emptyset$ there is either no need to subdivide z_i. In Figure 5.5 the case on which any of bound rules cannot be used is shown.

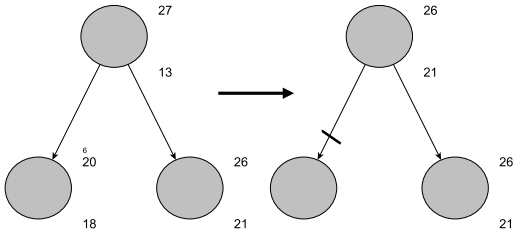


Figure 5.3. Example of pruning by bound.

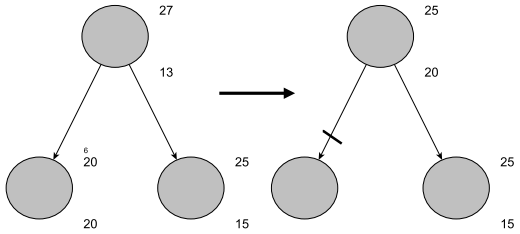


Figure 5.4. Example of pruning by optimality.

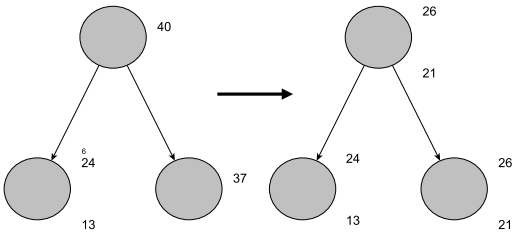


Figure 5.5. Example of case, where no pruning is possible.

The pseudocode of B&B algorithm is shown in Figure 5.6.

Algorithm 1: Brach and Bound Algorithm

```
Input: ILP problem
Output: optimal vector x*
/* L is List of problems to solve */
/* x* is current best integer solution*/
/* v* is lower bound*/
begin
    x ← null; v* ← −∞
    while L is not empty do
        Select and remove problem from L
        Solve LP relaxation of the problem
        if infeasible then
            continue
        else
            v ← Bound(Objective value)
            x ← Solution
        if v ≤ v* then
            continue
        if x is Integer then
            x* ← x; v* ← v
            continue
        Branch and add subproblems into L
```

Figure 5.6. B&B algorithm.

5.2.2 Gomory's Cutting-Plane Algorithm

Gomory's cutting-plane Algorithm proposed by Gomory [32] in 1958 is a general algorithm for solving ILP with using cutting plane. The idea of cutting plane method is following: consider that we want to solve a problem $\max\{c^t x : x \in Z^n\}$. We can relax this problem to LP problem $\max\{c^t x : x \in P\}$, where $P = \{x \in R+ : Ax \leq B\}$ is a rational polyhedron. Let's define $P_1 = conv(P \cap Z^n)$, thus P_1 is also rational polyhedron and $\max\{c^t x : x \in Z^n\}$ can be optimally solved as LP problem $\max\{c^t x : x \in P_1\}$ .It can be seen in Figure 5.7. The problem which must to be solved is to find a description of P_1.

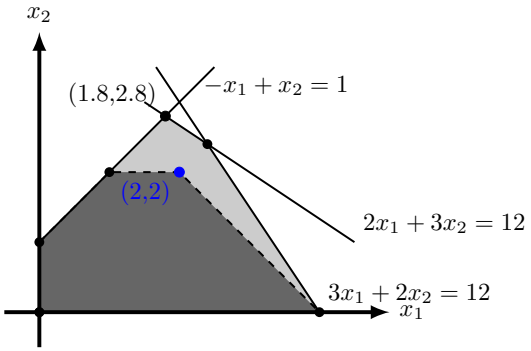


Figure 5.7. The light gray polyhedron is polyhedron of LP relaxation. The dark gray polyhedron is wanted polyhedron P_1 with optimal solution (2,2), which equals to optimal solution of our ILP problem.

One of the possibilities how to get a description of P_1 is used Gomory's cutting-plane algorithm. Consider that we have a following problem, which we want to solve:

$$\max c^t x \tag{5.9}$$
$$Ax = b \tag{5.10}$$
$$x \leq 0 \tag{5.11}$$
$$x \in Z^n \tag{5.12}$$

Where A is an integral $m \times n$-matrix and b is vector in Z^m. Suppose that we solve the relaxation

$$\max c^t x \tag{5.13}$$
$$Ax = b \tag{5.14}$$
$$x \leq 0 \tag{5.15}$$
$$x \in R^n \tag{5.16}$$

we get an optimal basis B and basis solution x^*. With using parametrization x^* by nonbasic variables:

$$x_B^* = A_B^{-1}b - A_B^{-1}A_N \cdot x_N^* = \overline{b} - \overline{A}_N \cdot x_N \cdot x_N^* \tag{5.17}$$
$$x_N^b = 0. \tag{5.18}$$

we can get the problem in basis representation

$$\max c_b^t x_b^* - \overline{c}_n^T x_n \tag{5.19}$$
$$x_B^+ = \overline{A}_N x_N \overline{b} \tag{5.20}$$
$$x \leq 0 \tag{5.21}$$
$$x \in R^n \tag{5.22}$$

In this step, two cases may occur x^* is integer and thus x^* is an optimal solution of integer problem, otherwise there exists a basic variable x_i^*, which is fractional e.i. $x_i^* = b_i \notin Z$. Then we derive an Gomory-Chvátal cutting-plane from equation (5.17). Let set $\overline{A} = (\overline{a}_{1k})$. Each feasible solution x of integer program satisfies (5.17), thus we have

$$x_i = b_i - \sum_{j \in N} \overline{a}_{ij} x_j \in Z \tag{5.23}$$
$$-\lfloor \overline{b}_i \rfloor - \lfloor a_{ij} \rfloor x_j \in Z \tag{5.24}$$

by subtracting (5.24) from (5.23) we get:

$$(\overline{b}_i - \lfloor \overline{b}_i \rfloor) - \sum_{j \in N} \{\overline{a}_{ij} - \lfloor \overline{a}_{ij} \rfloor\} \tag{5.25}$$

Because $0 < \overline{b}_i - \lfloor \overline{b}_i \rfloor < 1$ and $\sum_{j \in N} \{\overline{a}_{ij} - \lfloor \overline{a}_{ij} \rfloor\} \geq 0$ the following inequality is valid for P_1:

$$\sum_{j \in N} \{\overline{a}_{ij} - \lfloor \overline{a}_{ij} \rfloor\} \geq \overline{b}_i - \lfloor \overline{b}_i \rfloor \tag{5.26}$$

by rewriting this equation with slack variable s we get an Gomory-Chvátal cutting-plane:

$$\sum_{j \in N} \{\overline{a}_{ij} - \lfloor \overline{a}_{ij} \rfloor\} - s = \overline{b}_i - \lfloor \overline{b}_i \rfloor \tag{5.27}$$

In figure 5.8 is shown polyhedron(the dark gray one) after first cut is added.

If we use the lexicographic Simplex algorithm for solving the LPs and always derive the Gomory-Chvátal cut from first Simplex row in which the basic is fractional, and then algorithm terminates after a finite number of steps with an optimal integer solution, however, the algorithm can be slow to terminate. Presented version of the algorithm works only with 0-1 problems. Gomory also proposed cutting plane method for Mixed Integer Programming problems in 1960 [33] known as GMI `mixed-integer cut`. More about theory of valid inequalities for mixed integer linear sets such as lift-and-project cuts, Gomory's mixed integer cuts, mixed integer rounding cuts, split cuts and intersection cuts can be found in work of Cornu'ejols [34].

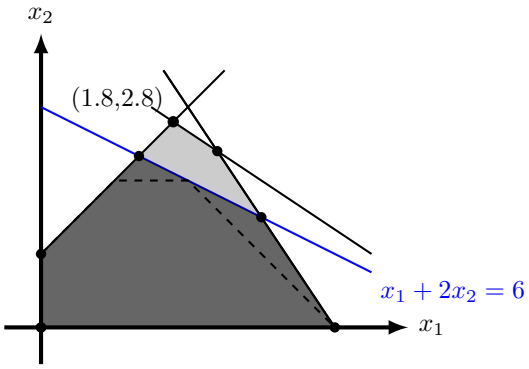


Figure 5.8. The light gray polyhedron is original polyhedron of LP relaxation. The dark gray polyhedron is given polyhedron after first cut by equation $x_1 + 2x_2 = 6$.

5.2.3 Branch and Cut Algorithm

Branch-and-Cut[1]) is the modification of Branch-and-Bound algorithm proposed by M. Padberg, G. Rinald [35] in 1991 for solving Symmetric Traveling Salesman Problems. The Branch-and-cut algorithm is combination of B&B algorithm and Cutting-plane methods, where linear programming relaxations of ILPs are tighten with using cutting planes. The pseudocode of B&C is shown in Figure 5.9

[1]) In some literature you can find also algorithm called Cut-and-Branch it is a special case of Branch-and-Cut algorithm where cutting plane method is used only once on first relaxation of MIP.

Algorithm 2: Brach and Cut Algorithm

```
Input: ILP problem
Output: optimal vector x*
/* L is List of problems to solve */
/* x* is current best integer solution*/
/* v* is lower bound*/
begin
    x ← null; v* ← −∞
    while L is not empty do
        Select and remove problem from L
        Solve LP relaxation of the problem
        if infeasible then
            continue
        else
            v ← Bound(Objective value)
            x ← Solution
        if v ≤ v* then
            continue
        if x is Integer then
            x* ← x; v* ← v
            continue
        if findCuts() ≠ ∅ then
            add cuts to LP and solve again
        Branch and add subproblems into L
```

Figure 5.9. B&C algorithm.

It should be noted that a third important algorithm for solving ILP exists. It is called branch-and-price algorithm, which is off course nothing else than dual problem of branch-and-cut algorithm were columns are generated instead of a rows.

5.3 Software

Many solvers for solving LP and ILP problems exist both commercial and free. Two of them can be considered as the main, namely CPLEX from IBM, which we use in our work, and GUROBI from Gurobi Optimization. Both of these solvers are representatives of commercial solvers. As a representative of free solver, we can mention GNU Linear Programming Kit (GLPK) developed by Andrew O. Makhorin of the Moscow Aviation Institute that was released in October 2000. The Gurobi is relatively young solver first published in 2009. Its name comes from the initial character of its authors Zongha Gu, Edward Rothberg and Roberta Bixby. It is interesting that Bixby is the founder of CPLEX and Gu and Rothberg led the development of CPLEX for almost 10 years. Already at the time of its release, it achieved the performance results as competitive CPLEX, and is now considered the fastest solver[1]).

[1]) By benchmark made by Gurobi, MIPLIB2010 benchmark set show showed a very equal results for MILP problems, where CPLEX is often faster.

5.3.1 CPLEX

As you may have already read, CPLEX is historically one of the most important solvers. It was developed by Robert E. Bixby and the first version was released in 1988 by CPLEX Optimization Inc. CPLEX offers an algorithm for solving ILP since 1991 (version 1.2). Now CPLEX is developed by IBM as IBM ILOG CPLEX Optimizer . And it can handle many types of problems: LP, MILP, (MI)QP, (MI)QCQP, (MI)SOCP (and some more).

Because CPLEX is one of the oldest solvers it can be used for showing improvement of scalability during the time. As you can see on performance graph 5.10 The scale on the left refers to the bars and shows version to version speed up and the scale on the right refers to the line and shows cumulative speed up. As you can see two bars in the chart stands outs it is comparing version 2.1 with version 3.0 with factor about 5.5, where the significant improvement corresponds to the implementation of the dual simplex algorithm and comparing version 6.0 with version 6.5 with factor about 10. This improvement was caused by implementing theoretical improvement as Branch-and-Cut. On the line you can see that overall speed up factor between version 1.2 and 11 is almost 100,000.

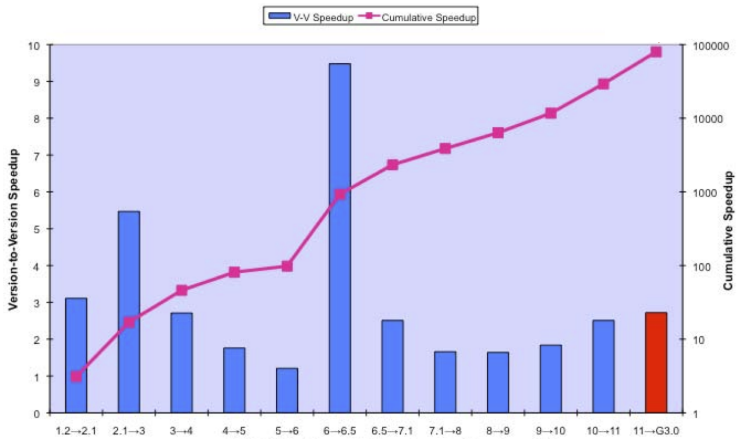


Figure 5.10. MIP performance improvements 1991-2010 [4] .

E. Bixby also shown [36] that in period of sixteen years from 1988 to 2004 LPs problems are solved in CPLEX 3300 times faster by algorithm improvement (thanks to machine improvement is solving of LP 1600x faster, which gives us total improvement factor of 5,280,00). One of the best advantages of CPLEX is a wide range of parameter by which, we can set properties of ILP algorithm as selection LP algorithm for solving starting an sub-problems LPs relaxation namely: primal simplex, dual simplex, network simplex, barrier method, sifting and concurrent. We can also chose a branch method namely: branch on variable with minimum infeasibility, branch on variable with maximum infeasibility, branch based on pseudo costs, strong branching, branch based on pseudo reduced costs CPLEX also allows an automatic mode, where it chose a branch method itself. We can also chose a node selecting strategy namely: depth-first search, best-bound search, best-estimate search or Alternative best-estimate search. We can also set many other parameters as node preprocessing, gap tolerance, frequency of heuristic etc. The whole list of parameters and more information about CPLEX settings can be seen in CPLEX User's guide [37].

5.4 Submodularity

Krause at al. [22] proposed near-optimal iterative algorithm for water sensor placement and rigorous proved that the produced solutions are guaranteed to be within $1 - \frac{1}{e}$ of the optimal solution, within computational time proportional to the number of nodes and scenarios considered. The algorithm had been proposed two years before by Uber at al. [38], however, without providing theoretical guarantees. The algorithm is based on maximization of submodular function. Nemhauser at al. [39] defined a submodular function in the following way: let N be a finite set and z be a real-valued function defined on the set of subsets on N that satisfies $z(S) + z(T) \leq z(S \cup T) + z(S \cap T)$ for all $S, T \in N$. Such function is called submodular. They also shown, that each function which satisfies

$$Z(S \cup \{k\}) - Z(S) \leq Z(R \cup \{k\}) - Z(R), R \subset S \subset N, K \in N - S$$

and

$$Z(S \cup \{k\}) - Z(S) \geq 0, S \subset N, k \in N - S$$

is (nondecreasing) submodular set function. In other words, submodular set function is such a function which satisfies that improvement of adding a new element k into smaller set R will be better than improvement which we get if we add an element k into larger set S. Moreover, rating of set S after adding new element k is never whore then rating of set S without added element k.

The algorithm works in the following way: we have a set of sensor positions S and a set of selected agent positions A. We start with empty an set $A = \emptyset$ and iteratively adding the required number of sensors in the such way, that in each step we add an sensor position which will decrease the expected penalty the most.

$$s_j = \arg\max_{s \in S} R(A \cup \{s\}) - R(A) A \leftarrow A \cup \{s\}$$

In our work we use a fact that even our problem is submodular where the proof can be seen in Section 7.4.2, thus we also use a iterative algorithm as one of ours speed-up techniques.

Chapter 6
Problem Formalization

Formally, the problem of oil pipeline monitoring using multiple UAVs can be seen as an optimization problem with multiple mobile agents optimizing a joint criterion function. In following sections, we abstract the environment into a graph, we explicitly define the movement of agents on the graph and we formulate the utility function to be optimized as a cost of potential damage to the environment. We use mathematical programming that allows us to capture constraints posed on the mobility of agents easily.

6.1 Time Discretization

As was mentioned above, agents are moving in discretized time horizon, below we will introduce two possibilities how to discretize given time horizon. Namely uniform time discretization where the given time horizon is divided into n uniform segments and non-uniform time discretization in which each of resulting time steps may represent a different time horizon. We have only one necessary condition: time horizon must be discretized in such way that time in time steps necessary to fly over an arbitrary edge by an arbitrary agent has to be integer. Our goal is to discretize given time horizon into a minimum number of time steps, because the number of time steps is the main bottleneck of provided algorithms.See Section 8.1.

6.1.1 Uniform Time Discretization

The uniform time discretization works in the following way: firstly, we compute the least common multiplier (LCM) of agents' speeds and then we multiply lengths of all edges by LCM value. This ensures fulfillment of the necessary condition, but number of time steps is not always minimal. Specifically we have now $t \cdot LCM$ time steps, where t is given horizon in seconds and each time step represent period of $\frac{1}{LCM}$ second(s), which is unnecessarily small number. Then we compute a great common divisor (GCD) of all possible time necessary for movement over arbitrary and by arbitrary agent. If we divide all these values by GCD, it will be still integers thus necessary condition is still met and also total number is reduced from $t \cdot LCM$ to $\frac{t \cdot LCM}{GCD}$ and each time step represents period of $\frac{GCD}{LCM}$ second(s). The pseudocode of this algorithm is shown in Figure 6.1.

Algorithm 3: Uniform Time Discretization

```
Input: Graph G, Set of Agent A, Time horizon T
Output: Timesteps
begin
    LCM ← computeLCM(A)
    T ← T · LCM
    for a ∈ A do
        for e ∈ E do
            TimeToFly ← e.length · LCM/a.speed
    GCD ← computeLCM(TimeToFly);
    T ← T/GCD ;
    for i = 0; i < T · LCM/GCD do
        Timesteps.add(GCD / LCM);
```

Figure 6.1. Uniform time discretization algorithm.

6.1.2 Non-uniform Time Discretization

Non-uniform time discretization is based on iterative graph search and storing all times in which some of nodes can be reached for each agent. This algorithm works in the following way: in the first step we make agents union according to their speed and bases, because is redundant to run search for two agents with same speed and base, because the result will be same. In the second step we iteratively go through a graph using depth first search, when agent is gradually flying into all neighboring nodes with time increased about value equals to time necessary to fly over edges which connect these two nodes. The agent is flying until the actual time is less or equal to given time horizon. For acceleration of the algorithm we also store information for each pair of agents' speed and node in which all times a node was visited by agent with given speed, if agent plunges into node n in time t and which has already been visited in time t by another agent with same speed, the plunging is stopped, because it will be redundant. The pseudocode of this algorithm is shown in Figure 6.2.

Algorithm 4: Non-Uniform Time Discretization

```
Input: Graph G, Set of Agent A, Time horizon T
Output: Timesteps
begin
    union ← computeUnion(A)
    for a ∈ A do
        DFS()
DFS(Agent a, Node n, Time t)
begin
    Timesteps.add(agent,node,time)
    for next ∈ getNeighbors(n) do
        DFS(a,next,time+getMovementTime(n,node)
```

Figure 6.2. Non-uniform time discretization algorithm.

6.2 Graph Representation

In our work, we consider two representations of graph coverage by agents, namely edge-oriented representation and node-oriented representation.

The primary motivation of a node-oriented representation is that the oil pipeline system is discretized with respect to agent radius, where positions of agent are represented by nodes; edges represent possible transitions between agents' locations over pipeline system. This discretization is very accurate, because it corresponds to physical restriction of UAVs' agent and concurrently the single segments of graph are relatively small, they are in range of tens to hundreds of meters. Accurate coverage of graph is also the biggest limitation of this representation, because for complete coverage of pipeline system with the size of tens to hundreds of kilometers, we need a graph with enormous number of nodes. The next disadvantage of this representation is that there is no way how to reduce number of nodes, by which we can improve scalability at the expense of accuracy of resulting graph, because sizes of edges are fixed.

The last disadvantage of node an oriented-representation is reflected by edge-oriented representation, where coverage of a pipeline is computed on edges and the agent radius is neglected, thus agent always covers whole edge irrespective of edge length and agent exact position on it. This representation allows us to improve scalability by graph simplification, where only restriction to maximal simplification is that graph must contains at least as many edges as many single pipelines is in pipeline system and every another edges improve accuracy of the representation.

It should be noted, that above we mentioned just a primary motivations and both of representations are interchangeable. In the case of edge-oriented graph with respect to agent's diameter, we will get a graph with unitary edges, with length equal to the diameter of the agent. And vice versa if we want a node oriented graph without respect to agent diameter the node in graph will be represent a center of coverage pipeline and edge length between two neighbors nodes will be equal to a sum of a half edge's length that singles nodes represent.

6.3 Movement

In each discretization described below we have a set of agents $K = \{{}^1a, {}^1a, \ldots, {}^ka\}$, which are moving over such oriented graph, that if an edge from node u to node v exists, then also an edge from node u to node v exists there. This is because agents can move in both directions on each edge, but we need to have information by which direction they are moving so we cannot use an undirected graph. There are also special loop edges in each node marked $\lambda(n)$, which are not directly considered for edge and they aren't used in time or area discretization. They represent waiting in node, which is used in model extension (see Section 7.2.3). Agents are moving over the graph in discretized time in `time steps`, where number of steps necessary for moving over edge e is set by variable σ_e; or σ_e^k if we consider non-uniform speed of agents.

Agents are always moving over edges of the given graph, so we establish a binary edge-entry variable ${}^ka_e^t$, which is set when the agent k enters the edge e in t^{th} time step. So when the move over the edge e takes more than one time step, the variable ${}^ka_e^t$ equals to one, but ${}^ka_e^{t+1}$ equals to zero.

In both representations, every agent covers exactly one edge or one node respectively. In the case of edge-oriented representation we establish variable ${}^kx_e^t$, which provides an information if edge e is covered by agent k in time t. The edge e is covered by agent

k in time t if and only if agent k is located somewhere over edge e i.e., if it enters the edge e in time range between $t - {}^k\sigma_e^t$ and t, where ${}^k\sigma_e^t$ provides an information, how many time steps agent k needs for flight over edge e. Thus if agent enters the edge in time $t = t - {}^k\sigma_e^t$, it is in time step t located on the last segment of given edge.

$$ {}^k x_e^t \geq \sum_{i=0}^{\sigma_e^t - 1} {}^k a_e^{t-i} \tag{6.1} $$

In the case of node-oriented representation, the coverage is similar: we establish variable ${}^k x_n^t$, which provides an information if node n is covered by agent k in times step t. The node is covered by an agent, if he is located on the second half of arbitrary edge leading into node n or on a first half of arbitrary edge leading from node n. If we express this by the variable ${}^k a_n^t$ we get that node is covered by agent k if he enters any arbitrary incoming edge in time range from $t - \sigma_e^k$ to $t - \frac{\sigma_e^k}{2}$ or if he enters to arbitrary outgoing edge in time range from $t - \frac{\sigma_e^k}{2}$ to t or if he is located on loop $\lambda(n)$.

$$ {}^k x_n^t \leq {}^k a_{\lambda(n)}^t + \sum_{\substack{e \in in(n) \\ e \neq \lambda(n)}} \sum_{i=\max\{t-\sigma_k+1;0\}}^{max\{t-\lfloor \sigma_k/2 \rfloor;0\}} {}^k r_e^i + \sum_{\substack{e \in out(n) \\ e \neq \lambda(n)}} \sum_{i=\max\{t-\lfloor \sigma_k/2 \rfloor+1;0\}}^{t} {}^k r_e^i \tag{6.2} $$

6.4 Area Discretization

In this section, we will focus on area discretization and introduce three different possibilities of discretization together with their advantages and disadvantages. All discretizations convert given pipeline system into to an oriented graph.

- The first one is `graph` discretization in which agents are moving strictly over pipelines of patrolled pipelines system. The example is shown in Figure 6.3.
- The second discretization is `hexagon convex hull` of given pipeline system with unitary length of edges. The example is shown in Figure 6.5.
- The last one is extended version of edge graph discretization. The example is shown in Figure 6.4.

All of these representations require as an input non-discretized graph of patrolled pipeline system.

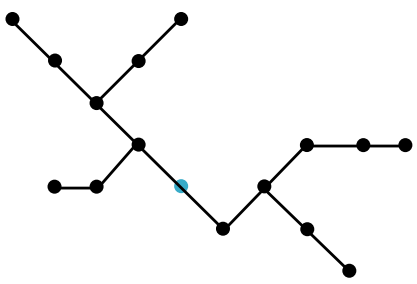


Figure 6.3. Example of graph discretization.

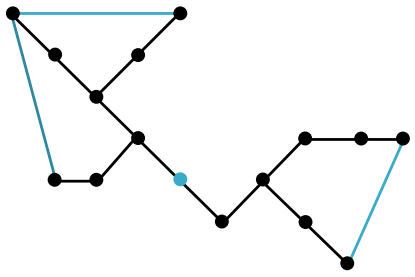


Figure 6.4. Example of extended graph discretization, where the blue lines represent added choice edges.

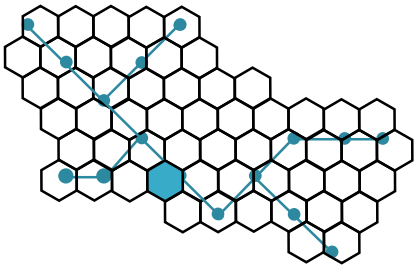


Figure 6.5. Example of convex hull discretization.

6.4.1 Graph Discretization

The first discretization is a graph model, where the main idea is to transform given graph $G(N, E)$ into graph $G_d(N_d, E_d)$, which has exactly n edges respective nodes $|E_d| = n$. Where the necessary condition is that $|Ed|$ is greater or equal to the number of single pipelines in the system, i.e., each pipeline in the system has to be represented by at least one edge(Figure 6.7).

$$|E_d| \geq \sum_{e \in E, in(e)=1} e + \sum_{e \in E, in(e)>2} e \tag{6.3}$$

Discretization works in the following way: Firstly, we remove all redundant nodes i.e., nodes that have exactly two neighbors where the angle between edges connecting neighbors with given node, is equals to 180 degree. After removing redundant nodes (Figure 6.6), we continue with respect to current size of graph $G'(E', N')$. If current number of edges $|E'|$ is bigger than the required number of them we remove another $|E'|\{n$ edges. Otherwise we add $n - |E'|$ edges. The edges are removed in the following way: we select all nodes, which have exactly two neighbors. Then we sort them in ascending order according to the sum of length of edges which connect a node with its neighbors. The first node is removed together with edges which lead into or from it and its two neighbors are connected together with new edge. it is repeated until $n - |E'|$ nodes (thus $n - |E'|$ edges) are not removed. The edges are added in the following way: we select the longest edge of E', add new node n into N' which is located at the middle of selected edge. Then we remove the selected edges and add a new edge from edges source and edge's target to added node n. It is repeated until $|E'| - n$ nodes is not added. The pseudocode of this algorithm is shown in Figure 6.8.

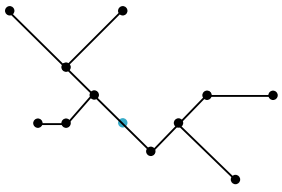


Figure 6.6. Example of graph discretization with maximal nodes reduction.

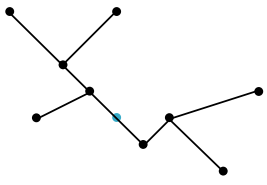


Figure 6.7. Example of graph discretization with shape deformation.

Algorithm 5: Graph Discretization

```
Input: Graph G, Number of edges n
Output: Discretized Graph G'
begin
    G' ← G
    RemoveNodeInLine(G')
    if n > |E'| then
        while i ≤ |E'| − n do
            node ← getNodeWithClosestNeighbor(N')
            G'.removeNode (node)
            connectAdjacentEdges (node)
            i++;
    else
        i ← 0;
        while i ≤ n − |E'| do
            edge ← getLongestEdge(E')
            n ← G.addNodeInCenterOfEdge(edge)
            G'.removeEdge(edge)
            G'.addEdge(getAncestor(edge), n)
            G'.addEdge(n, getSuccesor(edge))
            i++
```

Figure 6.8. Graph discretization algorithm

6.4.2 Hexagon Hull Discretization

The second representation is hexagonal convex hull of oil pipeline system. The hexagonal shape was chosen, because a hexagon inscribed around a circle with sensor radius can cover the entire area without any blind spots with relatively little overlap. (see Figure 6.9)

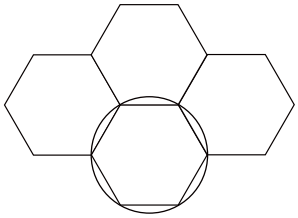


Figure 6.9. An example of hexagon approximation of circle radius.

The next advantage of hexagon is that all edges connecting single nodes representing center of nodes have a uniform length.

Discretization is performed as follows: firstly, we find bounding points (i.e., points of pipeline system with minimal and maximal longitude and latitude) of given system, then we cover this rectangle by hexagon hull. This hull is reduced to convex hull of system in such way that we create a full graph of system and remove hexagons that are not intersected by any edge. The last step is computation of utility function from

given graph to hull, where utility function of single hexagon equals to sum of edges' utility functions which are covered by hexagon. This is implemented in a following way: we compute positions of hexagon's lines and iteratively test an intersection with edges from given graph. In total, three cases may occur: the first is that we do not detect any intersection. The second one is that we detect one intersection, - one of edges ends points is inside of hexagon and we must detect which of ending points is located inside of hexagon. Then the potential error (PE) of hexagon is increased by $\frac{PE(e)\cdot|x_i,x_e|}{length(e)}$ where $PE(e)$ is value of potential error of edge e and $|x_i, x_e|$ is distance between ending point located inside hexagon and point of intersection. The third case that may occurs is that we find two intersection points in this case UF value of hexagon is increased by $\frac{PE(e)\cdot|x_1,x_2|}{length(e)}$ where x_1 and x_2 are intersection points. The pseudocode of this algorithm is shown in Figure 6.10.

Algorithm 6: Hexagon Hull Discretization

```
Input: Graph G
Output: Triangle Graph G'
begin
    bounds ← findBounds(G)
    G' ← createGrid(bounds)
    fullGraph ← createFullGraph(G)
    for n ∈ N' do
        for e ∈ fullGraph.edges do
            if !isIntersected(n, e) then
                remove(n)
    RecomputeUf()
```

Figure 6.10. Hexagon hull discretization algorithm.

6.4.3 Extended Edge Graph Discretization

Ondráček et al. [3] demonstrate, that deterioration of problem scalability, when the hexagon discretization is used, is not counterbalanced by improvement of the quality of solutions enough. In response to this finding we created an extended version of graph discretization mentioned above inspirited by Bošanský [40], who combines advantages of both algorithms: better scalability of graph discretization and possibility of movement out of oil pipeline to a limited extent. This discretization works in the following way: In the first step an extra choice, edges are added, where number of added edges is given by parameter as a percentage of the original edge.

Choice edges are selected in a such way, that number of connected pipes will be maximized and also that length of added choice edges were minimized, because with increasing length of choice edge the probability that movement over is will be more optimal than go back to the base over the same pipe decreases. It is implemented in the following way:

Firstly, we decompose a given graph to the set of single pipelines. This is done in such way, that to each of node we assign an index, which represents to which pipeline given node belongs. Thus, two nodes belong to the same pipeline if their indexes are equal.

After the graph is decomposed, we create a set of potential choice edges in such a way that we create an edge between each pair of nodes, where both from the pair belongs to another pipelines (i.e., has a different labels) and an edge connecting this two nodes does not exists there. We also need to store information how many times two different pipeline (i.e. sets of nodes with the same label) were connected. Then we sort potential edges by their length weighted by information how many times were pipelines, which given edge connects, connected before. The weighting of edges length increases the probability of connections of more different pipelines.

After this initialization, potential edges are sequentially added into the graph where after each addition, given edge is removed from a set of potential edges, information about pipelines connection is actualized and the set of potential edges is re-sorted. When choice edges are added, we created edge discretization of pipeline system with using edge graph discretization as was described above. The pseudocode of this algorithm is shown in Figure 6.11.

Algorithm 7: Extended Graph Discretization

```
Input: Graph G,Number of choice edges p, Number of edges n
Output: Discretized Graph G'
begin
    indexes ← createIndexes(N)
    for n ∈ N do
        if isLast(n) then
            potentionalNodes.add(n)
    for n ∈ potentionalNodes do
        for m ∈ potentionalNodes do
            if indexes(n)! = indexes(m) and !existsEdge(n, m) then
                potentionalEdges.add(n, m)
    sort (potentionalEdges)
    n ← 0
    E' ← E
    while n < |E| · p do
        e ← potentionalEdges[0]
        E'.add(e)
        potentionalEdges.remove(e)
        n++
        sort (potentionalEdges)
    G' ← discretizeGraph(G')
```

Figure 6.11. Extended Edge Graph discretization algorithm.

6.5 Utility Function

Given the problem objectives, we aim to minimize the time of all parts of the oil pipeline system being unobserved by some UAV, weighted by the importance of the area. Formally, we minimize weighted age of information (AoI) for all edges $e \in E$ and for time steps $t \in T$, we can compute a cost of potential damage caused from the last

visit. If an agent covers an edge in a given time step, AoI of that edge is set to zero. If no agent covers the edge, AoI is increased in each step by a predefined value τ, which indicates what time span is represented by the given time step. The age of information for an edge e at a time step t is defined recursively as:

$$AoI_e^t = \begin{cases} 0 & \text{if } \exists k, k \in K, {}^k x_e^t \geq 1 \\ AoI_e^{t-1} + \tau^t & \text{otherwise} \end{cases} \tag{6.4}$$

Equation (6.4) set the coverage of the edge e by an agent(Section 6.3). AoI is incremented by τ^t (time in seconds, which is represented by given time step) if no agent is covering the edge e. Using AoI and properties of the environment and pipeline, we can define cost of expected damage c_e^t for each edge at each time step:

$$c_e^t = AoI_e^t \cdot es_e \cdot l_e \cdot f_e \tag{6.5}$$

The cost is given by the age of information (capturing number of steps when the edge was not covered) multiplied by the value of vicinity environment sensitivity for the edge es_e, by length l_e of the edge e and by failure rate f_e. f_e and es_e represent degree of freedom in our model and it can be replaced by a parameter supplied by subject matter experts. (see Section 3.2).

For given system, es_s, l_e and f_e are fixed. We can thus replace the multiplication term by constant $C_e^t = es_e \cdot l_e \cdot f_e$, reformulating the cost to:

$$c_e^t = AoI_e^t \cdot C_e \tag{6.6}$$

By combining the equation (6.4) and (6.6) we get:

$$c_e^t = \begin{cases} 0 & \text{if } \exists k, k \in K, {}^k x_e^t \geq 1 \\ c_e^{t-1} + C_e \cdot \tau^t & \text{otherwise} \end{cases} \tag{6.7}$$

And we can express the utility for the system and time horizon T as:

$$U = \sum_{t \in T} \sum_{e \in E} c_e^t \tag{6.8}$$

The problem can be then specified as a problem of design of a constrained movement of agents such that the utility defined in equation (6.8) is minimized.

Computation of utility function is done on edges, but in the case of node-oriented graph, we need to recompute the utility function for nodes. It is done in the following way: the value of utility function for node n is equals to sum of utilities on incoming and outgoing edges, divided by two.

$$c_n^t = \frac{1}{2} \sum c_e^t, e \in out(n) \cup in(n) \tag{6.9}$$

If we want to recompute to edges back, it can be done in a such way that edge's utility function sum of source node's cost divided by number of incoming and outgoing edges of this node and target node's cost divided by number of incoming and outgoing edges of this node.

$$c_e^t = \frac{c_n^t}{(|in(n)| + |out(n)|)} + \frac{c_m^t}{(|in(m)| + |out(m)|)} \tag{6.10}$$

Chapter 7
Algorithms

In this chapter, we introduce our developed algorithms. Firstly, we explain the problem of optimal bases placement. Then we introduce basic NOOPP algorithm together with set of its extension as well as EOPP algorithm. In the end of this chapter, we introduce three speed-up techniques which we developed.

7.1 Bases Placement and Distribution of UAVs over Bases

First problem, which we introduce in this chapter, is the problem of bases placement and distribution of agents over them. The bases placement problem is not a primary problem of infrastructure monitoring, however, it should has an important influence on the quality of the solution, because the optimal solution of a problem with inappropriately placed bases can be a lot of worse than the optimal solution of a problem with optimally placed bases. Placement of bases can be of course solved as a part of model, which is described in section 7.2.8, then the given solution is optimal, however, time complexity of model is growing rapidly after this extension. This is because state space of agent movement contains whole space in each time step unlike in the problem with fixed bases.

More concretely without loss of generality, we can assume a case when the given graph representing monitored area has edges with unitary length (for example hexagon hull 6.4.2 satisfies this property) and we also have one agent, which satisfies an constraint, that fly over an arbitrary edge takes exactly one time step and we are solving that problem over τ time steps. If we have fixed base b, the state space of agent movement in time step $t = 1$, let's call it σ_1, is restricted to edges landing from the base. In formal terms:

$$e \in \sigma_1 \iff e \in out(B) \tag{7.1}$$

In time step $t = 2$ the state space contains all edges from σ_1 and also all edges which are leading from nodes to which some edges from σ_1 are leading i.e,

$$e \in \sigma_2 \iff (e \in \sigma_t \vee (\exists n \in N, f \in \sigma_1 e \in out(n) \wedge f \in in(n)$$

The state space is gradually extended in according to the previous way until time $t = \theta$ [1]),which represents a time step from which the state space is gradually reduced, in formal terms

$$e \in \sigma_t \iff (e \in \sigma_{t-1} \vee (\exists n \in N, f \in \sigma_{t-1} e \in out(n) \wedge f \in in(n))), \quad \forall t < \theta \in T$$

so of course, applies

$$|\sigma_{t-1}| < |\sigma_t|, \forall t \leq \theta \in T$$

[1]) It should be noted, that size of σ_t may not increases until to $t = \sigma$, but may has a same size for several step, because already contains whole graph and therefore it may not be greater anymore. Only thing that can be guaranteed, is that size of σ_{t-1} is nondecreasing until $t < \theta$.

The reduction of the state space is caused by restriction that each agent lands on its base in time step $t = \tau$ i.e., in time $t = \tau - 1$ has to be located on some edges leading into base, so

$$e \in \sigma_{\tau-1} \iff e \in in(B)$$

In order to previous restriction in time $\tau - 2$ state space contains all edges from state space in time $\tau - 1$ and also all edges leading in node from which leading some edge from σ_{t-1}. In formal terms

$$e \in \sigma_{\tau-2} \iff (e \in \sigma_{\tau-1} \vee (\exists n \in N, f \in \sigma_1 e \in in(n) \wedge f \in out(n)))$$

and generally

$$e \in \sigma_{t-1} \iff (e \in \sigma_t \vee (\exists n \in N, f \in \sigma_t e \in in(n) \wedge f \in out(n))), \quad \forall t > \theta.$$

As we shown above, if we are solving single agent distribution over some graph with fixated bases, the state space of movement in each time step t is some subset of edges $\sigma_t \subset E$, when especially in time step near to 0 or T is given subset very small. But if we solve the problem of bases placement together with main monitoring problem the state space of movement will be in each time step t equals to edge set $\sigma_t = E$. It may be simply proved, we can set for each node variable σ_t^n that state space of movement in time t if the bases is in node n. As was showed above σ_t^n contain always at least all edges leading from node n. Because σ_t is union of all σ_t^n

$$(\sigma_t = \sigma_t^{n_1} \cup \sigma_t^{n_2} \cup \ldots \cup \sigma_t^{n_N})$$

σ_t contain in each step outgoing edges from all nodes which equal to E ($\sigma_t = E$).

From this reason, it may to be preferable to decompose the problem into separate bases location problem and single UAS distribution problem. The problem of bases location is classical example of so-called maximum coverage problem, which can be formalized with using MIL program in the following way: we have an set of sets $S = S_1, S_2, \ldots, S_m$ and number k. The objective is chose a subset of maximally k sets $S' \subset S, |S'| \leq k$ so that number of coverage elements $|\bigcup_{S_i \in S'} S_i|$ will be maximal. In our case we have set containing as many sets as many nodes has the given graph over which we optimize $|S| = |V|$, and single set represents an location of bases in single node of the graph. Each set contains all nodes, which are distanced from given node maximally $\frac{1}{2}$ flying range of UAV with maximal flying range, in formal terms $n_i \in S_j$iff $|n_i, n_j| \leq \max_{a \in A} \frac{1}{2} a_{range}$.

Model 7.1.

$$max \sum_{j \in E} y_j \tag{7.2}$$

$$\sum_{e_j \in S_i} x_i \geq y_j \qquad \forall j \in E \tag{7.3}$$

$$\sum x_i \leq k \qquad \forall i \in S \tag{7.4}$$

$$y_j \in (0, 1) \qquad \forall j \in E \tag{7.5}$$

$$x_i \in \{0, 1\} \qquad \forall i \in S \tag{7.6}$$

This basic problem can be extended by several known way. The best known is weighted maximal coverage problem, when each element $e \in E$ has assigned the value

$w(e)$ and the objective is not now chose the subset from set so that the coverage of elements were maximal, but that a sum of value of coverage elements will maximal. In our case $w(e)$ corresponds to rating by utility 6.5 function of given node.

Model 7.2.

$$max \sum_{j \in E} w(e_j) \cdot y_j \tag{7.7}$$

$$\sum_{e_j \in S_i} x_i \geq y_j \qquad \forall j \in E \tag{7.8}$$

$$\sum x_i \leq k \qquad \forall i \in S \tag{7.9}$$

$$y_j \in (0,1) \qquad \forall j \in E \tag{7.10}$$

$$x_i \in \{0,1\} \qquad \forall i \in S \tag{7.11}$$

We cannot use a coverage problem in the form as it was described above, because we also want to distribute the agent. It is done in the following way: we replace x_i variable for ${}^k b_i$, which is equal to one if node i is a base for agent k. It is easy to see that because each of agent must has a base the number of selected bases will be equal to number of agent. Thus, we must to reformulate a restriction to number of bases where we now don't want to select maximally B bases, but maximally B different basis

$$\sum_{n \in N} \min\{\sum_{k \in K} {}^k b_n; 1\} \leq B$$

. The problem of bases placement and distribution of agents over them can be solved with using following model:

Model 7.3.

$$max \sum_{e_j \in E} w(e_j) \cdot y_j \tag{7.12}$$

$$\sum_{e_j \in S_i} x_i \geq y_j \qquad \forall j \in E \tag{7.13}$$

$$\sum_{n \in N} \min\{\sum_{k \in K} {}^k b_n; 1\} \leq B \tag{7.14}$$

$$y_j \in (0,1) \qquad \forall j \in E \tag{7.15}$$

$${}^k b_i \in \{0,1\} \qquad \forall i \in S \tag{7.16}$$

It should be noted, that disadvantage of separated base location problem is, that we loss the solution optimality. It is because the algorithm is only able to determine a set with a maximal cost coverage of given area, however, it is no longer able to decided, if some of agents is not able to prevent more damage by monitoring in, from general point of view, less preferred area against monitoring more preferred one. This can occurs for example by location of nodes in graph, when the total sum of nodes values in set is bigger, but sum of nodes that can be covered by agent is lower.

7.2 Node Oriented Optimal Pipeline Patrolling Algorithm (NOOPP)

In this and the next section we will focus on deterministic optimal models, thats distribute agents over finite time horizon τ, whose discretization was mentioned in the

section 6.1. Concreatly in this section we focuse on model which optimizes over graph's nodes, which represent given area, more about area discretization can find in the section 6.4, where each node represents location of agent in time. This section is written in the following way. Firstly we introduce a basic multi-agent model which is simplified and does not take into account all abilities of agents and constraints posed by the real world. This model we latery extend by using a set of extension and thus we remove some of model restriction. aAt the end of the section is described a posibility how to integerate basic location problem into given model.

7.2.1 Basic NOOPP

As has been said the basic model has several restriction that have to be fulfilled. First restriction is, that all agents have one common base from which they lift off in time step $t = 0$ and to which they land to in time step $t = \tau$. Next restriction is, that agents have not any restriction on maximum flight distance, e.i. their flight distance is just restricted by given time horizon through which is area monitored. Another restriction is that all agents have to have an unitary speed and in the case of when we understand the node as is described in section 6.4 is next restriction that all agents have a sensor with unitery radius, in the case that we understand the node as is described in alternative version of node discretization,this condition is not required. The last restriction of this model is, that does not permit wainting in node for several time steps, This can lead to sub-optimal solutions as is mentoined in section 7.2.4

Model 7.4.

$$min \sum_{t\in T}\sum_{n\in N} c_n^t \tag{7.17}$$

$$c_n^t \geq c_n^{t-1} + C^n \cdot \tau^t - M \cdot \sum_{k\in K} {}^k x_n^t \qquad \forall n \in N, \forall t \in [1,T] \tag{7.18}$$

$$c_n^t \geq 0 \qquad \forall n \in N, \forall t \in T \tag{7.19}$$

$$c_n^0 = 0 \qquad \forall n \in N \tag{7.20}$$

$$\sum_{f\in out(n)} {}^k a_f^t \leq \sum_{e\in in(n)} {}^k a_e^{max\{t-\sigma_e;0\}} \qquad \forall k \in K, \forall n \in N, \forall t \in [1,T] \tag{7.21}$$

$$\sum_{e\in E} {}^k a_e^t \leq 1 \qquad \forall k \in K, \forall t \in T \tag{7.22}$$

$$\sum_{e\in E} {}^k a_e^0 = 0 \qquad \forall k \in K \tag{7.23}$$

$$\sum_{e\in out(B)} {}^k a_e^1 = 1 \qquad \forall k \in K \tag{7.24}$$

$$\sum_{e\in in(B)} {}^k a_e^{T-\sigma_e} = 1 \qquad \forall k \in K \tag{7.25}$$

$$\begin{aligned} {}^k x_n^t \leq & \sum_{e\in in(n)} \sum_{i=max\{t-\sigma_e+1;0\}}^{max\{t-\lfloor\sigma_e/2\rfloor;0\}} {}^k a_e^i + \\ & + \sum_{e\in out(n)} \sum_{i=max\{t-\lfloor\sigma_e/2\rfloor+1;0\}}^{t} {}^k a_e^i \end{aligned} \qquad \forall k \in K, \forall n \in N, \forall t \in [1,T] \tag{7.26}$$

$${}^k a_e^t \in \{0;1\} \qquad \forall e \in E, \forall k \in K, \forall t \in T \tag{7.27}$$

$${}^k x_n^t \in R \qquad \forall k \in K, \forall n \in N, \forall t \in T \tag{7.28}$$

$$c_n^t \in R \qquad \forall n \in N, \forall t \in T (7.29)$$

Equation (7.17) is the criterion minimizing the utility function over all time steps and nodes of given graph. Equations from (7.18) to (7.20) describe the advancement of cost on nodes in time (see section 6.5).Equation (7.18) defines how a value of node n is changing in time depending on agents' movement. Equation (7.21) guarantees continuity of agents' movemet: it defines that number of agents entry on edge leading from node n in time t is less or equal to number of agents entry on edge leading to node n in time $t - \sigma_e$, where σ_e is time in time steps, that agent needs to flight over edge e. Equation (7.22) defines that the k-agent is located on at most one edge in one time step. Equation (7.23) inicializates the positions of the agents in time step $t = 0$ in such way, that agents are not located on any edge. Equation (7.24) inicializates the position of the agents in time step $t = 1$ so that each agent has to be located on some edge leading from the base. Equation (7.25) defines that each agent has to entry on some edge leading to the base in time step $T - \sigma_e$,which guarantee that in time step T each agent will be located in the base. Equation (7.26) expresses the conversion of location on edges to coverage of nodes (see section 6.3)

The route of agents can be reconstruct from ${}^k a_e^t$ variables. One of ${}^k a_e^t$ is equal to 1 in each time step thus sequence of ${}^k a_e^t$ that equal to one for $t = \{1\cdot, T\}$ is optimal route for agent a.

7.2.2 Multiple Bases Extension

The First restriction of the basic model is that there is just a single base in the graph. This improvement was made by adding the information for each agent k in which base ${}^k B$ from set of bases ${}^k B \in B$ it takes off and lands on. A single base can be used by multiple agents. For incorporating this extension into the model, we added base index in equations (7.24) and (7.25):

$$\sum_{e \in out({}^k B)} {}^k a_e^1 = 1 \qquad \forall k \in K (7.30)$$

$$\sum_{e \in in({}^k B)} {}^k a_e^{T-\sigma_e} = 1 \qquad \forall k \in K (7.31)$$

7.2.3 Endurance of Agents

One of the real-world UAV restrictions is that each agent has a maximum flying endurance representing how many time steps an agent can fly without recharging. To incorporate this restriction, we define maximum endurance ${}^k D$ for each agent k and we add a variable ${}^k d_t$ for all agents and time steps $t \in T$ which represents how many time steps the agent have been away from its base, i.e., the flight time of the agent. These properties are reflected in equations (7.32). The agent's endurance is computed in the following way. If the agent is not located in its base at a current time step, then the flight time is increased by 1 according to equation (7.34). Otherwise, the result of the equation (7.34) is negative and the flight time of the agent is set to 0 according to the stronger restriction (7.33) (we assume that the agent needs just one step for the recharging in its base).

$${}^k d^t \leq {}^k D \qquad \forall k \in K, \forall t \in T (7.32)$$

$$
\begin{aligned}
{}^{k}d^{t} &\geq 0 && \forall k \in K, \forall t \in T \quad (7.33)\\
{}^{k}d^{t} &\geq {}^{k}d^{t-1} + \tau^{t} - M \cdot {}^{k}x_{B}^{t} && \forall k \in k, \forall t \in [1,T] \quad (7.34)\\
{}^{k}d^{t} &\in \{0; {}^{k}D\} && \forall k \in K, \forall t \in T \quad (7.35)
\end{aligned}
$$

7.2.4 Waiting in a Node

An additional restriction of the model is that the agents cannot stay in one node for more than one time step which can lead to a suboptimal solution. This restriction is solved by adding loops into all nodes, where looping in the node, marked as$\lambda(n)$, is equivalent to waiting in the node. Loops in all nodes also solved an another restriction where the agents have to take off from its base in time t = 0, i.e., by looping in the base, they can take off in an arbitrary step. Notice that thanks to equations (7.33) and (7.34), even after looping in the base, the endurance is still 0 when taking off from the base in an arbitrary step. The possibility of waiting in a node have to be considered in equations (7.21) and (7.26).

$$
\sum_{f \in out(n)} {}^{k}a_{f}^{t} \leq {}^{k}a_{\lambda(n)}^{t-1} + \sum_{\substack{e \in in(n)\\ e \neq \lambda(n)}} {}^{k}a_{e}^{max\{t-\sigma_e;0\}} \qquad \forall k \in K, \forall n \in N, \forall t \in [1,T] \quad (7.36)
$$

$$
\begin{aligned}
{}^{k}x_{n}^{t} \leq {}^{k}a_{\lambda(n)}^{t} + \sum_{\substack{e \in in(n)\\ e \neq \lambda(n)}} \sum_{i=max\{t-\sigma_e+1;0\}}^{max\{t-\lfloor \sigma_e/2 \rfloor;0\}} {}^{k}a_{e}^{i} +\\
+ \sum_{\substack{e \in out(n)\\ e \neq \lambda(n)}} \sum_{i=max\{t-\lfloor \sigma_e/2 \rfloor+1;0\}}^{t} {}^{k}a_{e}^{i} \qquad \forall n \in N, \forall k \in K, \forall t \in [1,T] \quad (7.37)
\end{aligned}
$$

7.2.5 Various Speed Of Agents

An additional restriction of the above-mentioned model is that works with homogeneous set of agent, where each of agents has unitary speed and hasn't any endurance restriction. The problem of endurance restriction were solved in previous section. For solving the homogeneous speed, we must to reformulate σ_e variable, which says how many time steps need arbitrary agent to flight over edge e to σ_e^k which defines number of necessary time steps for a corneae agent k. This change was made in equation (7.21) (7.25) and (7.26). It should be recalled that various speed of agents is not difficult for formulating, but has a big impact to model scalability, especially with unsuitable speed ratio (see section 6.4).

$$
\sum_{f \in out(n)} {}^{k}a_{f}^{t} \leq \sum_{e \in in(n)} {}^{k}a_{e}^{max\{t-\sigma_e^k;0\}} \qquad \forall k \in K, \forall n \in N, \forall t \in [1,T] \quad (7.38)
$$

$$
\sum_{e \in in(B)} {}^{k}r_{e}^{T-\sigma_e^k} = 1 \qquad \forall k \in K \quad (7.39)
$$

$$
\begin{aligned}
{}^{k}x_{n}^{t} \leq \sum_{e \in in(n)} \sum_{i=max\{t-\sigma_e^k+1;0\}}^{max\{t-\lfloor \sigma_e/2 \rfloor;0\}} {}^{k}a_{e}^{i} +\\
+ \sum_{e \in out(n)} \sum_{i=max\{t-\lfloor \sigma_e^k/2 \rfloor+1;0\}}^{t} {}^{k}a_{e}^{i} \qquad \forall k \in K, \forall n \in N, \forall t \in [1,T] \quad (7.40)
\end{aligned}
$$

7.2.6 N-step Recharging

The last solved restriction of the model is that we assume that the agent needs just one step for the recharging in its base, which is in conflict with reality. The solution of this restriction allows partial charge and it is formulated in the following way: We define ${}^kR$ constant for each agent $k \in K$ which indicates by how many time step is endurance of agent k recharged for a single time step. Then we change the equation (7.34) so that in each time step, when agent looping over base,which corresponds to landing at the base, is his endurance decrease by ${}^kR$ If the result of this new equation was negative, it would to be set to zerro acooring to stronger restriciton at equation (7.33).

$$ {}^kd^t \geq {}^kd^{t-1} + \tau^t - \tau^t \cdot {}^kR \cdot \lambda({}^kB)^t \quad \forall k \in k, \forall t \in [1,T] (7.41)$$

7.2.7 Complete NOOPP

Model 7.5.

$$min \sum_{t\in T}\sum_{n\in N} c_n^t \tag{7.42}$$

$$c_n^t \geq c_n^{t-1} + C^n \cdot \tau^t - M \cdot \sum_{k\in K} {}^kx_n^t \qquad \forall n \in N, \forall t \in [1,T] (7.43)$$

$$c_n^t \geq 0 \qquad \forall n \in N, \forall t \in T (7.44)$$

$$c_n^0 = 0 \qquad \forall n \in N (7.45)$$

$$\sum_{f\in out(n)} {}^ka_f^t \leq {}^ka_{\lambda(n)}^{t-1} + \sum_{\substack{e\in in(n)\\ e\neq\lambda(n)}} {}^ka_e^{max\{t-\sigma_e^k;0\}} \qquad \forall k \in K, \forall n \in N, \forall t \in [1,T] (7.46)$$

$$\sum_{e\in E} {}^ka_e^t \leq 1 \qquad \forall k \in K, \forall t \in T (7.47)$$

$$\sum_{e\in E} {}^ka_e^0 = 0 \qquad \forall k \in K (7.48)$$

$$\sum_{e\in out({}^kB)} {}^ka_e^1 = 1 \qquad \forall k \in K (7.49)$$

$$\sum_{e\in in({}^kB)} {}^ka_e^{T-\sigma_e^k} = 1 \qquad \forall k \in K (7.50)$$

$$ {}^kx_n^t \leq {}^ka_{\lambda(n)}^t + \sum_{\substack{e\in in(n)\\ e\neq\lambda(n)}} \sum_{i=max\{t-\sigma_e^k+1;0\}}^{max\{t-\lfloor\sigma_e^k/2\rfloor;0\}} {}^ka_e^i + $$

$$ + \sum_{\substack{e\in out(n)\\ e\neq\lambda(n)}} \sum_{i=max\{t-\lfloor\sigma_e^k/2\rfloor+1;0\}}^{t} {}^ka_e^i \qquad \forall k \in K, \forall n \in N, \forall t \in [1,T] (7.51)$$

$$ {}^kd^t \leq {}^kD \qquad \forall k \in K, \forall t \in T (7.52)$$

$$ {}^kd^t \geq 0 \qquad \forall k \in K, \forall t \in T (7.53)$$

$$ {}^kd^t \geq {}^kd^{t-1} + \tau^t - \tau^t \cdot {}^kR \cdot \lambda({}^kB)^t \qquad \forall k \in k, \forall t \in [1,T] (7.54)$$

$$ {}^ka_e^t \in \{0;1\} \qquad \forall e \in E, \forall k \in K, \forall t \in T (7.55)$$

$$ {}^kx_n^t \in R \qquad \forall k \in K, \forall n \in N, \forall t \in T (7.56)$$

$$c_n^t \in R \qquad \forall n \in N, \forall t \in T (7.57)$$

$$ {}^kd^t \in \{0; {}^kD\} \qquad \forall k \in K, \forall t \in T (7.58)$$

7.2.8 Bases Placement Extension

Problem of bases location and agents distribution over them can be directly integrated into program, with the fact that computational complexity is significantly increased as is described in ??, however the solution will be optimal in this case. This problem can be integrated into program in the following way: In the program as is mentioned in previouse sections is base(or bases in extended verison) established as constant, so we replace constant ${}^kB$ for variable ${}^kb_n$,which indicates if a given node is a base of agent k. i.e, ${}^kb_n = 1$ if and only if agent k is in time t=1 located on some edges lading from node n, this fact is written in equation (7.59). We introduce a constat B, which state the maximal number of different bases. The restriction is maked in the following way:

$$\sum_{n \in N} \min\{\sum_{k \in K} {}^kb_n; 1\} \leq B$$

This equation can be linearized with using binary variable m_n. The m_n varible is set to zerro if no agent has his base in node n, otherwise is set to 1, thanks to restriction that $m_n \geq \epsilon \cdot \sum$, where ϵ is some small value no grater than $\frac{1}{|K|}$.

It is also necessary to adapt equation (7.30) (7.31) and (7.41). All of the equations are adapted in the same way, so we generalize this equations for $\forall n \in N$ and in case of equations (7.30) and (7.31) multiply the right side of equations by ${}^kb_n$ variable. In case of equation (7.41) we multiply $\lambda(n)$ on the right side of equation by ${}^kb_n$.

$$\sum_{e \in out({}^kB)} {}^ka_e^1 \leq {}^kb_n \qquad \forall k \in K, \forall n \in N (7.59)$$

$$m_n \geq \epsilon \cdot \sum_{k \in K} {}^kb_n \qquad \forall n \in N (7.60)$$

$$\sum_{n \in N} m_n \leq B \qquad (7.61)$$

$$m_n \geq 0 \quad m_n \leq 1 \qquad \forall n \in N (7.62)$$

$$\sum_{e \in out(n)} {}^ka_e^1 = {}^kb_n \qquad \forall k \in K, \forall n \in N (7.63)$$

$$\sum_{e \in in(n)} {}^ka_e^{T-\sigma_e^k} = {}^kb_n \qquad \forall k \in K, \forall n \in N (7.64)$$

$${}^kd^{t-1} + \tau^t - \tau^t \cdot {}^kR \cdot \lambda(n)^t \cdot {}^kb_n \leq {}^kd^t \quad \forall k \in k, \forall n \in N, \forall t \in [1, T] (7.65)$$

$${}^kb^n \in \{0; 1\} \qquad \forall k \in K, \forall n \in N (7.66)$$

$$m_n \in \{0; 1\} \qquad \forall n \in N (7.67)$$

7.3 Edge Oriented Optimal Pipeline Patrolling Algorithm (EOOPP)

In this section we introduce a mix integer linear program that optimize utility function on edges instead on nodes as above mentioned model does. According to fact, that booth models are ideologically close, we wont introduce a model as a basic model and set of it's extensions, but we directly show completed model and we will focus on differences between models.

Model 7.6.

$$min \sum_{t\in T}\sum_{e\in E} c_e^t \tag{7.68}$$

$$c_e^{t-i} \geq c_e^{t-1} + C_e \cdot \tau^t - M \cdot \sum_{k\in K}\sum_{i=0}^{\sigma_e^k-1} \left({}^k a_e^{t-i} + {}^k a_{\overline{e}}^{t-i}\right) \forall e \in E, \forall t \in [2,T] \tag{7.69}$$

$$c_e^t \geq 0 \qquad \forall e \in E, \forall t \in T \tag{7.70}$$

$$c_e^0 = 0 \qquad \forall e \in E \tag{7.71}$$

$$\sum_{f\in out(n)} {}^k a_f^t \leq {}^k a_{\lambda(n)}^{t-1} + \sum_{\substack{e\in in(n)\\ e\neq\lambda(n)}} {}^k a_e^{max\{t-\sigma_e^k;0\}} \qquad \forall e \in E, \forall k \in K, \forall t \in [1,T] \tag{7.72}$$

$$\sum_{e\in E} {}^k a_e^t \leq 1 \qquad \forall k \in K, \forall t \in T \tag{7.73}$$

$$\sum_{e\in E} {}^k a_e^0 = 0 \qquad \forall k \in K \tag{7.74}$$

$$\sum_{e\in out({}^k B)} {}^k a_e^1 = 1 \qquad \forall k \in K \tag{7.75}$$

$$\sum_{e\in in({}^k B)} {}^k a_e^{T-\sigma_e^k} = 1 \qquad \forall k \in K \tag{7.76}$$

$${}^k d^t \leq {}^k D \qquad \forall k \in K, \forall t \in T \tag{7.77}$$

$${}^k d^t \geq 0 \qquad \forall k \in K, \forall t \in T \tag{7.78}$$

$${}^k d^t \geq {}^k d^{t-1} + \tau^t - \tau^t \cdot {}^k R \cdot \lambda({}^k B)^t \qquad \forall k \in k, \forall t \in [1,T] \tag{7.79}$$

$${}^k a_e^t \in \{0;1\} \qquad \forall e \in E, \forall k \in K, \forall t \in T \tag{7.80}$$

$$c_e^t \in R \qquad \forall n \in N, \forall t \in T \tag{7.81}$$

$${}^k d^t \in \{0; {}^k D\} \qquad \forall k \in K, \forall t \in T \tag{7.82}$$

As you can see the first difference between two representation is different minimizing criterion function, which is now computed over edges, so equations from (7.69) to (7.71) and (7.81) are no longer defines on nodes but naturally on edges.

In (7.69) variable ${}^k x_n^t$ was replaced by $\sum_{i=0}^{\sigma_e^k-1} {}^k a_e^t + {}^k a_{\overline{e}}^t$ where ${}^k a_e^t$ denote that agent k entered on edge e in time t and ${}^k a_{\overline{e}}^t$ denote that agent k entered on edge inversely orientated to e in time t. This is done because graph over which we optimize is oriented graph, where for each edge exits inversely orientated one. as was described in section 6.4 this was done for reasons, that we need to be able decided by which direction is agent moving, but in real area are booth of these edges the same undirected edge. The agent is located somwhere on edge e in time t if he entered on edge in time $t' > t - \sigma_e^k$, this is guarantee, because any of agents has not ability to change the direction of his movement during its movement over edge. Thus we get an information whether an agent is located on edge in time te by summarazing $\sigma_e^k - 1 {}^k a_e^t$ and ${}^k a_{\overline{e}}^t$ over such time steps, that it was located on edge e.In summary cost of edge e is set to zero in time t, if some of agent is located somewher over it or its inversely edge in time t.

The another difference is that we have not to considered ${}^k x_n^t$ variable, which defines relationship between agent location on edges and nodes coverage, and thus we can remove equation (7.51), which is very complex and unclear on the first look.

7.4 Speedup Techniques

According to fact, that booth models are computationally hard, we created three modifications of standard algorithm with better scalability. The first modification is algorithm based on decomposition of given are to subareas and solving the single subareas separately in that way, that optimality of solution is maintained. The second modification is suboptimal iterative algorithm, which iteratively construct joint plan by solving a subplan for single agent with fixation sub-plan of others agents. The third modification is suboptimal iterative algorithm based on rolling horizon, which divided given time horizon to n sub-horizons which are overlapping in twos. The plan is construct by gradual computing of sub-problems, where the next sub-problem is initialized by overlapping part of previous model. We can get next modification by combination these methods. It is especially advantageous use suboptimal always uses a suboptimal algorithms with decomposition algorithm, because a solution quality will be maintained.

7.4.1 Area Decomposition Algorithm

Before we explain the algorithm, we must first definite a reachable space of each agent R^k and reachable graph G'. The reachable space of the agent k contains whole nodes, which aren't remote from the agent's base for more than half of his range

$$n \in R^k \iff dist(B^k; n) \leq range(k)$$

, where $dist(B^k; n)$ sets the length of shortest path from node B^k to n and $range(k)$ is range of agent k, which is compute as proportion of of speed and endurance. The reachable graph $G'(N', E')$ is sub-graph of given graph $G(N, E)$, which contains just nodes thats are reachable by arbitrary agent $N' = \bigcup_{k \in K} R^k$ and E' contains all edges that start and end in nodes from set N' thus:

$$e \in E' \iff in(e) \in N' \wedge out(e) \in N', \forall e \in E.$$

The motivation of decomposition algorithm is that if for pair of agents i and j doesn't exists node, which is reachable by both of them thus $R^j \cap R^i = \emptyset$ we can compute routs for agent separately without lost on optimality, because action of one agent can not has any influence to decision of the second agent. Our goal is thus find how to divided agents into nonintersect sets $S^1 \dots S^s$ such that, if exists node n which is reachable by agent x_i and x_j then x_i and x_j belong into same set

$$a^i \in S^s \wedge a^j \in S^s \iff R^i \cap R^j \neq \emptyset, \forall s \in S, \forall i, j \in K.$$

If we have the distribution agent in non-intersecting sets, we could solve the problem by solving $|S|$ sub-problems. The single sub-problem is solved graph $G^s(N^s, E^s)$, where $N^s = \bigcup_{k \in S} R^k$ and

$$e \in E^s \iff in(e) \in N^s \wedge out(e) \in N^s, \forall e \in E$$

. and set of agent S^s. With different words we decompose our problem P of monitoring graph $G(N, E)$ with set of agent K $P(G(N, E), K)$ into

$$P(G^1(N^1, E^1), S^1), \dots, P(G^s(N^s, E^s), S^s)$$

where $K = S^1 \cup \dots \cup S^s$ and

$$G'(N', E') = G^1(N^1, E^1) \cup \dots \cup G^s(N^s, E^s)$$

, By solving this above mentioned sub-problems we get an optimal route for all agent, however if we want to know also a optimal value of the problem we have to add potential damage on unreachable edges, which is easy to compute,because this edges cannot be never protected.

It should be note, that sometimes exist node, which is reachable by all agents and in this case is there any influence of decomposition algorithm to scalability, however in real situation when we monitoring a structures with length in the tens to hundreds of kilometers this situation is very unlikely. A necessary condition for the functioning of the algorithm is that there is more then just one base.

The algorithm works in the following way: At the beginning we compute a reachable nodes for each agent. Then we create reachable graph, compute an offset of unreachable edges and put each of agent into his own set. After initialization we iteratively test first set with each other sets until each set is not compared with each others. If this two compared set contains some common e.i. intersection of reachable nodes in not empty, set comparing is stopped; all agent in the first set are added into second one as well as reachable nodes. And first set is deleted. Otherwise if there is no set with common reachable node we move first set into set of final sets.

After we have division of agent into set, we compute a rout for each of this set separately with using a standard algorithm described above. the final join plan is composition of these partial plans.

7.4.2 Iterative Algorithm

Seconds speed up technique is an iterative greedy algorithm, which compute join plan by computing sub-plan for single agent with fixation sub-plan of others agents. A significant acceleration of the algorithm is caused by the fact that number of considered possible strategies is reduced from Cartesian product of possible strategies for each agents to their sum. Nevertheless it is also the cause of sub-optimality of given solution because we are just able to find optimal strategy for kth agent only with respect to previous agent, which off course doesn't count with possibility of another agents, same like this one.

The algorithm is based on fact, that maximization of submodular set function can be solved with using iterative algorithm (see Section 5.4) with bounded quality of solution about 63%. So firstly, we need to proof, that our problem is submodular. Let us start with proposing of $R(K, I)$ penalty reduction function, the $R(K, I)$ function is define as a difference of total potential damage of solution without any agent and with set of agents' strategies K of instance I. If we want to proof this, we must to show two things.

- $R(S \cup \{k\}, I) - R(S, I) \leq R(T \cup \{k\}, I) - R(T, I), S \subset T \subset N, k \in N - T$
- $R(S \cup \{k\}, I) - R(S, I) \geq 0$

So that solution quality can be degraded by adding another agent and that the addition of an agent to a larger set of agents brings less improvement less or equal than when the agent is added to a smaller set.

Is easy to see that restriction $R(S \cup \{k\}, I) - R(S, I) \geq 0$ is satisfy, according to definition of computing potential damage for single edge ?? the potentional damage cannot be increased by agent movement. In the whorst case there is no avaiable edges that could be covered by new agent k and the result is same, thus requirment is satisfied. Now we want to show that $R(S \cup \{k\}, I) - R(S, I) \leq R(T \cup \{k\}, I) - R(T, I), S \subset T \subset N, k \in N - T$ is satisfied. There could exist 2 cases:

The first one is that adding K int S wont change a penalty function e.i. $R(S \cup \{k\}, I) - R(S, I) = 0$, this is because there doesn't exists any uncovered node that should be patrolled by agent k. Because $S \subset T$ it is also true $R(T \cup \{k\}, I) - R(T, I) = 0$, thus in this case is restriction satisfied. The second one is that adding K int S wont change a penalty function e.i. $R(S \cup \{k\}, I) - R(S, I) > 0$. This means that exist a set of nodes which are not covered by S in some time steps. If the instance I is patrolled by set of agents T. Instance is also patrolled optimally thus all of nodes covered by R are also covered by set of agents T concurrently there is $T - S$ agents remains to coverage nodes that was uncovered by S; this agents will cover the most valuable part of uncovered nodes . Because of this reason is increment of adding k into set T smaller than if it is added into set S thus restriction is satisfied.

We are improving a quality of solution by repeat of plan creation while the quality of solution is improving. Advantage of repetition is that during the second and next repetition agent compute his sub-plan according to sub-plans of all others agents. The algorithm works in the following way: the sub-plan for single agent is computed iteratively, with respect to previously solved plans. This iteration is repeated until the quality of solution is increased or the maximal number of repetition is done.

Above we proved that out problem is submodular for homogenous set of agent. The problem is submodular even for a set of nonhomogeneous agent, however, the problem of nonhomogeneous agent is that we are not able to find a best agent for next sept of the algorithm. It is because we are no able say if it is better to use an agent with small speed and long endurance or agent with high speed and short endurance. For this reason if we want to preserve a bounded sub-optimality of the solution we must to modify the algorithm in the following way: In each step we don't just a select a random agent, but we solve the sub-plan for all of available agent and we chose an agent with gave us best increment to the quality of the solution.

7.4.3 Time Decomposition Algorithm

The last speed up technique is based on rolling horizon decomposition, where given time horizon is divided into n part for which is compute sub-plan satisfying the restriction that position of agents and value of theirs endurance in the first step of sub-model equals to position of agents and value of theirs endurance in previous sub-model id exists. In this case we compute an join plan,but on the other hand agent are able to compute with respect to future only to a limited extent, depending on the number of steps.

The run of algorithm works in the following way. firstly we decompose time horizon into he required number of subproblems. Then we compute first problem by standard algorithm where we just neglect the requirement that agents ends its tours in theirs bases. For next problems, which are not last we cerate a model where we neglect both of requirements how requirement to start in base, instead which we initialize agents position and endurance according to previous sub-plan so requirement to ending in the base.In the case of last sub-problem we only replace the start in base restriction for initialization. The result we get by connection of all sub-plans.

Chapter 8
Scenarios and Evaluation

In this chapter, we evaluate algorithms proposed above. The chapter is divided into two sections. The first section deals with the influence of parameters on scalability and quality of the solution. All tests in this chapter were performed on synthetic data. The second section deals with the real use of proposed algorithms. We show a disparity of the solution with respect to selected speed-up technique and area discretization. The tests in the second part were performed on three real world scenarios.

8.1 Evaluation

Firstly, we show the influence of graph representation on the scalability of solution. Then we show the influence of area discretization algorithms on the scalability and quality of the solution. At the end of the section, we show scalability and quality of the solution with respect to a selected speed-up technique.

8.1.1 Graph Representation

Firstly, we focus on influence of the graph orientation on the scalability of the solution, thus we compare a basic-NOOPP algorithm (nodes) with basic-EOPP algorithm (edges). We test scalability with respect to number of nodes and to the number of time steps. For scalability test with respect to the number of nodes we use a default graph with the number of nodes $N = \{10, 30, 50, 80, 100\}$ and number of time steps $T = 12$. We evaluate this tests for set of two agents (Figure 8.2) and five agents (Figure 8.1).

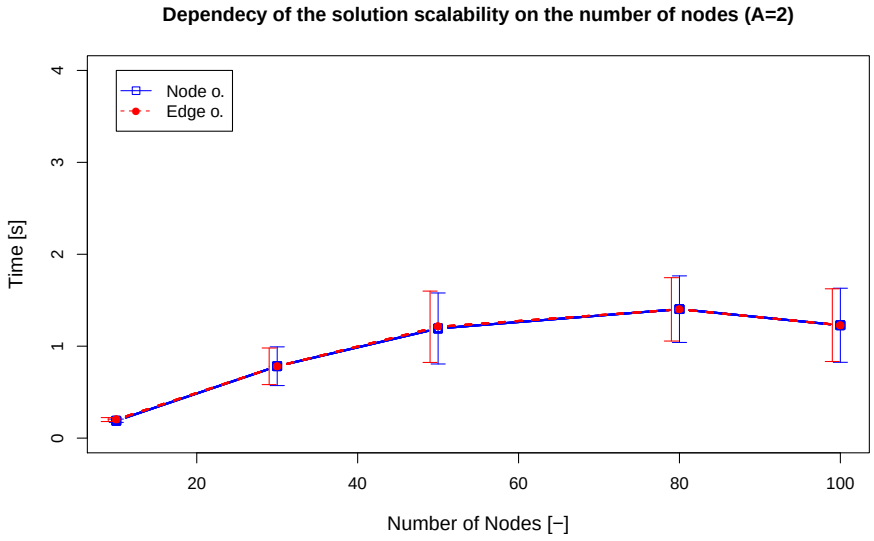


Figure 8.1. Comparison of the scalability of the EOOP and NOOPP algorithms with respect to number of nodes $N = \{10, 30, 50, 80, 100\}$ with 2 agents.

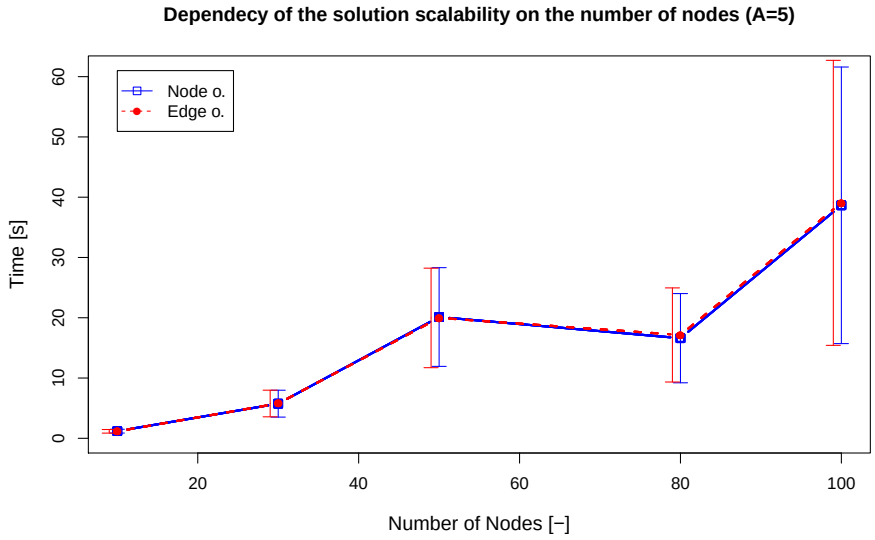


Figure 8.2. Comparison of the scalability of the EOOP and NOOPP algorithms with respect to number of nodes $N = \{10, 30, 50, 80, 100\}$ with 5 agents.

For scalability test with respect to the number of time steps we use thw default graph with number of nodes $N = 50$ and with number of time steps $T = \{2, 4, 6, 8, 12\}$. We evaluate these tests for set of two agents (Figure 8.3) and five agents (Figure 8.4).

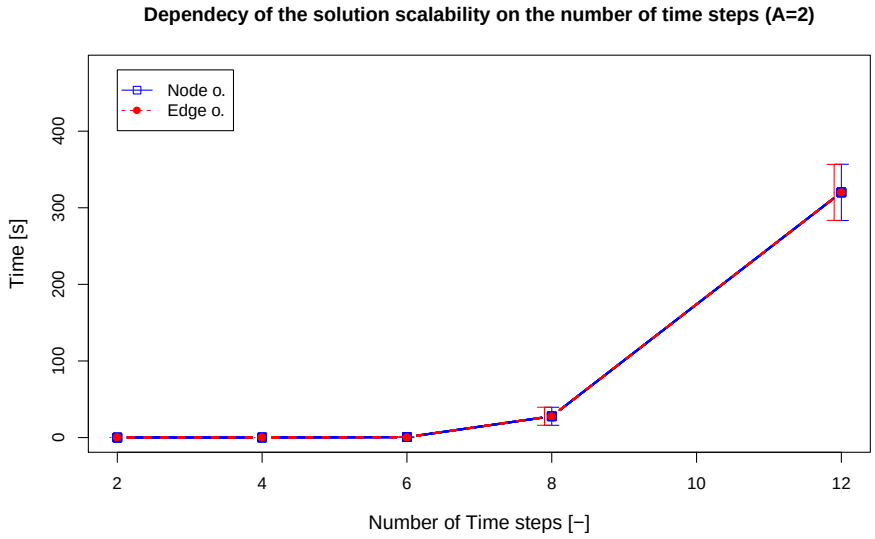


Figure 8.3. Comparison of the scalability of the EOOP and NOOPP algorithms with respect to number of time steps $T = \{2, 4, 6, 8, 12\}$ with 2 agents.

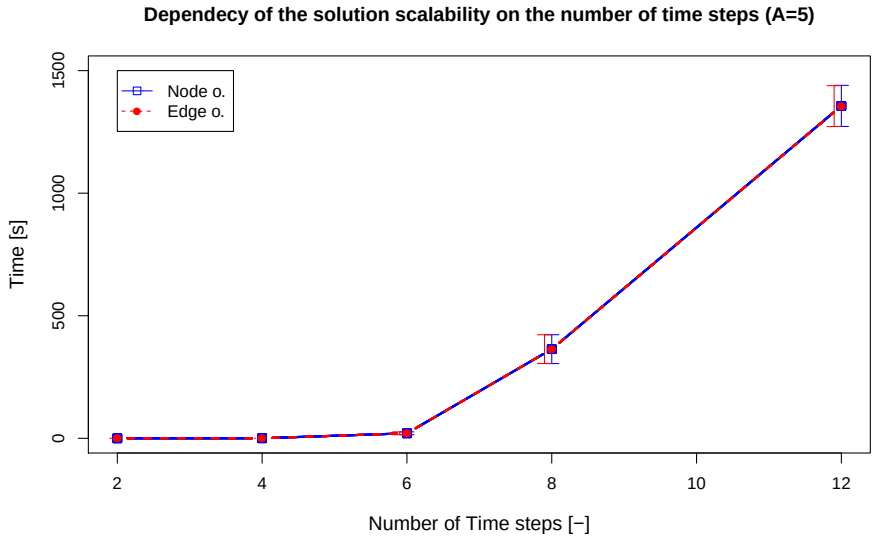


Figure 8.4. Comparison of the scalability of the EOOP and NOOPP algorithms with respect to number of time steps $T = \{2, 4, 6, 8, 12\}$ with 5 agents.

From the evaluation results we can say that selection of graph orientation has no influence on the scalability and both models are performing equally. The difference of average solutions is less than one second. We can also see that scalability of the algorithms does not directly depend on the number of nodes. It is because of the fact that with the increasing number of nodes, the strategy space of the agent is restricted by a smaller planning horizon (i.e., number of time steps). The scalability is dependent on the number of time steps, i.e., on the planning horizon where the dependency is exponential. The scalability of algorithms also depends on the number of agents where the scalability with 5 agents differs in order of magnitude compared to case with 2 agents. In following test we used only the complete-NOOP algorithm.

8.1.2 Area Discretization

In this section, we show the influence of area discretization on scalability and quality of solution. Before we show the tests themselves, we have to explain how the number of nodes is selected. In Figure 8.6 we can observe dependency of scalability on the number of nodes. The number of nodes means the number of nodes of graph discretization. For the hexagon hull discretization, the number of nodes means that resulted graphs were created from a convex hull of graph with n nodes. Dependency of the number of nodes of the hexagon hull on the size of graph from which it is created can be seen in Figure 8.5.

As you can see with increasing number of original nodes, the ratio between number of nodes of original graph and hexagon hull is decreased. As it can be seen, for small original graphs the number of nodes in hexagon hull is larger, however, for large original graphs the number of nodes in hexagon hull is smaller. It is because fact, that with increasing number of nodes the density of pipelines increases. The difference between number of edges is more remarkable. It is because of the fact, that the original graph is a tree graph thus it has $n - 1$ edges but in hexagon hull, each of nodes which is not located on the edge of hull is connected by its neighbours by 6 edges. These results are of course valid only for our generator.

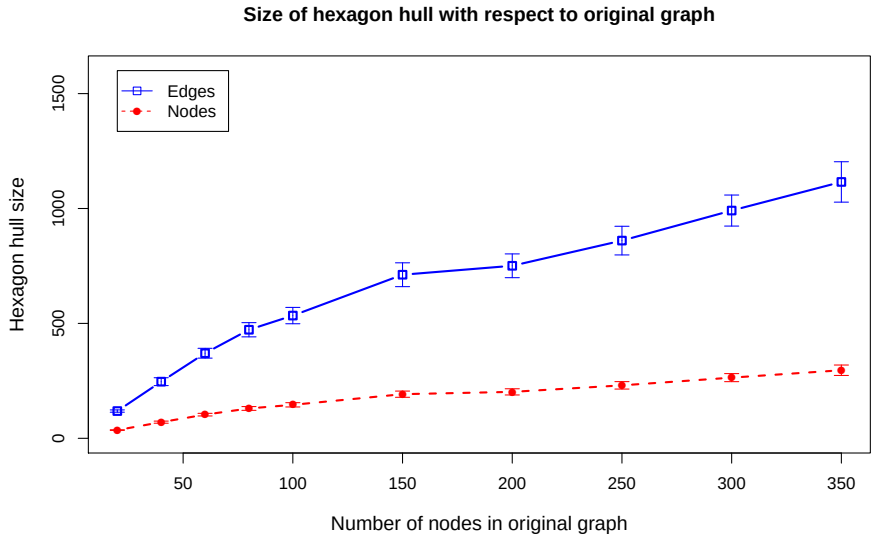


Figure 8.5. Number of nodes and edges in hexagon hull with respect to number of nodes in original graph.

We test the influence of area representation on the scalability with respect to both the number of nodes and the number of time steps. The Complete-NOOPP algorithm was used for testing. For scalability test with respect to the number of nodes, we use a default graph with the number of nodes $N = \{10, 30, 50, 80, 100\}$ and the number of time steps $T = 12$ and 3 agents.

The result of this test can be seen in Figure 8.6. As you can see, the scalability of the hexagon hull representation is about 40-time worse than the scalability of graph area representation. It is because agents have a higher degree of freedom of movement in each time step. The scalability of extended graph representation is about 5-10 time worses than scalability of graph representation. It is also because the fact that the degree of freedom of movement is higher in each step as in the hexagon hull representation, however, the increment of freedom is significantly lower.

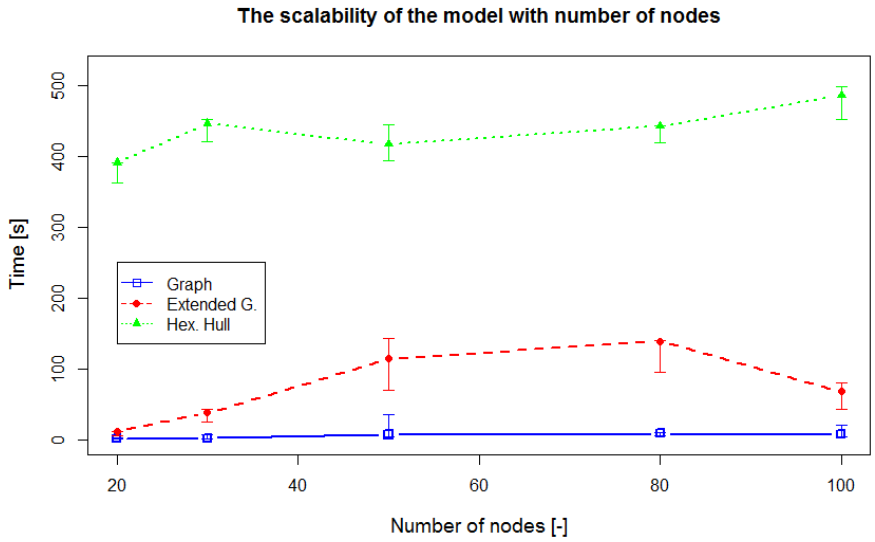


Figure 8.6. Comparison of the area discretization's influence on the scalability with respect the number of nodes N = [10, 30, 50, 80, 100] with 3 agents on the default graph with time horizon T= 12.

For scalability test with respect to the number of nodes, we use a default graph with the number of nodes $N = \{6, 12, 18, 24\}$ and the number of nodes $N = 50$ and 3 agents. The result of this test can be seen in Figure 8.7, where the results are similar as in scalability of the model with the respect to the number of nodes Figure 8.6

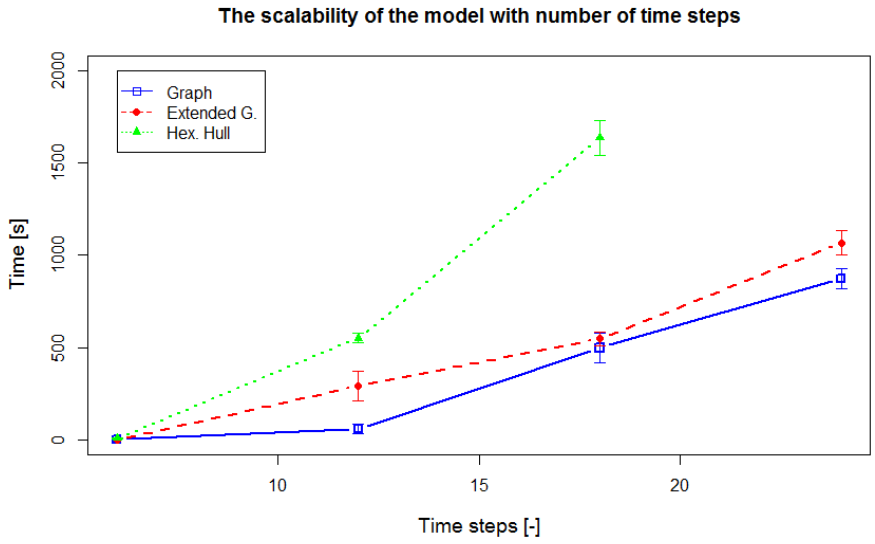


Figure 8.7. Comparison of the area discretization's influence on the scalability with respect to the number of time steps $T = \{6, 12, 18, 4\}$ with 3 agents on the default graph with 50 nodes.

We also test the influence of the area representation on the quality of the solution with respect to the number of nodes. The quality of the solution is measured as the ratio between the sum of potential damages over all nodes and time steps when the area is not patrolled and when is patrolled using given strategy. For this test we use same data initialization as for scalability testing. The influence of area representations

on the quality of the solution can be seen in Figure 8.8. The results are displayed in percentages with respect to the quality of graph area representation. As you can see, the quality of solution for the hexagon hull decreases with respect to times steps compared with the quality of solution for the graph representation. It means that hexagon hull is inappropriate for our solution because it provides both the worst scalability and the worst quality. The extended graph representation provides quality of solution better about 4% then graph representation in case of 24 time steps, however, it looks that the difference between quality will be higher with increasing number of time steps.

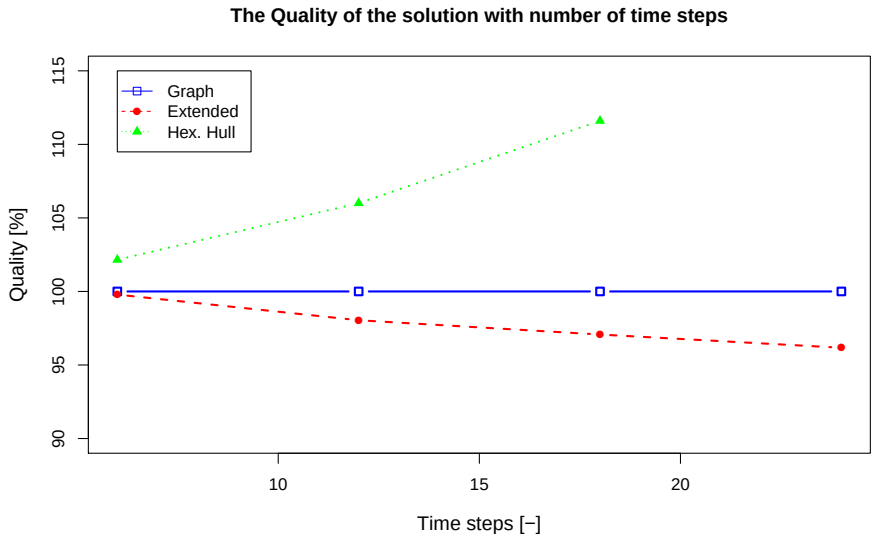


Figure 8.8. Comparison of the area discretization's influence to the quality of the solution with respect to the number of time steps $T = \{6, 12, 18, 4\}$ with 3 agents on the default graph with 50 nodes.

According to the results, both graph representation and extended graph representation are suitable. The extended graph representation provides the best quality of solution; however, the scalability of the solution is worse than scalability of graph representation. The choice of appropriate representation dependents on preference of a better scalability or a better quality.

8.1.3 Influence of Endurance

In this section, we focus on dependency of the solution quality (Figure 8.9) and scalability (Figure 8.10) on the endurance restriction. For test we use default graph with the number of nodes $N = 80$, with the number of time steps $T = 18$, three agents and endurance $E = \{6, 10, 12, 18\}$. The problem was evaluated with using Complete-NOOPP algorithm.

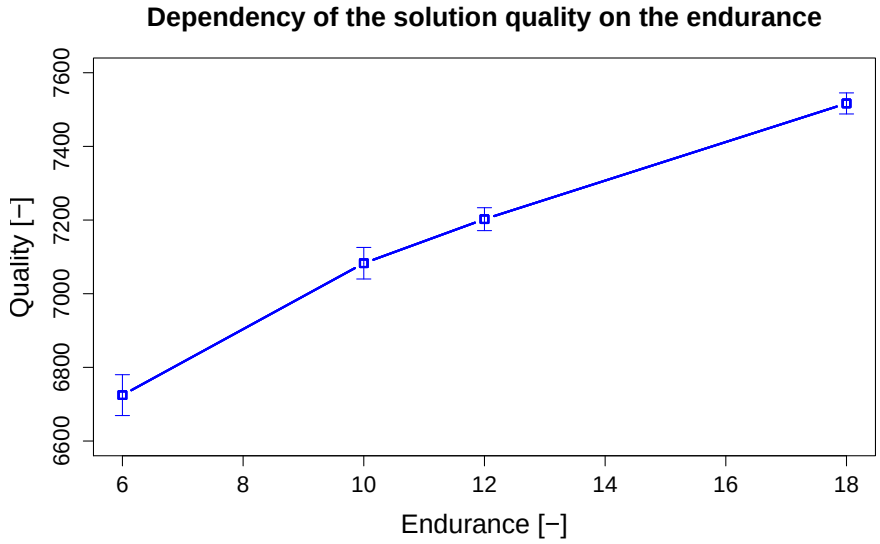


Figure 8.9. Dependency of the scalability of the solution on the endurance $E = \{6, 10, 12, 18\}$ with 3 agents and 18 time steps on the default graph with number of nodes $N = 50$.

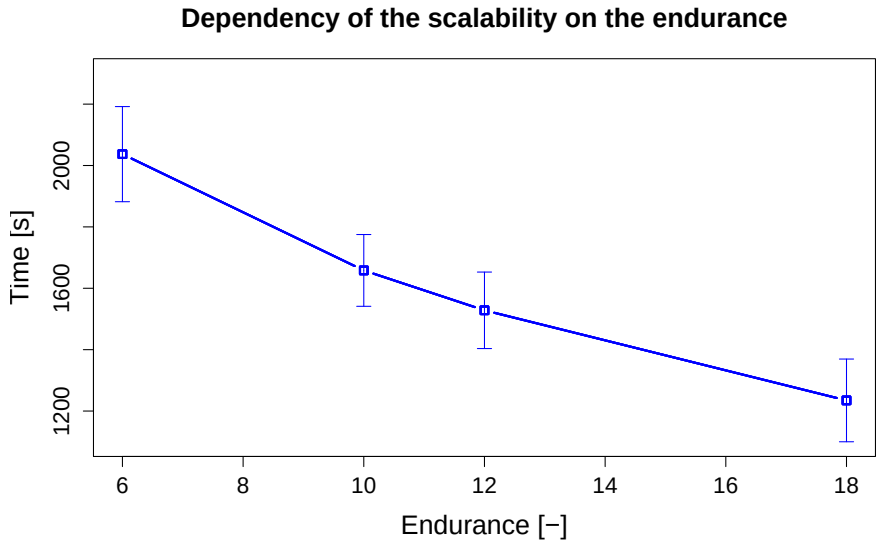


Figure 8.10. Dependency of the solution quality on the endurance $E = \{6, 10, 12, 18\}$ with 3 agents and 18 time steps on the default graph with number of nodes $N = 50$.

According to the result, we can say that decreasing of the endurance has a negative impact on both the quality and the scalability of the solution. It is because for one agent the optimal strategy is one large circle, but with decreasing endurance the resulting strategy is combination of circles with respect to endurance restriction which is more difficult to compute. Concurrently with decreasing endurance, the agents' range also decreases thus the quality of the solution gets worse.

8.1.4 Speed-up Techniques

In this section, we focus on dependency of the solution quality (Figure 8.14) and scalability (Figure 8.11) on the number of timesteps when different speed-up techniques

are used. For test we use default graph with the number of nodes $N = 200$, with the number of time steps $T = \{6, 12, 18, 24, 36\}$, three agents and endurance $E = 6$.

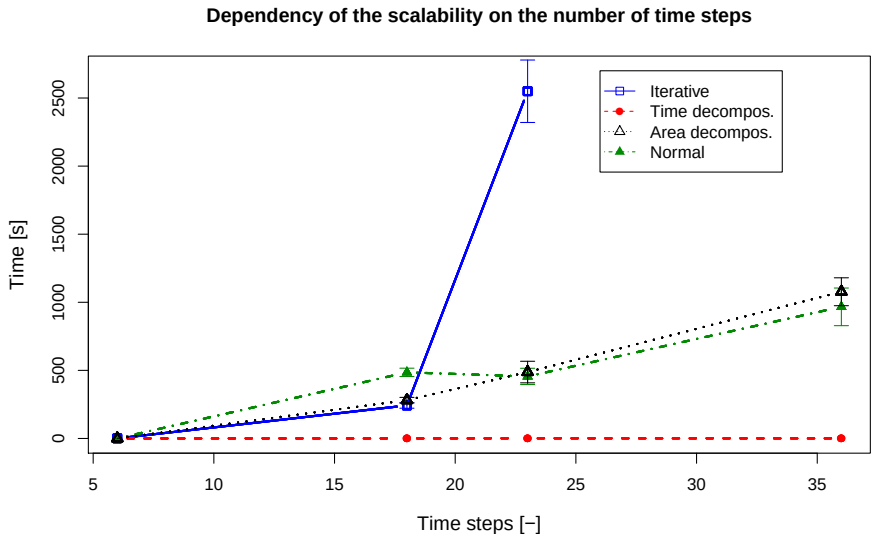


Figure 8.11. Dependency of the solution scalability on the number of time steps $T = \{6, 18, 24, 36\}$ with 3 agents and endurance $E = 6$ on the default graph with number of nodes $N = 200$.

According to results, we can say that scalability of time decomposition is significantly better than scalability of normal algorithm. It is because that the computation complexity of time decomposition is $O(\frac{T}{E} \cdot \tau^e)$ where τ^e is time necessary to solve problem with time horizon equal to endurance, which is significantly lower than time necessary to solve the problem on the full time horizon.

The scalability of area decomposition is not better than scalability of normal model and the scalability of the iterative algorithm is even significantly worse. It is an interesting fact which is inconsistent with the results of our previous work [3] (Figure 8.12).

We try to describe why the results are so different. Let us start with time complexity of iterative algorithm and when is suitable to use this speed-up technique. The iterative algorithm works on principle that with the increasing number of agents the scalability of the model decreases (see difference between Figure 8.3 and Figure 8.4). Algorithm uses this fact and decomposes the problem into set of single agent sub-problems where union of their strategies gives us the suboptimal solution of the original problem. Normally submodular iterative algorithms compute the sub-problem once for each agent. In our work, we increase the quality of the given solution by repeating this algorithm until the quality of the next iteration stays the same. The time complexity of our algorithm is $O(n \cdot |K| \cdot \tau^S)$ where n is the number of iterations of the algorithm and on average equals to 3 and τ^S is time complexity of a single agent sub-problem.

Thus the iterative algorithm will be faster than original model if the computation of single agent model is at least $n \cdot |K|$ faster than solving original model.

We show the property which model must satisfy that it was suitable for using iterative algorithm. Now we explain the question why the ratio between multi-agent model time complexity and single agent time complexity differs. In general, we can say that three parameters have an influence on this ratio. The first of them is endurance restriction. A lower endurance worsens the scalability of the model solution (Figure 8.10) and the

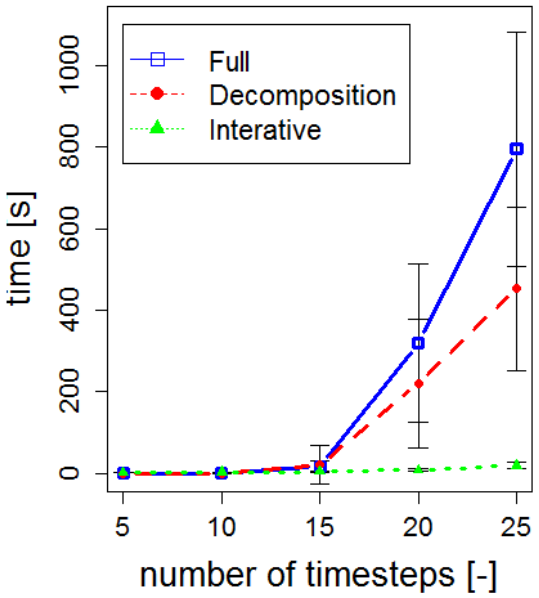


Figure 8.12. Dependency of the solution scalability with respect to number of time steps.

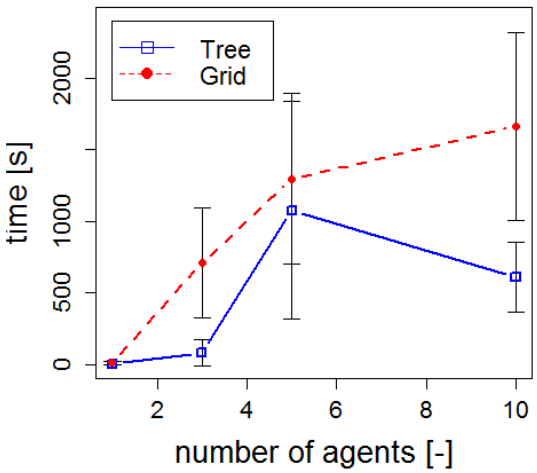


Figure 8.13. Dependency of the solution scalability with respect to the number of agents.

degradation of scalability is much more significant in the case of single agent model thus the ratio decreases.

The second parameter is the number of agents. As it can be seen in Figure 8.13, when increasing the number of agents we reach a point from which the scalability of models begins to improve. In this case, the ratio decreases.

Another parameter, which has influence on the suitability of iterative algorithm, is the calculation process of Branch-and-Cut algorithm(B&C). In the case of models that are suitable for solving by iterative algorithm, the B&C finds the optimal value most of time and rest of time needs to prove that the solution is optimal. In the case of problems that are unsuitable for solving by iterative algorithm the situation is opposite. The problem is that the iterative algorithm significantly improves the time complexity of the first part, however, the second one (with slightly better time complexity) is computed $r \cdot |K|$-times thus the result of the iterative algorithm may be worse than result of the normal algorithm.

One of the possibilities how to partially solve this problem is settings the `Gap factor`. The `Gap factor` sets gap between the best integer objective and the objective of the best node remaining. When this difference falls below the value of this parameter, the mixed integer optimization is stopped.

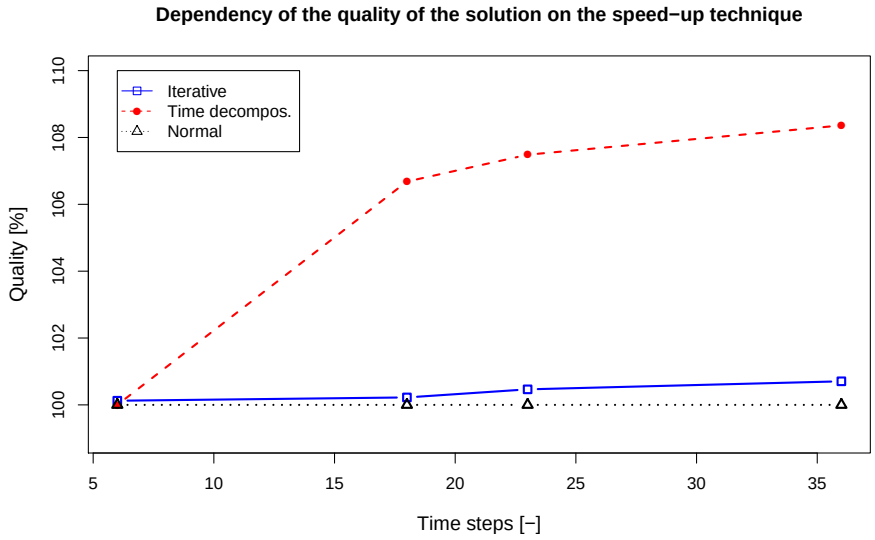


Figure 8.14. Dependency of the solution quality on the number of time steps $T = \{6, 18, 24, 36\}$ with 3 agents and endurance $E = 6$ on the default graph with number of nodes $N = 200$.

In Figure 8.14 is shown dependency of the solution quality on the number of time steps for diferent speed-up techniques due to optimal solution (Normal). As you can see, iterative algorithm achieves the solution quality greater than 99 percent of the optimal solution. The quality of iterative is a bit worse but stil achves the solution quality greater than 90 procent of the optimal solution.

8.2 Scenarios

In this section we show application of our algorithm on two real world pipeline systems namely oil pipeline system in Russia (Scenario 2) and oil pipeline system in Rostock's harbor(Scenario 1).

We used Open Street Map [1]) for obtaining realistic oil pipeline system data,where there is Oil and Gas infrastructure project [2]). Oil and Gas infrastructure is a project to map oil and gas fields, individual wells and rigs, pipelines, transshipment centers and refineries.

We used hexagon hull algorithm for area decomposition, notwithstanding that we have shown that hexagon hull algorithm is the least suitable. We used this algorithm because of incompleteness of data obtained from Open Street Map, where pipelines are often interrupted therefore we do not have a continuous graph if we used a graph or extended graph discretization algorithm.

We create two tests on both scenarios. The first test is monitoring of given system with 5 agents and 3 bases and the second one is monitoring of given system with 10 agents and 5 bases. Both scenarios were solved with using time decomposition speed-up technique and the time horizon was decomposed into segments with 6 time steps. All agents had a unitary speed and their endurance was equal to time horizon. Number of time steps was set to 180 in both scenarios, which equals to 10 minutes of monitoring. Computation time was in both scenarios lower than 1 minute in the case of first test or

[1]) `http://www.openstreetmap.org/`
[2]) `http://wiki.openstreetmap.org/wiki/WikiProject_Oil_and_Gas_Infrastructure`

5 minutes in the case of the second test; computing times include solving of the optimal bases placement problem. Because information about pipelines as a diameter, age etc. is not freely available, we set a priority of all segments of the pipeline system to 1.

We use Cesium [1]) for scenarios visualization. All visualizations can be found on following pages: `http://pipeprotection.jakubondracek.cz/`.

8.2.1 Scenario 1

The first scenario is pipeline system in Rostock's harbour. The system is discretized into hexagon hull with 650 nodes, where each line has 180 meters. The pipeline system together with its discretization can be seen in Figure 8.15.

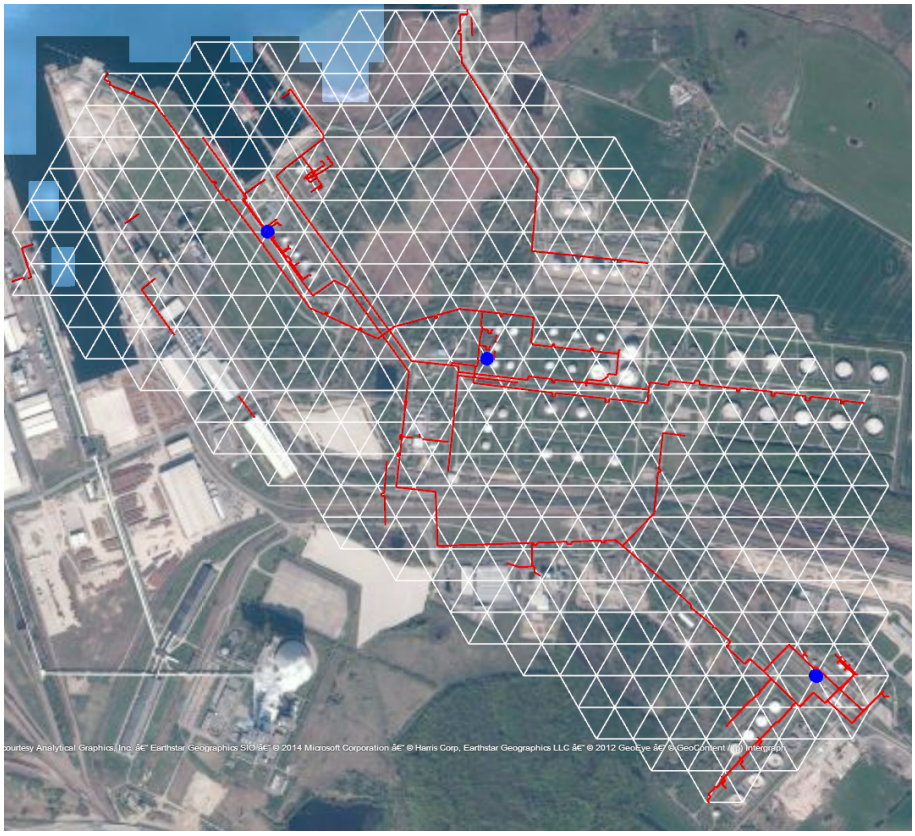


Figure 8.15. Pipeline system in Rostock's harbour (red lines) together with its hexagon hull discretization (white lines) and bases location for first test(blue cylinder).

In Figure 8.16 it is shown the solution of the problem with 5 agents and 3 bases and in Figure 8.17 it is shown the solution of the problem with 10 agents and 5 bases. In these figure you can see a problem of time decomposition algorithm where sometimes agents patrol narrow area instead of a longer cycle (blue strategy). This is due to the fact that the bases are located in areas with high density of pipelines and agetns plan their plan only with regard to the very near future, therefore to them it is not optimal to patrol the remote area. However, as you can see in cyan strategy despite the use of time decomposition agents sometimes behave as expected. In `http://pipeprotection.jakubondracek.cz/` you can see how monitoring strategies are changing with changing size of subproblem in time decomposition algorithm.

[1]) `http://cesiumjs.org/`

Figure 8.16. Visualization of the solution for 5 agents and 3 bases.

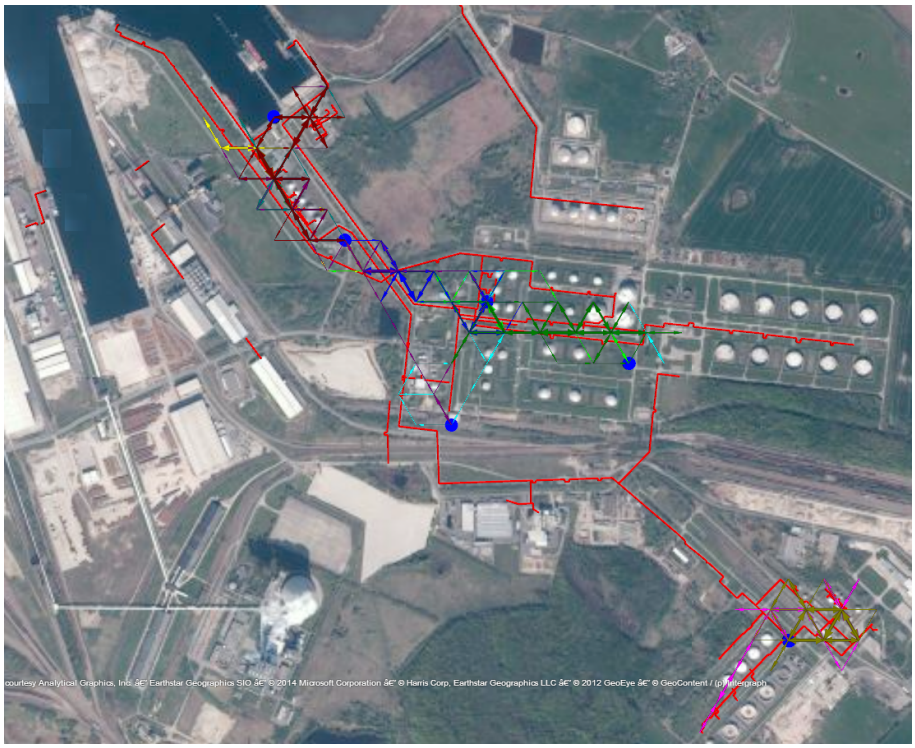


Figure 8.17. Visualization of the solution for 10 agents and 5 bases.

8.2.2 Scenario 2

The second scenario is pipeline system in Russia. The system is discretized into hexagon hull with 636 nodes, where each line has 360 meters. The pipeline system together with its discretization can be seen in Figure 8.18.

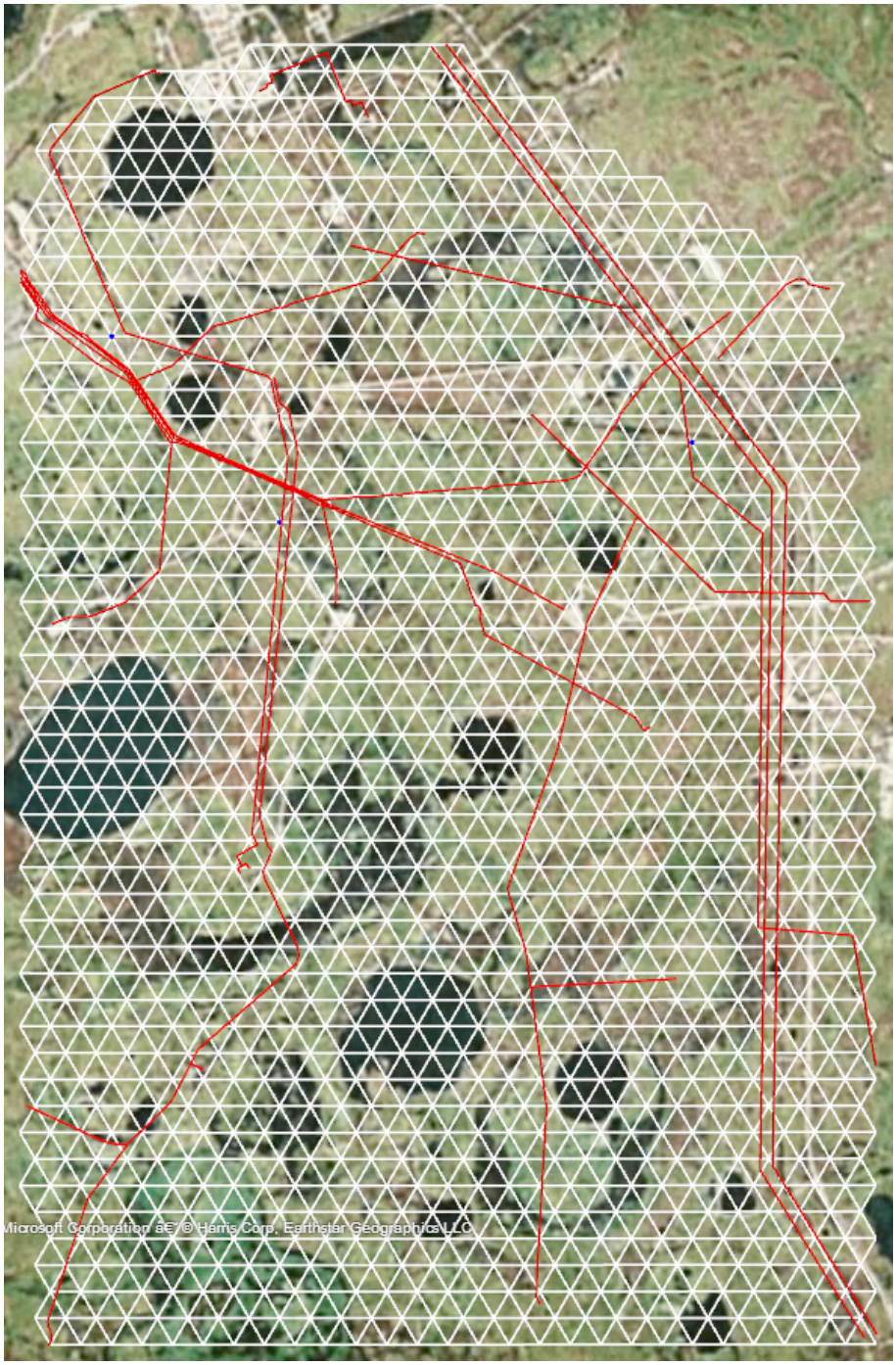


Figure 8.18. Pipeline system in Russia (red lines) together with its hexagon hull discretization (white lines) and bases location for first test(blue cylinder).

In Figure 8.19it is shown the solution of the problem with 10 agents and 5 bases. Due to lower density of pipelines system the quality of the solution is much better than quality of the solution of first scenario. But even here there are exceptions which are probably suboptimal (blue and cyan strategies).

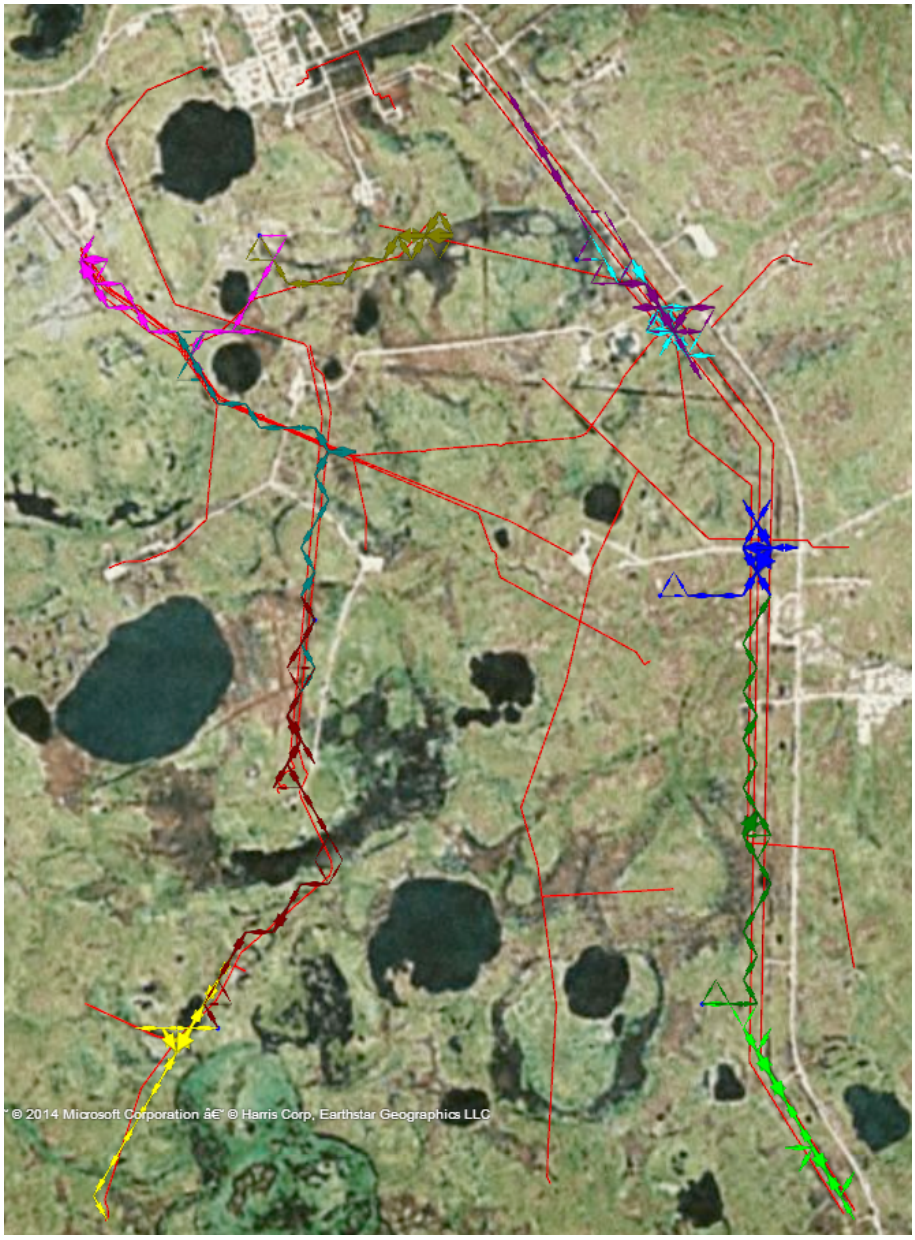


Figure 8.19. Visualization of the solution for 10 agents and 5 bases.

We have assessed capabilities of the developed algorithms on two scenarios. Both cases show current limitations of our approach, however, they also demonstrate the ability of our approach to reasonably solve real-world problems.

Chapter 9
Discussion and Conclusion

We dealt with the problem of monitoring of the oil pipeline systems with using a set of UAVs. Our goal was to detect oil spills as soon as possible thus minimize damage to the environment. We solved this by minimizing age of information weighted by significance of a given segment over all segments of monitored pipeline system.

We have created three different spatial discretizations namely graph discretization, hexagon convex hull discretization and extended graph discretization. We have also proposed two different time-space discretizations with both homogeneous and non-homogeneous time periods.

We have created a basic ILP program. Concurrently we created a set of extensions taking into account:

- Range of UAS, reflected by their flight endurance
- Recharge time
- Various speed of UAS

Consideration of physical properties of UAS is the largest increment of our work compared with previous work typically using travelling salesman problem (TSP) formulation. However, it is impossible to find TSP cycle on large patrolled are as pipeline systems are, due to physical restrictions of UAS. Therefore, our resulting strategy is not a cycle, but a set of circles for a given time horizon. Another increment of our work is that we have extended the problem of area surveillance by a problem of optimal bases placement.

We have shown that finding the optimal solution of the problem is computationally hard, so we have developed three speed up techniques to improve scalability. The first of these techniques is area decomposition algorithm that has been used often in sub-optimum form in previous work. In our work, we introduced a version resulting from endurance restrictions and preserves optimality of the final solution. Another speedup technique is iterative iterative algorithm utilizing submodular property of our problem to find bounded-suboptimal solution. We have also expanded this algorithm by a version with non-homogeneous agents. The third technique is time discretization without any theoretical proven bound on the quality of the solution.

We have created two real-world scenarios on which we show that developed algorithms are suitable for solving real-world problems. Developed algorithms were successfully implemented into project Arethus, which is Decision Support System for surveillance developed in the Agent Technology Center.

Currently, this topic allows future extensions, whether the inclusion of optimization against an attacker, which is a question of the game theory. Another open question is solving the problem for an arbitrarily long time horizon because, as it has been shown, scalability with the number of time steps is the largest bottleneck of the proposed algorithms. There are two potential solutions to this problem: solve the problem as Markov decision process, or we can find a strategy in a form of a cycle which repetition is optimal; however, such strategy may not always exist.

References

[1] Christopher O Orubu, Ayodele Odusola, and William Ehwarieme. The nigerian oil industry: environmental diseconomies, management strategies and the need for community involvement. *Journal of Human Ecology*, 16(3):203–214, 2004.

[2] CO Orubu, A Fajingbesi, A Odusola, and N Magbagbeola. Environmental regulations in the nigerian petroleum industry: Compliance status of compiance and implications for sustainable development. Technical report, Research Report, ACBF/NCEMA, Ibadan, 2002.

[3] Jakub Ondráček, Ondřej Vaněk, and Michal Pěchouček. Monitoring oil pipeline infrastructures with multiple unmanned aerial vehicles. In *PAAMS 2014, LNAI 8473*, pages 219–230, 2014.

[4] Robert E Bixby. A brief history of linear and mixed-integer programming computation. *Optimization Stories*, pages 107–121, 2012.

[5] George B Dantzig. Programming in a linear structure. *Washington, DC*, 1948.

[6] Victor Klee and George J Minty. How good is the simplex algorithm. Technical report, DTIC Document, 1970.

[7] WW Garvin, HW Crandall, JB John, and RA Spellman. Applications of linear programming in the oil industry. *Management Science*, 3(4):407–430, 1957.

[8] CARLTON E Lemke. The dual method of solving the linear programming problem. *Naval Research Logistics Quarterly*, 1(1):36–47, 1954.

[9] Leonid G Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53–72, 1980.

[10] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 302–311. ACM, 1984.

[11] Harlan Crowder, Ellis L Johnson, and Manfred Padberg. Solving large-scale zero-one linear programming problems. *Operations Research*, 31(5):803–834, 1983.

[12] Harlan Crowder and Manfred W Padberg. Solving large-scale symmetric travelling salesman problems to optimality. *Management Science*, 26(5):495–509, 1980.

[13] Tony J Van Roy and Laurence A Wolsey. Solving mixed integer programming problems using automatic reformulation. *Operations Research*, 35(1):45–57, 1987.

[14] John Allen and Brian Walsh. Enhanced oil spill surveillance, detection and monitoring through the applied technology of unmanned air systems. In *International oil spill conference*, volume 2008, pages 113–120. American Petroleum Institute, 2008.

[15] Erich R Gundlach and Miles O Hayes. Vulnerability of coastal environments to oil spill impacts. *Marine technology society Journal*, 12(4):18–27, 1978.

[16] Jill Petersen, Jacqueline Michel, Scott Zengel, Mark White, Chris Lord, and Colin Plank. Environmental sensitivity index guidelines. version 3.0. *NOAA Technical Memorandum NOS OR&R*, 11:1–192, 2002.

[17] Jasper Agbakwuru. Pipeline potential leak detection technologies: Assessment and perspective in the nigeria niger delta region. *Journal of Environmental Protection*, 2(8), 2011.

[18] Phil Hopkins. The structural integrity of oil and gas transmission pipelines. *Comprehensive Structural Integrity, Elsevier Publishers Penspen Ltd., Berlin*, 2002.

[19] Alex W Dawotola, PHAJM van Gelder, and JK Vrijling. Multi criteria decision analysis framework for risk management of oil and gas pipelines. *RELIABILITY, Risk and Safety. London: Taylor & Francis Group*, pages 307–314, 2010.

[20] Prasanta Kumar Dey, Stephen O Ogunlana, and Sittichai Naksuksakul. Risk-based maintenance model for offshore oil and gas pipelines: a case study. *Journal of Quality in Maintenance Engineering*, 10(3):169–183, 2004.

[21] Imad Jawhar, Nader Mohamed, and Khaled Shuaib. A framework for pipeline infrastructure monitoring using wireless sensor networks. In *Wireless Telecommunications Symposium, 2007. WTS 2007*, pages 1–7. IEEE, 2007.

[22] Andreas Krause, Jure Leskovec, Carlos Guestrin, Jeanne VanBriesen, and Christos Faloutsos. Efficient sensor placement optimization for securing large water distribution networks. *Journal of Water Resources Planning and Management*, 134(6):516–526, 2008.

[23] Michal Jakob, Eduard Semsch, D Pavlıcek, and Michal Pěchoucek. Occlusion-aware multi-uav surveillance of multiple urban areas. In *6th Workshop on Agents in Traffic and Transportation (ATT 2010)*, pages 59–66, 2010.

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

[25] Nikhil Nigam and Ilan Kroo. Persistent surveillance using multiple unmanned air vehicles. In *Aerospace Conference, 2008 IEEE*, pages 1–14. IEEE, 2008.

[26] Fabio Pasqualetti, Joseph W Durham, and Francesco Bullo. Cooperative patrolling via weighted tours: Performance analysis and distributed algorithms. *Robotics, IEEE Transactions on*, 28(5):1181–1188, 2012.

[27] Ondřej Hrstka. Intelligent surveillance algorithms for harbor security. Master's thesis, Czech Technical University, 2013.

[28] Martin Selecký, Petr Váňa, Milan Rollo, and Tomáš Meiser. Wind corrections in flight path planning. *International Journal of Advanced Robotic Systems*, 10, 2013.

[29] C. H. Achebe, U. C. Nneke, and O. E. Anisiji. Analysis of oil pipeline failures in the oil and gas industries in the niger delta area of nigeria. *Proceedings of the International MultiConference of Engineers and Computer Scientists*, 2:1274–1279, 2012.

[30] Alexander Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[31] Ailsa H Land and Alison G Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960.

[32] Ralph E Gomory et al. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, 64(5):275–278, 1958.

[33] Ralph Gomory. An algorithm for the mixed integer problem. Technical report, DTIC Document, 1960.

[34] Gérard Cornuéjols. Valid inequalities for mixed integer linear programs. *Mathematical Programming*, 112(1):3–44, 2008.

[35] Manfred Padberg and Giovanni Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM review*, 33(1):60–100, 1991.

[36] Robert E Bixby. Solving real-world linear programs: A decade and more of progress. *Operations research*, 50(1):3–15, 2002.

[37] IBM ILOG Cplex. 12.2 user's manual, 2010.

[38] James Uber, Robert Janke, Regan Murray, and Philip Meyer. Greedy heuristic methods for locating water quality sensors in distribution systems. In *ASCE World Water and Environmental Resources Congress 2004*, pages 481–489, 2004.

[39] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical Programming*, 14(1):265–294, 1978.

[40] Branislav Bošanský, Ondřej Vaněk, and Michal Pěchouček. Strategy representation analysis for patrolling games. In *Proceedings of AAAI Spring Symposium GTSSH*, 2012.

Appendix A
Zadání práce

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Jakub O n d r á č e k

Studijní program: Otevřená informatika (bakalářský)

Obor: Informatika a počítačové vědy

Název tématu: Inteligentní algoritmy pro monitorování životního prostředí v blízkosti ropovodů pomocí bezpilotních systémů

Pokyny pro vypracování:

1. Nastudujte problém ochrany prostředí v blízkosti ropovodu.
2. Nastudujete použití bezpilotních prostředků (UAS) pro monitorování infrastruktur.
3. Navrhněte a implementujte sadu algoritmů schopných plánovat trajektorie pro množinu UAS, které monitorují ropovod.
4. Vytvořte sadu scénářů pro patrolování ropovodu.
5. Vyhodnoťte navržené algoritmy na vytvořených scénářích.

Seznam odborné literatury:

[1] Hrstka O.: Intelligent Surveillance Algorithms for Harbor Security. Master Thesis. 2013.
[2] Hooker J.: Planning and Scheduling by Logic-Based Benders Decomposition. 2005.
[3] Pitkin J.: Oil, Oil, Everywhere: Environmental and Human of Oil Extraction in the Niger Delta. 2013.

Vedoucí bakalářské práce: Ing. Ondřej Vaněk, Ph.D.

Platnost zadání: do konce letního semestru 2014/2015

L.S.

doc. Dr. Ing. Jan Kybic
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 10. 1. 2014

Appendix B
Specification

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

BACHELOR PROJECT ASSIGNMENT

Student: Jakub O n d r á č e k

Study programme: Open Informatics

Specialisation: Computer and Information Science

Title of Bachelor Project: Intelligent Algorithms for Monitoring of Environment Around Oil Pipe Systems Using Unmanned Aerial Systems

Guidelines:

1. Study the problem of environmental protection in the vicinity of oil pipes.
2. Study the utilization of unmanned aerial systems (UAS) for infrastructure monitoring.
3. Design and implement a set of algorithms able to plan trajectories for a set of UAS monitoring the pipe line.
4. Create a set of scenarios for oil-pipe patrolling.
5. Evaluate the designed algorithms on created scenarios.

Bibliography/Sources:

[1] Hrstka O.: Intelligent Surveillance Algorithms for Harbor Security. Master Thesis. 2013.
[2] Hooker J.: Planning and Scheduling by Logic-Based Benders Decomposition. 2005.
[3] Pitkin J.: Oil, Oil, Everywhere: Environmental and Human of Oil Extraction in the Niger Delta. 2013.

Bachelor Project Supervisor: Ing. Ondřej Vaněk, Ph.D.

Valid until: the end of the summer semester of academic year 2014/2015

L.S.

doc. Dr. Ing. Jan Kybic
Head of Department

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, January 10, 2014