Czech Technical University in Prague

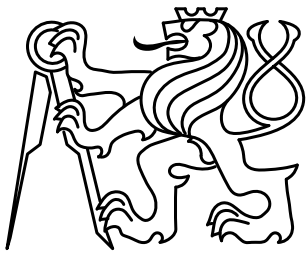
Faculty of Electrical Engineering

Doctoral Thesis

June 2013 *Ondřej Vaněk*

Czech Technical University in Prague

Faculty of Electrical Engineering
Department of Cybernetics



Computational Methods for Transportation Security

Doctoral Thesis

Ondřej Vaněk

Prague, June 2013

Ph.D. Programme: Electrical Engineering and Information Technology (P2612)
Branch of study: Artificial Intelligence and Biocybernetics (3902V035)

Supervisor: Prof. Ing. Vladimír Mařík, DrSc.
Supervisor-Specialist: Ing. Michal Jakob, Ph.D.

Dedicated to my parents Milena and Luboš,
my sister Kristýna,
and my niece Baruška.

Acknowledgments

This work would not exist without the support of many people. First of all, I would like to thank my supervisor prof. Vladimír Mařík who offered me to start PhD at his department and supported me throughout the years. I am indebted to all colleagues from the Agent Technology Center, especially to prof. Michal Pěchouček who introduced me to world-class research and to Michal Jakob who taught me how think about any kind of problem. Their insights and discussions helped me to make the first research steps captured in this thesis. I would like to thank to Branislav Bošanský and Viliam Lisý for convincing me to stay close to the game theory and for their numerous advice about my research, to Ondřej Hrstka who implemented a large part of the AgentC simulation framework, to Michal Čáp, Antonín Komenda, Jiří Vokřínek, Tomáš Pevný, Peter Novák, Martin Rehák, the CAMNEP team, Martin Selecký, Štěpán Kopřiva, and other ATG members for valuable discussions about our research topics and to Barbora Jeniková and Marie Kavanová for the smooth operation of our research group.

Over the years I have been fortunate to collaborate with many great researchers. I would like to thank to all members of the TEAMCORE group from the University of Southern California, especially to prof. Milind Tambe, Manish Jain, Zhengyu Yin, and James Pita. They taught me how to collaborate on a joint project, how to write solid, rejection-proof papers and how to transfer results from abstract mathematical models into real-world applications. I would like to thank to Rong Yang, Jun-young Kwak, Jason Tsai, Matthew Brown, Eric Shieh, Bo An, Albert Jiang, Fei Fang, Thanh Nguyen and others for making my stay in their research group pleasant and comfortable. I would like to thank to Vincent Conitzer and Dmytro Korzhyk for endless corrections of our joint paper and to Janusz Marecki for sharing his excitement about AI with me.

Finally, I would like to thank to my family for their endless support and to my friends who are making my life enjoyable and exciting.

I acknowledge U.S. Office of Naval Research project N000140910537—Adversarial Modeling and Reasoning in the Maritime Domain, and Czech Ministry of Education, Youth and Sports project LH11051—Formal Models and Effective Algorithms for Intelligent Protection of Transportation Infrastructure and Czech Science Foundation grant P202/12/2054—Security Games in Extensive Form which supported my research.

Abstract

Effective security of today's transportation infrastructures is difficult to achieve without computer-aided approach. We develop a set of security-enhancing computational methods for transportation infrastructures where many independent agents operate on daily basis in presence of an intruder. Specifically, we consider an interaction between two agents representing two principal conflicting sides: an evading agent (Evader) moving through the infrastructure to reach a destination and an intercepting agent (Defender), aiming to detect the evading agent and prevent him from reaching the destination, motivated by scenarios of contemporary maritime piracy, border patrol, terrorist attacks or drug trafficking.

We model the problem within the game-theoretic framework as a two-player zero-sum game on a graph in normal form while considering different mobility modes of the Defender. Due to huge strategy spaces of both players, we utilize iterative oracle-based algorithms to compute a Nash Equilibrium of the game: we design a set of oracles for each player, we decompose the algorithm template into two principal phases, provide an extension of the algorithm template to work with oracle hierarchies, and utilize fast suboptimal oracles without losing optimality guarantees.

We step outside the game-theoretic framework when considering multiple evading agents and propose grouping mechanism to group agents together when transiting the area, taking into account their speeds and risk aversion. We design different constraint sets and formulate a bi-objective mixed-integer linear program to compute optimal grouping.

Finally, we present a computational model of the maritime domain in the form of an agent-based simulation. We model the maritime traffic in the Indian Ocean with the merchant vessel adopting the role of the evading agent and with pirates, adopting the role of the intercepting agent. All algorithms are evaluated within the simulation framework with different space and time abstractions, with slight violation of assumptions made and with a perturbation of parameters fixed in the mathematical models. The results show robustness of formal mathematical models and their possible utilization for scenarios shifted from the original problem.

Contents

1 Introduction and Thesis Overview 1
1.1 Research Goals 3
1.2 Thesis Structure 4

2 Related Work 5
2.1 Game Theory 5
2.2 Nash Equilibrium Search Algorithm 11
2.3 Grouping Mechanisms 14
2.4 Agent-based Simulations 18

3 Models and Algorithms for Transportation Security 23
3.1 Formalization of the Problem 24
3.2 Solution approach 30
3.3 Implementation of Oracles 42
3.4 Evaluation 45
3.5 Summary 56

4 Optimization of Transit Grouping 59
4.1 Problem Description and Motivation 60
4.2 Group Transit Scheme Optimization 62
4.3 Evaluation 67
4.4 Summary 72

5 Agent-Based Model of Maritime Piracy 73
5.1 Problem Description 73
5.2 Domain Model 76
5.3 Model Calibration 88
5.4 Summary 93

6 Simulation-based Validation 95
6.1 Comparison of Models' Fidelity 95
6.2 Experiment Design 97
6.3 Statistical Parameters of the Simulation 99

6.4 Validation of Game-theoretic Transit and Grouping Mechanisms 101
6.5 Summary 110

7 Conclusions 113
7.1 Thesis Achievements 115
7.2 Selected Related Publications 117

References 119

Appendix Mathematical Programs for Oracles 127

Appendix Computed Strategies for Merchant Vessel and Pirate in the Simulation-based Evaluation 135

Chapter 1
Introduction and Thesis Overview

On thesis motivation, research questions, and outline

> *I'm a big believer that the unknown is a really wonderful thing. I know a lot of people who don't act because everything is uncertain and so they don't know where they're headed. The wonderful thing about not knowing is that anything can happen. It might be good and it might be bad, but being open to finding out is where so many great experiences come from. I think it's a good philosophy to live by.*
>
> *– Aaron Dignan*

Contemporary evolution of the world and the civilizations within steer towards interconnected infrastructures which are vital for the smooth operation and functioning of our societies. These infrastructures, being it urban transportation systems, logistics chains or computer networks, are inherently large, very complex and with many agents using, shaping, or operating the infrastructure to reach their own or common goal.

Unfortunately, antagonistic interests of different bodies from disparate societies often stir a lasting conflict mirrored as a persistent threat in the infrastructure where an intruder is trying to harm the agents, the infrastructure itself or misuse the infrastructure for his own purposes. A vital question is how to effectively protect the infrastructure or the agents from the intruder and prevent him from reaching his goals.

The aim of this thesis is (1) to contribute to a set of methods for modeling the problem, (2) to extend existing algorithms for design of policies for secure movement of agents within an infrastructure under a persistent threat of an attack, and (3) to develop a validation framework suitable for the assessment of the quality of computed policies. We are motivated by an archetypal scenario of contemporary maritime piracy: a substantial part of the Indian Ocean is now under an imminent threat of a pirate attack where thousands of transiting merchant vessels sail unprotected and have to choose a route through the area to minimize the probability of a hijack without any protection. However, techniques presented in this thesis have an overreaching application to securing sensitive targets in the urban infrastructure, efficient border patrolling and other scenarios.

The described set of conflicts can be modeled within a game-theoretic framework which was successfully used in last years for designing of protection policies in various critical infrastructures, such as airports, harbors or urban networks (Tambe, 2011). Game-theoretic framework naturally captures possible interaction of two or more agents, considering explicit models of reasoning processes of the agents as well as their utility models. Practically, all game-theoretic models guarantee the existence of an equilibrium: a fixed point in strategy spaces of all agents participating in the game; if one agent plays his equilibrium strategy, it is optimal with respect

to the type of interaction, the opponent model, and the utility model considered. The main difficulty typically lies in the design of efficient algorithms for computing the equilibrium strategy for one or more agents. We present a game-theoretic model of two rational agents representing two principal sides of the conflict, we capture their interactions and utility model using zero-sum normal form game model (Fudenberg and Tirole, 1993) and propose an extension of existing algorithms to compute equilibrium strategies for both players.

Fundamentally, due to the computational complexity of general game-theoretic models, large scale problems with many independently reasoning agents with differing utility functions are principally unsolvable (Nisan et al., 2007). If we resign on the explicit consideration of opponent's reasoning model and his utility function, we can solve large problems within classical optimization framework with an implicit opponent model in the form of a risk function and thus design optimal policies for tens of agents. We present a multi-objective optimization model of grouping of independent agents moving in the infrastructure and we use the model to compute optimal groupings of agents subject to a risk function and movement constraints.

One of the limits of game-theoretical and optimization frameworks is the rigid mathematical frame in which one has to continuously balance the richness of the model and computational scalability of the model. Typically, a problem is solved with a given set of assumptions which do not have to be satisfied and with a set of abstractions which do not hold true in reality. To tackle the limits of formal mathematical models, more expressive, albeit less formal tools can be used to validate proposed solutions computed from mathematical models. Multi-agent simulation is one of such tools, allowing a rich modeling of reasoning processes of agents, a rich representation of the environment and the dynamics of the system, all of which are finer representations of the real world situation. This approach, of course, comes with a trade-off: the simulation itself is difficult to be used as the optimization tool providing solutions.

Additional difficulties in empirical evaluation of game-theoretic solutions are inherent complications when setting up an experimental environment (if possible)—monetary, resource and time costs for field tests are huge, expert evaluation requires focus of an expert group. In both cases, the repeatability of the experiments is very limited and many human reasoning biases have to be taken into account when quantifying evaluation results. Evaluation within a simulation framework elevates all these problems and provides a reasonable compromise between costs and the quality of evaluation.

We present a computational model of the maritime domain in a form of an agent-based simulation and utilize it as a validation tool to assess the validity of proposed models, their robustness in the richer environment and the impact of assumption violation on the quality of the solution. Designed models and algorithms are implemented within the simulation framework in the form of decision-processes of the agents and evaluated with different space and time abstractions, with a slight violation of assumptions made and with a perturbation of parameters fixed in mathematical models. The results show a robustness of formal mathematical models and their possible utilization for scenarios shifted from the original problem.

1.1 Research Goals

The goals of the thesis are directly derived from the motivation described above.

1. **Formal model of the problem.**To be able to compute policies for agents operating in the infrastructure, a formal model of the domain, as well as of the conflict is needed. We capture the structure of the problem with game-theoretic framework. We will thus propose a suitable model of the environment, we will model the opposing agents as players playing a game against each other and define their utility functions. To be able to compute an optimal strategy, we will propose a suitable interaction model capturing mobility of one or both players.

2. **Scalable optimal algorithms.** Having a game-theoretic model of the problem, we will focus on finding optimal strategies for one or both players, using the standard concept of Nash Equilibrium. Due to the mobility of both agents involved, we expect the strategy space of both players to be large. We will extend oracle-based algorithms (McMahan et al., 2003) which allow us to solve huge games.

3. **Design of models with multiple players.** The model proposed above will consider explicitly only two players; however, in real-world situations, multiple units of the same side can play the game independently. Due to the scalability limits of joint strategy spaces, we will focus on other approaches and we will propose optimal grouping schemes for multiple units which are trying to minimize the interaction with the other agent, taking into account their attributes, such as speed and risk aversion.

4. **Simulation-based validation.** The solution of the game is optimal only with respect to the mathematical model it has been formulated in. It is thus necessary to validate the quality of the computed strategy on a richer model which is a more accurate representation of the real-world. We will focus on the maritime domain and we will create a multi-agent computational model of the domain. We will capture the conflict between merchant vessels transiting the Indian Ocean and pirates, roaming the high sees and trying to hijack the merchant vessels. We will compute optimal strategies for the merchant vessels and the pirates using the game-theoretic model and integrate it with the simulation. Finally, we will slightly violate some of the assumptions made, perturb parameters fixed in the formal model and observe the robustness of the computed strategies.

1.2 Thesis Structure

The text of the thesis is organized as follows:

- **Chapter 2** presents (i) an overview of related work in game theory with a focus on the most related models and algorithms used to compute Nash Equilibrium of a zero-sum two player game in the normal form, (ii) grouping mechanisms currently used for related problems and techniques used to optimize a group assembly and (iii) an introduction into agent-based simulation concepts together with the most relevant agent-based simulations.
- **Chapter 3** addresses the first and the second goal of the thesis: the problem of the secure transit is formalized within the game-theoretic framework with the accent on different constraints on agents' movement capabilities. Additionally, the chapter describes the main improvements designed to be able to solve proposed models, together with implementation details.
- **Chapter 4** addresses the third goal of the thesis: the problem of optimal grouping is motivated by real-world needs of the maritime transport and formalized for different constraint sets as a bi-objective mixed-integer linear program.
- **Chapter 5** describes the agent-based model which captures the main actors in the contemporary maritime-piracy phenomenon and briefly describes the calibration of the complete model. The model paves the way for the last goal of the thesis: the simulation-based validation.
- **Chapter 6** addresses the last goal of the thesis: all algorithms are implemented within the simulation framework and evaluated on a richer model with some assumptions violated. The focus lies on the exact replication of the game model proposed within more realistic conditions and on the observation of perturbation of parameters considered in the game-theoretic model.

We provide a brief summary at the end of each chapter highlighting the main ideas, the most important results achieved and the contributions to the state-of-the-art of the author.

Chapter 2
Related Work

The first step in making rabbit stew is catching the rabbit.

–Isaac Asimov

The chapter is divided into three main sections, reflecting the decomposition of the problem into three relatively independent branches: (i) game-theoretic models and algorithms, (ii) grouping mechanisms and combinatorial optimization, and (iii) agent-based simulations. The first section focuses on existing game-theoretic models considering similar problems to those being targeted in this thesis. Additionally, a number of algorithms has been proposed to solve game-theoretic models (the solution is typically a strategy for one or both players, which they play in a Nash Equilibrium); we describe only the most relevant — optimal algorithms for finding Nash Equilibrium in huge normal form games. The second section is focused on algorithms and models related to grouping and coalition formation. Most of the related problems are modeled within an optimization framework. We assess both of the branches and we focus on differences prohibiting recycling of existing algorithms and models. The third section summarizes current and past research done in agent-based simulations. This thesis develops an agent-based computational model of the problem which is used for the validation of solutions provided by the game-theoretic model, we thus focus on agent-based modeling methodologies and existing closely related simulation frameworks.

2.1 Game Theory

Game theory (Fudenberg and Tirole, 1993) has been applied to a wide number of problems and scenarios ranging from economics (auctions, voting, bargaining, oligopolies, social network formation etc.) through political science (public choice, fair division, war bargaining etc.) and biology (mainly evolutionary game theory) to military operations (operations research, military planning, negotiation etc.).

Given the broadness (and depth) of the related research, we will introduce here only its part: non-cooperative games between two players with antagonistic interests. In a finer focus, the categorization gets blurry. The names of the games are selected as to describe various capabilities of the players (e.g., pursuit-evasion game (Parsons, 1976)) and properties of the environment (e.g., network interdiction games (Washburn and Wood, 1995)) or the area of application (e.g., security games (Jain et al., 2010a)) and we cannot construct a clear hierarchy. However, our

research can be categorized into a narrow slice between and over game models introduced in following subsections.

Note on player naming

In different games the players are called differently. The player trying to maximize the probability of encounter is called Pursuer, Searcher, Seeker, Guard, Patroller. The player trying to minimize the probability of encounter is called Evader, Hider, Attacker or Transporter. We use different terms when describing different game models, keeping the original names of the players from respective game models.

2.1.1 Pursuit-Evasion Games

Pursuit-Evasion games are games between two players, the *Pursuer* and the *Evader*. The Pursuer, disposing with one or a group of mobile units is trying to catch one or more Evader's mobile units, which is visible to the Pursuer. This abstract formulation allows to apply these game models to a wide range of problems from civil as well as military domains. Pursuit-Evasion games are basically an extension of art-gallery problem (Chvátal, 1975) to mobile guards. Many variations exist with a limited view of guards, guards with uncertain sensors, an unknown or partially known environment etc. (more can be found in a short survey (Cheng, 2003)).

Parsons (1976) introduces a discrete pursuit-evasion formulation whereby the movement of the players is constrained by a graph. A typical example is a *cop and robber game*, where pursuers and evaders occupy nodes on a graph and can see each other. The players move in alternate turns in which they can move along an edge to an adjacent node or stay at the same node. If the players meet at the same node, the game ends. The problem is usually formulated as: how many pursuers are needed to capture the evader? To reach the solution (of this and other game variants) one has to typically find one of graph properties. Even though pursuit-evasion graphs are NP-hard on general graphs (LaPaugh, 1993), Ellis and Warren (2008) show that the solutions for grid-like graphs can be found in linear time (for example an m-by-n grid graph can be cleared by $min(m, n) + 1$ pursuers).

Other variants, such as *hunter rabbit games*—called randomized pursuit-evasion games (Adler et al., 2002)—pose different questions regarding the minimum time required for the hunter to catch the rabbit, if the rabbit moves on the same graph (restricted) or is unrestricted, i.e., it can jump to any node every turn. Vidal et al. (2002) evaluated a variant of the pursuit-evasion game on a deployed setting on a team of UAV[1] and UGV[2] in an unknown environment.

The continuous version of the pursuit-evasion game is often solved by the means of differential game models and it is thus further from our research, focusing on graph-based games. More information can be found in (Khan, 2007).

[1] Unmanned Aerial Vehicle.

[2] Unmanned Ground Vehicle.

2.1.2 Search Games

Search games are basically a variant of Pursuit-Evasion games (or vice versa), where the Pursuer has no information about the position of the Evader. Gal (1980) provides a systematic introduction (which is later extended by Alpern and Gal (2003)) to all variants of search games in a continuous space and on an arbitrary graph G. The Searcher (corresponding to the Pursuer) starts from a fixed point termed origin o and his set of all pure strategies $\mathcal{S}$ is a set of all possible paths leading from o. The Hider (corresponding to the Evader) chooses an arbitrary continuous trajectory $h \in \mathcal{H}$ as his pure hiding strategy which he is following with a maximal velocity w. In case $w = 0$, the Hider is immobile and its strategy is a single point/node, in case $w = \infty$, his movement is unconstrained and the Hider can jump to any point/node in a single turn. Gal assumes that the Searcher and the Hider cannot see one another until their distance from each other is smaller than the discovery radius r. For graphs, the discovery radius is set to $r = 0$, i.e., the players meet only at the same node. If the players meet, the game ends. A space with r and w varied in different places is called a non-homogeneous search space. A question is posed: what is the optimal strategy of the Searcher to find the Hider in minimum amount of time?

The problem is modeled as a two-person zero-sum game and a minimax strategy, i.e., a Nash Equilibrium of the game is sought to find the optimal strategy for the Searcher. The utility $u(S, H)$ represents the loss of the Searcher if the Searcher uses a strategy $S \in \mathcal{S}$ and the Hider uses strategy $H \in \mathcal{H}$.

Washburn (1981) assumes that if one player does not have any information about the position of the other player, there is no reason to choose one direction of movement over any other. This insight leads to a two-random-walks model where a detection event occurs with a known probability, when the Brownian motion of both participants brings them in proximity of each other.

Brooks et al. (2009) build on this assumption and introduce a representation of a non-homogeneous environment that is discretized into homogeneous regions, which are then represented by a node in a graph. The Searcher and the Hider then choose whether to stay in the current region/node or to move to an adjacent one. They look for a Nash Equilibrium of a game using standard linear techniques and the resulting strategy of the Searcher is expressed as Markov policy, i.e., transition probabilities between any two adjacent nodes.

Halvorson et al. (2009) define a problem of a Hider moving through the graph and a Searcher able to search c subsets of the graph nodes in each step. They model the game as a zero-sum Bayesian game. Because the strategy space of the Searcher is exponential, they use an advanced, oracle- based (i.e., column/constraint generation) approach to solve the game (see Section 2.2.2).

2.1.3 Ambush Games

Ambush games are a variation of search games, where the Guard (corresponding to the Searcher) is immobile and the Evader (corresponding to the Hider) has an additional goal: he has to cross the area from an origin to a destination (of course, not caught by the Searcher). Ruckle

introduced several variants of ambush games in continuous (Ruckle, 1981) and discrete (Ruckle et al., 1976) space and formulates solution of a zero-sum setting for some game models.

More recent Joseph's work (Joseph, 2005; Joseph and Feron, 2005) on ambush games describes a scenario of an important person or a convoy (the Evader) that can be ambushed by an immobile Guard when repeatedly transiting a dangerous area. An optimal randomized strategy of the area transit is sought for the Evader (the zero-sum formulation gives the same results as in Ruckle's work) and the network flow formulation is also successfully used for strategy space reduction. Joseph also introduces more Guards and extends the game to a multi-stage setting, where the Guard observes the Evader and can use the information gathered to place the ambush when the Evader is in the middle of the transit. He solves this problem by means of dynamic programming and decomposition on a set of relatively small sub-games. This game model is very closely related to our problem, however, we need to deal with uncertain environments and mobile guards, not only static ones.

2.1.4 Interdiction Games

Interdiction games—introduced by Washburn and Wood (1995)—-are an extension of ambush games, where the successful ambush depends not only on the requirement of the Evader to pass through the node/arc that is inspected by the Guard, however, the Guard detects the Evader only with a detection probability on the particular place as is our case. The Guard then searches the strategy maximizing the probability of detection. He proposes a zero-sum formulation and notes an exponential number of paths required to enumerate, however, using network flow techniques, he reduces the complexity of the solution to polynomial time. Even though the game is formulated as zero-sum, most of the scenarios in Washburn's work are centered on smuggling and trafficking (i.e., the strategies are sought for the Guard).

2.1.5 Infiltration Games

The infiltration games (Auger, 1991a) are another variant of search games (in more general form introduced by Gal (1980)), i.e., the Guard has no information about the position of the Evader. The Evader (as well as in the Ambush and Interdiction games) starts at an origin node O and has to go through a graph with n arcs to a destination node D (the first formulation was on graphs with just one arc (Auger, 1991b), i.e., a sequence of connected nodes (Auger, 1991b)). Every arc consists of $x < n$ nodes and connects only O and D and is not connected with any other arc. The Evader can in every step stand still or go to a neighboring node. The Guard can inspect every step one node except O and D. The Evader's aim is to reach D without being caught by the Guard within a time interval T. The capture occurs only if both players occupy the same node in the same time step, however only with a probability $1 - \lambda$ (λ is called the "miss factor" and it is analogous to the probability of detection in Interdiction games).

This problem was later extended and successfully solved by Alpern (1992) to a game on arbitrary graphs and Garnaev et al. (1997) provides a solution for cases where the Guard can inspect at most k nodes in the whole time interval T.

2.1.6 Patrolling Games

Patrolling games share similarities with a large group of search games (i.e., the Patroller is mobile and the Attacker is immobile), however, the Attacker is not present in the graph from the beginning of the game; the Attacker enters the graph (in any place/target) at any time step after the game starts. Bošanský et al. (2011) provides a comprehensive description of patrolling games: the Patroller has a very limited amount of resources available for the task, i.e., given all possible targets, the number of deployable units is significantly smaller (typically one). This means that it cannot be always guaranteed to prevent all the attacks, but the Patroller optimizes a utility based on the probability of a successful attack. As further noted by Bošanský et al. (2011), in patrolling games, the attacks are durative. They take a pre-defined period of time, during which the Patroller can interrupt the attack and save the target. The method employed by the Patroller therefore consists of repeated visits of the targets that minimize the probability that a target will not be seen for longer than the defined time period. The solution is mostly sought in the form of a Stackelberg equilibrium, creating a strategy that is efficient even if it is known to the attacker.

Agmon et al. (2008a) analyzes the problem of patrolling a perimeter, i.e., the environment is modeled as a circular graph, where each of the nodes is a potential target. The Patroller strategy is sought as a simple Markovian policy and as a policy with an additional state representing the facing of the agent in one of two directions. The crucial assumption here is made about the Attacker knowing the strategy used by the Patroller. Basically, the Attacker can wait unlimitedly long and observe the Patroller and thus infer his strategy. If we limit the Attackers knowledge, we get a game model analyzed by Agmon et al. (2008b). The perimeter patrol strategies cannot be directly applied on more general environment topologies. Arbitrary graphs are studied by Basilico et al. (2009a), where they provide a general model (termed BGA model) for finding the optimal strategy for the Patroller, which is defined as a higher-order Markovian policy. Further work extending this approach is the analysis of the impact of the Attacker's knowledge about the Patroller's policy on a general graph (Basilico et al., 2009b) and an extension of the model for multiple Patrollers (Basilico et al., 2010).

Dickerson et al. (2010) define static and dynamic asset protection problem, where the asset is either stationary, located on vertices in a graph, or mobile, following fixed and widely known route and has to be protected. They show that a random allocation of resources on a single edge cut solves this problem for static allocation, the dynamic version is shown to be NP-hard (even its approximation) and a greedy heuristics is provided and shown to work well in practice.

2.1.7 Security Games With Stationary Defenders

The previous sections introduced all game models that are relevant to security games (there is no clear distinction of what is a security game and what is not. The term was first introduced in work of Teamcore research group (Kiekintveld et al., 2009; Pita et al., 2008, e.g.) and was later re-used for various game models (Jain et al., 2010a). However, all the models correspond to problems of securing an infrastructure with limited resources). The Defender is immobile, placing limited amount of resources (i.e., units) on strategic places to prevent the Attacker from attacking one of the potential targets $T = \{t_1, t_2, \ldots, t_n\}$. There are various settings, where there are more types of Attackers (Bayesian game models), the Attacker can observe the Defender (Stackelberger game models) or not etc. In security games, in contrast with Patrolling games, once the Attacker reaches the target, the game ends, i.e., the attacks are instantaneous—the only way to prevent them is to allocate corresponding resource to protect the target prior to the attack.

Despite many differences, all the game models have a specific utility function:

- if a target t_i is attacked while t_i is **covered** by some Defender resource, the Defender gets $U_d^c(t_i)$ and the Attacker $U_a^c(t_i)$.
- if a target t_i is attacked while t_i is **uncovered** by some Defender resource, the Defender gets $U_d^u(t_i)$ and the Attacker $U_a^u(t_i)$.

We use $\Delta U_d(t_i) = U_d^c(t_i) - U_d^u(t_i)$ to denote difference between Defender's covered and uncovered utilities. Similarly, $\Delta U_a(t_i) = U_a^u(t_i) - U_a^c(t_i)$ is the difference for the Attacker. In all security games holds the following (visualized on the Figure 2.1):

$$\Delta U_d(t_i) > 0 \wedge \Delta U_a(t_i) > 0$$

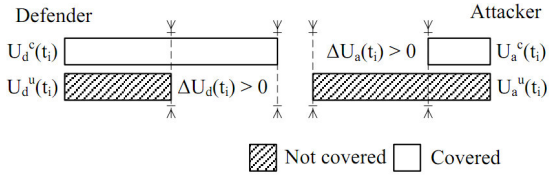


Fig. 2.1: Visualization of the Security games utility function property. Courtesy of Z. Yin (Yin et al., 2010).

For this utility function, Yin et al. (Yin et al., 2010) showed that Nash Equilibrium and the Strong Stackelberg Equilibrium (Fudenberg and Tirole, 1993) are equivalent and interchangeable. Note that this definition can be used in zero-sum setting as well.

The model of the security games was successfully deployed in the air traffic security. In one case, the model was used to propose a randomized schedules for the LAX airport (ARMOR) (Jain et al., 2010b), (Pita et al., 2009), (Pita et al., 2008) and to propose randomized schedules for federal air marshals (IRIS) (Tsai et al., 2009). Recently, the a model of Mumbai

was developed to propose a schedule for setting up police checkpoints in the city to protect important buildings (Tsai et al., 2010b), (Jain et al., 2011b).

2.2 Nash Equilibrium Search Algorithm

According to (Ferguson, 2005), we formulate our games $(\mathbf{X}, \mathbf{Y}, A)$ to be a zero-sum game of Player 1 and Player 2 with strategies $\mathbf{X}$ resp. $\mathbf{Y}$ and a real-valued function A defined on $\mathbf{X} \times \mathbf{Y}$ (creating a game matrix $\mathbf{A}$, i.e., $A(x, y)$ is a real number for every $x \in \mathbf{X}$ and $y \in \mathbf{Y}$). For this game, we seek a Nash Equilibrium (for zero-sum games equal to minmax or maxmin strategies) using one of the following techniques.

2.2.1 Linear program for Computation of a Nash Equilibrium

It is possible to find the Nash Equilibrium of the game by solving a linear programming problem constructed from the game matrix $\mathbf{A}$. Before the formulation, let us define required terms.

A mixed strategy σ_X for Player 1 is represented by a column vector $\sigma_X = (p_1, p_2, \ldots, p_m)^T$ of probabilities that add to 1. Similarly, the mixed strategy σ_Y for Player 2 is defined as a row vector of probabilities $\sigma_Y = (q_1, q_2, \ldots, q_n)$ summing to 1 ($p_i \geq 0,\ \forall p_i \in \sigma_X$ and $q_j \geq 0,\ \forall q_j \in \sigma_Y$).

Then, if both players pursue their optimal strategies, the expected outcome of the game is

$$\mathcal{V}^* = \min_{\mathbf{X}} \max_{\mathbf{Y}} \mathcal{V} = \max_{\mathbf{Y}} \min_{\mathbf{X}} \mathcal{V} \tag{2.1}$$

and the game value can be computed as

$$\mathcal{V} = \sigma_X \cdot \mathbf{A} \cdot \sigma_Y. \tag{2.2}$$

The linear problem[3] can be formulated by maximizing the value of the game

[3] We expect the reader to be familiar with Linear Programming and basic methods for solutions of linear programs, such as Simplex method, Interior point method or others.

$$\min \mathcal{V} \tag{2.3}$$

$$\mathcal{V} \geq \sum_{i=1}^{m} p_i \cdot \mathbf{A}_{i1}$$

$$\vdots \tag{2.4}$$

$$\mathcal{V} \geq \sum_{i=1}^{m} p_i \cdot \mathbf{A}_{in}$$

$$\sum_{i=1}^{m} p_i = 1 \tag{2.5}$$

$$p_i \geq 0 \qquad \forall i = 1, \ldots, m \tag{2.6}$$

where the set of constraints (2.4) are created from the game matrix $\mathbf{A}$ ($\mathbf{A}_{ij}$ is the element of the i-th row and j-th column of the game matrix $\mathbf{A}$) and equations (2.5) and (2.6) are probability constraints.

The solution of this linear program can be found using a wide range of linear programming algorithms, such as the Simplex method, the Interior point method etc. Solution of linear programs can be found in a polynomial time, however, if the number of strategies for one or both players is exponential in size of the problem (e.g., number of nodes/edges in the graph), classical methods – such as Simplex method – are not scalable in number of constraints or variables. We thus employ more advanced methods, described in following sections.

2.2.2 Oracle-based Algorithms

The requirement of enumeration of all strategies of both players prior solving the linear program is a great disadvantage. Several algorithms has been developed (mostly in Operations Research community) to tackle this problem. Basically, these algorithms start with a small, relaxed linear sub-program and then iteratively add additional information (i.e., rows or columns) about the problem. Mostly, not all original information has to be added before the termination of the algorithm, which results in a smaller version of the original problem that can be solved much faster with lesser memory requirements.

The Dantzig-Wolfe decomposition (Dantzig and Wolfe, 1960) delays the enumeration of columns (corresponding to variables $p1, p2, \ldots, p_m$) by iteratively adding only the variables that are improving the objective function. Similarly, the Benders' decomposition (Benders, 1962) iteratively adds rows (corresponding to constraints in equations (2.4)) that are currently violated. These techniques are often referenced as column or row (or constraint) generation, which are transformed into branch-and-price or branch-and-cut algorithms respectively. These methods are used to solve huge linear and integer programs (Barnhart et al., 1994b), such as crew scheduling problems (Barnhart et al., 1994a) and many others.

From the perspective of game-theory, the initial small linear sub-program corresponds to a small sub-game of the original game, the columns of the linear program (the variables) cor-

Algorithm 1 Single-Oracle Algorithm

$\hat{\mathbf{Y}} \leftarrow \{\text{arbitrary strategy } y \in \mathbf{Y}\}$, break $\leftarrow$ **false**
$\sigma_X^{(0)} \leftarrow$ uniform distribution over all $\mathbf{X}$
repeat
  $(\sigma_X^{(k)}, \sigma_Y^{(k)}) \leftarrow$ compute NE using LP for $\sigma_X^{(k-1)}$ and $\hat{\mathbf{Y}}$
  $y \leftarrow \Omega_Y^*(\sigma_X^{(k)})$
  if $y \notin \mathbf{Y}$ **then**
    $\hat{\mathbf{Y}} \leftarrow \hat{\mathbf{Y}} \cup \{y\}$
  else
    break $\leftarrow$ **true**
  end if
until not break

respond to strategies of Player 1 and rows (constraints) correspond to strategies of Player 2. This means that we need to iteratively select strategies of respective players and gradually add them to the current linear sub-program. These algorithms are termed in the multi-agent community oracle-based algorithms. If we generate only columns or only rows (i.e., strategies for only one player while enumerating the complete strategy set for the second player), the algorithm is referred as a single-oracle algorithm (McMahan et al., 2003). If we generate columns *as well as* rows (i.e., strategies for both players), the algorithm is referred as a double-oracle algorithm (McMahan et al., 2003).

Following subsections describe oracle-based algorithms in greater depth, answering posed questions and uncovering their limitations.

2.2.2.1 Single-oracle Algorithm

Let us consider a two-player normal-form zero-sum game Γ with pure strategy sets $\mathbf{X}$ and $\mathbf{Y}$ for Player 1 and Player 2, respectively, and the corresponding mixed strategy sets Σ_X and Σ_Y. The single-oracle algorithm iteratively constructs a sequence of sub-games $[\Gamma^{(0)}, \Gamma^{(1)}, \ldots]$ where each game $\Gamma^{(k)}$ consists of the complete pure strategy set $\mathbf{X}$ for Player 1 but only a subset $\hat{\mathbf{Y}}^{(k)} \subseteq \mathbf{Y}$ of the full pure strategy set $\mathbf{Y}$ for Player 2. In each iteration of the single-oracle algorithm, a Nash equilibrium $(\sigma_X^{(k)}, \sigma_Y^{(k)}) \in \Sigma_X^{(k)} \times \Sigma_Y^{(k)}$ of the current sub-game $\Gamma^{(k)}$ is first sought. Next, a Player's 2 *best-response oracle* $\Omega_Y^* : \Sigma_X^{(k)} \mapsto \mathbf{Y}$ is consulted to obtain a pure best-response strategy $y \in \mathbf{Y}$ of Player 2 against the Player's 1 strategy $\sigma_X^{(k)} \in \Sigma_X^{(k)}$; note that $oracle_Y^*$ seeks the best response *in the full game*, i.e., considering all pure strategies $y \in \mathbf{Y}$. If the resulting pure strategy $y \in \mathbf{Y}$ is already in $\hat{\mathbf{Y}}$, the algorithm terminates and the NE $(\sigma_X^{(k)}, \sigma_Y^{(k)})$ is the NE of the full game Γ; otherwise, the strategy y, now termed *sub-game expanding strategy*, is added to $\hat{\mathbf{Y}}$ and the algorithm continues. The Algorithm 1 depicts the pseudocode for the single-oracle algorithm.

In the ideal case, the iterative oracle-based algorithm would only add such pure strategies $y \in \mathbf{Y}$ to the pure strategy subset $\hat{\mathbf{Y}}$ that are in the *support*[4] of the Player's 2 resulting mixed NE strategy of the full game Γ. Best response calculation by the oracle Ω_Y^* can be viewed as a heuristic for selecting such sub-game expanding pure strategies $y \in \mathbf{Y}$ that the algorithm

[4] The *support* of a mixed strategy σ_i is a set of pure strategies $\{s_i | \sigma_i(s_i) > 0\}$.

Algorithm 2 Double-Oracle Algorithm

```
break ← false
X̂ ← {arbitrary strategy x ∈ X}
Ŷ ← {arbitrary strategy y ∈ Y}
repeat
  (σ_X^(k), σ_Y^(k)) ← compute NE using LP for X̂ and Ŷ
  x ← Ω*_X(σ_Y^(k))
  y ← Ω*_Y(σ_X^(k))
  if (x ∈ X̂) AND (y ∈ X̂) then
    break ← true
  else
    if x ∉ X̂ then
      X̂ ← X̂ ∪ {x}
    end if
    if y ∉ Ŷ then
      Ŷ ← Ŷ ∪ {y}
    end if
  end if
until not break
```

terminates after as few iterations as possible. The proof of convergence and correctness of this approach are immediate from the corresponding proofs for Bender's decomposition (Benders, 1962).

2.2.2.2 Double-oracle Algorithm

The double-oracle algorithm (Algorithm 2) is a direct extension of the single oracle algorithm. It uses incrementally expanded strategy subsets $\hat{\mathbf{X}}$ and $\hat{\mathbf{Y}}$ for both players.

As in the case of the single-oracle algorithms, the oracles Ω_X^* and Ω_Y^* provide best responses to the current mixed strategies $\sigma_Y^{(k)}$, $\sigma_X^{(k)}$ respectively and these best response strategies are added to the current sub-game. Termination condition requires that best responses for both players are already present in the respective strategy subsets. The proof of convergence to a Nash equilibrium can be found in (McMahan et al., 2003).

The main requirement for the oracle-based algorithms is fast oracles that can provide the response in constant, linear or polynomial time. If the computations of the best response problems are NP-hard, the scalability and performance of these algorithms are significantly decreased. However, it is still possible to find solution (optimal or approximate) even for these problems (Halvorson et al., 2009).

2.3 Grouping Mechanisms

Related work on this subject can be roughly divided into three parts: first, we describe work related to the dynamic group transit optimization; second, we describe state-of-the-art methods for optimization multi-objective mixed integer program; and third, we describe limited work related to the currently deployed fixed GTS deployed in the Gulf of Aden.

2.3.1 Convoy Movement Problems

Most related work from the domain and method perspective has been done in Convoy Formation, Routing, and Movement Problems (Kumar and Narendran, 2010) that involve routing and scheduling military or emergency rescue convoys within strategic constraints. Montana et al. (1999) solve a typical convoy moving problem: they minimize total movement time of convoys moving in a directed graph, subject to a set of following constraints: the convoys do not stop en-route (same requirement), they do not cross each other, they have the same speed and others, less relevant to our problem. We need to allow groups to have different speeds and to cross each other during the transit. Montana et al. additionally focus on convoy scheduling, i.e., the grouping of trucks into one convoy, posing constraints on the size of the group (similar to our formulation) and what type of load the trucks transport (which is irrelevant in our problem).

Typical variants of the convoy movement problem are considered to be NP-hard (Goldstein et al., 2010) and operation research techniques, such as mathematical modeling, are typically used to find a solution. Montana et al. (1999) use genetic algorithms to find optimum convoy schedules. Chardaire et al. (2005) model the problem as an integer program; they solve large-scale instances by using Lagrangian relaxation and evaluation of the dual function and obtain heuristic solutions for the original formulation. For the problem defined in this thesis, integer programing is suitable to capture the structure of the problem, however, we are interested in optimal solutions. Finally, Kumar et al. (2009) address a bi-criteria version of the convoy movement problem with minimizing total travel time and travel span as objectives. We would need to capture our objectives as a bi-criterion function as well, however, we will look at travel time and risk taken and use different solution approach.

2.3.2 Convoy Driving on Highways

We seek inspiration in the research of related problems in the transportation domain. Convoy driving on public highways is a similar convoy movement in the military domain. Previous work is motivated mainly by the military or emergency rescue domain. We can find similar work in classical transportation domain as well where similar convoy driving problem arises on public highways. Khan and Boloni (2005) formalize the problem of vehicles joining and leaving a convoy while having an upper and lower speed limit and acceptable utility for being in a convoy. The formulation is similar to our problem; however, the work does not consider any temporal constraints. Algorithms are based on coalition formation with non-transferable utility techniques and solutions reflect optimum decision from the vehicle's point of view, not from the social welfare maximization perspective.

One of the frequently solved problems in urban transportation is car pooling (Baldacci et al., 2004) problem consisting of defining subsets of passengers that will share cars and the paths the cars should follow, so that number of passengers per car is maximized and the sum of the path costs is minimized. The goal is to plan a set of minimum cost vehicle routes capable of serving as many passengers as possible, under a set of constraints arising from the spatial distribution of the problem. The special case of the carpooling problem with all cars being identical is called

a Dial-a-Ride Problem (Cordeau and Laporte, 2003). Both problems can be solved heuristically or exactly using integer programming techniques. Again, even though we plan on a primitive graph, the methods here cannot be directly reused, as we pose different constraints on the groups and we cannot group arbitrary agents into a single group.

2.3.3 Grouping and Clustering

Another body of related work examines the Grouped Sweeping Scheduling (GSS) problem (Chen et al., 1993; Yu et al., 1993). In GSS—frequently used in multimedia storage management—the problem is to minimize buffer space in retrieval of heterogeneous multimedia streams by dividing set of streams into groups, subject to a set of constraints posed by the physical architecture of the disk, which differ from our problem which has to capture the spatial distribution of agents and their differing attributes (such as speed).

On-line clustering is another approach to group entities arriving in time and it is employed across a number of domains. Typically, a data stream is clustered into a number of clusters which arise in time. The clusters are represented as K-Means (Beringer and Hüllermeier, 2006) or K-Medians (Guha et al., 2000) or as more advanced, density-based models (Chen and Tu, 2007). There is no proper definition of optimality for such techniques as they are designed for domains with huge amount of arriving entities and another qualities (such ability of trend capturing, memory efficiency etc.) are aimed for. In our case, the number of entities is not that large (typically tens or hundreds) and we pose more complex constraints on samples from a single cluster and on the clusters themselves. Additionally, optimality of the solution cannot be simply verified, which is, with respect to the size of our problem, unnecessary limiting.

Finally, a canonical problem of spatio-temporal collaboration *children in the rectangular forest* is proposed by Luo and Boloni (Luo and Bölöni, 2007). Here, two agents negotiate, how to cross a rectangular dangerous area, possibly together and which route to take through the area. In our case, the problem contains multiple agents and the route is given, however, time of the crossing and speed of the transit matters.

2.3.4 Mathematical Programming and Multi-Objective Optimization

Multi-objective optimization (MOP) is a frequently used set of techniques having roots in the work of Edgeworth and Pareto in economics (Edgeworth, 1881; Pareto, 1964). Talbi (2009) explains that the optimal solution for MOPs is not a single solution as for mono-objective optimization problems, but a set of solutions defined as Pareto optimal solutions. A solution is Pareto optimal if it is not possible to improve a given objective without deteriorating at least another objective. This set of solutions represents the compromise solutions between the different conflicting objectives. The main goal of the resolution of a multi-objective problem is to obtain the Pareto optimal set and, consequently, the Pareto front. The difficulty in solving MOPs lies in the following general facts Talbi (2009):

- There are no commonly used definitions on the global optimality of a solution as in mono-objective optimization. The order relation between solutions of a MOP problem is partial, and the final choice depends on the decision maker.
- The number of Pareto optimal solutions increases according to the size of the problem and mainly with the number of objectives being considered. Indeed, at least all Pareto solutions of an n-objective problem are necessary Pareto solutions of the same problem with an additional objective function.
- The structure of the Pareto front (e.g., continuity, convexity, multimodality) depends on the studied MOP. For instance, the Pareto optimal solutions may be localized on the frontier and inside the convex hull of feasible solutions. Moreover, most of the MOPs are NP-hard problems.

2.3.4.1 Formal Framework

Definition 1. Multi-objective optimization problem is defined as (Talbi, 2009)

$$MOP = \begin{cases} \min F(x) = (f_1(x), f_2(x), \ldots, f_n(x)) \\ \text{s.t.} \quad x \in S \end{cases} \tag{2.7}$$

$$\tag{2.8}$$

where n ($n \geq 2$) is the number of objectives, $x = (x_1, \ldots, x_k)$ is the vector representing the decision variables and S represents the set of feasible solutions associated with equality and inequality constraints and explicit bounds. $F(x) = (f_1(x), f_2(x), \ldots, f_n(x))$ is the vector of objectives to be optimized.

The search space S represents the *decision space* or parameter space of the MOP. The space in which the objective vector belongs to is called the *objective space*. The vector F can be defined as a cost function from the decision space in the objective space that evaluates the quality of each solution $(x_1, \ldots, x_k)$ by assigning an objective vector $(y_1, \ldots, y_n)$, which represents the quality of the solution. A partial order relation could be defined, known as dominance relation.

Definition 2. Pareto dominance. An objective vector $u = (u_1, \ldots, u_n)$ is said to dominate $v = (v_1, \ldots, v_n)$ (denoted by $u \leq v$) if and only if no component of v is smaller than the corresponding component of u and at least one component of u is strictly smaller.

A Pareto optimal solution denotes that it is impossible to find a solution that improves the performances on a criterion without decreasing the quality of at least another criterion. A MOP may have a set of solutions known as the Pareto optimal set. The image of this set in the objective space is denoted as the Pareto front.

Definition 3. Pareto optimal set. For a given MOP (F, S), the Pareto optimal set is defined as $P^* = \{x \in S / \nexists x' \in S, F(x') \leq F(x)\}$.

Definition 4. Pareto front. For a given MOP (F, S) and its Pareto optimal set P^*, the Pareto front is defined as PF $= \{F(x), x \in P^*\}$.

2.3.4.2 Solution Approach

We introduce only a single method out of many in detail, the reader can find other methods, such as *weighted metrics* (Talbi, 2009), *goal programming*(Charnes et al., 1955), *achievement functions* (Wierzbicki, 1980), *goal attainment* (Talbi, 2009), ϵ-*constraint method* (Haimes et al., 1971) or a range of metaheuristics (Talbi, 2009) in respective references.

Aggregation Method

The aggregation (or weighted) method is one of the first and most used methods for the generation of Pareto optimal solutions. It consists in using an aggregation function to transform a MOP into a mono-objective problem (MOP_λ) by combining the various objective functions f_i into a single objective function f generally in a linear way (Hwang et al., 1979):

$$f(x) = \sum_{i=1}^{n} \lambda_i f_i(x), \quad x \in S \tag{2.9}$$

where the weights $\lambda_i \in [0, \ldots, 1]$ and $\sum_{i=1}^{n} \lambda_i = 1$. The solution of the weighted problem is weakly Pareto optimal. The final solution is Pareto optimal if $\lambda_i > 0$ for all $i \in [1, n]$ or the solution is unique. The main drawback of this method is that it generates only supported solution, i.e., solutions on the convex set of the Pareto front.

2.3.5 Group Transit Schemes

Grouping mechanism relevant to our problem set is currently deployed in the International Recommended Transit Corridor in the Gulf of Aden, where the transiting merchant vessels are aligned into a corridor and follow a prescribed schedule which assigns to the arriving vessels five distinct speed levels an arrival time at the beginning of the corridor. This group transit scheme, together with the description of the International Recommended Transit Corridor, is well explained by Intertanko (Intertanko, 2009); however, the computation of the currently used times and speeds for groups is not described—to our best knowledge—in any of the public sources. In (Hrstka and Vaněk, 2011), we derived formal model of the problem and proposed a set of algorithms able to compute optimal schedules, however, the approach was not scalable even though the problem is solvable in polynomial time. We have extended this work by proposing more compact formal model and we design new scalable algorithm able to compute optimal fixed schedules for tens of groups in seconds (Vaněk et al., 2013a,b).

2.4 Agent-based Simulations

The use of agent-based or simulation-based models to support policy design and operational management has a very long-standing tradition in the transportation field. The use of simulation as a validation or solution quality evaluation tool is relatively novel and not fully explored

research branch. In this section, we first look at agent-based simulations themselves: multi-agent simulation design and internals. Then we provide overview of the most relevant simulations (focusing on transportation domain, infrastructure security and maritime piracy).

2.4.1 Multi-Agent Simulation Concept

Multi-Agent Systems (MAS) provide a description framework that is appropriate for many real world systems consisting of a set of interacting autonomously deciding actor. Human and animal societies form prominent and intuitive examples for real-world multi-agent systems. Klügl (2009) provides very detailed insight into design and "engineering" of the multi-agent simulations. In every MAS there are four aspects that are relevant for capturing the notion of multi-agent systems and agent-based software: The *Agents* forming an *Multi-Agent System* based on their *Interactions* and situated in an *Environment*.

2.4.1.1 Agent and Multi-agent Systems

Franklin and Graesser (1997) define an autonomous agent as follows:

Definition 5. Autonomous Agent. An autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future.

To understand the concept of an agent, it is necessary to discuss the essential properties associated with it more deeply (Klügl, 2009):

1. **Situatedness** An Agent is situated in the environment,i.e., there is an ongoing interaction between the agent and its surroundings through sensors and effectors and having form of manipulation of parts of the environment, environment perceiving or message communication. The interaction continues over time implying persistence of the agent in the environment.
2. **Autonomy** An Agent is autonomous in execution of its actions and fulfilling its goals, i.e., next actions are determined without direct influence from the outside.
3. **Pro-activity and Reactivity** An Agent should be pro-active in the environment to attain its goals and react on percepts from the environment in form of adaptation or learning.
4. **Level of Rationality** An Agent is working towards its goals in a non-random way, typically selecting actions with maximum (or satisfiable) expected outcome with respect to its goals.
5. **Sociality** An Agent is able to interact with other agents by, e.g., communication channels or through the environment.
6. **High-Level Description** An Agent's behavior should be describable by a high-level framework, such as Belief, Desire,Intention (BDI) (Rao et al., 1995) architecture or other anthropomorphic ways of describing agent structure and behavior.

The notion of a multi-agent system is easily given when the term agent is clear: A multi-agent system is a system that consists of interacting agents with the aim of fulfilling some common or individual goals (Klügl, 2009).

2.4.1.2 Environment and Interactions

In general, the environment is everything the agents interact with, more formally (inspired by FIPA framework[5]), following components are typically found in the environment: (1) mediators, facilitators, page servers and other service providers; (2) general infrastructure, such as transport information and physical transportation bodies (e.g., a bus or a vessel) and (3) resources, such as information sources, data sources.

Typically, the environment is classified into one of following classes: accessible vs. in-accessible, dynamic vs. static, stochastic vs. deterministic and continuous vs discrete (Russell et al., 2010).

One of the central aspects of MAS is the way agents interact which could be distinguished according to frequency, persistence, level and media, variability and goal of interaction (Klügl, 2009).

2.4.1.3 Design of Multi-agent Simulations

As described in Sislak et al. (2009), there are two approaches to agent-based simulations design – *analytic simulation* and *distributed virtual environments* (DVE). The first approach is used when we need maximum detail and accuracy without any human interaction. The simulation is required to be deterministic and is thus usually synchronous. The DVE simulations are used to create an illusion of a real-world environment for training or human user purposes. Thus they do not usually support an absolute synchronization and often use a human-in-a-loop for the simulation control.

Following categories are considered when classifying a multi-agent simulation (Klügl, 2009): time advance paradigm, granularity, goal of the simulation. In temporal category, we can consider following three forms: *continuous simulation time*, *event based simulation* and *time-stepped simulations*. The continuous time are models consisting of differential equations; the event-based simulations use events as a basis for it execution – each event is tagged with a time stamp and sorted into an event-queue; the simulation then triggers events with the earliest time stamp to determine new states of the agents and the environment to advance the simulation. The time-step based simulation advanced in predefined virtual time steps and new states of the environment and agents are computed after each step.

The granularity of the simulation typically determines the level of detail of the model: two basic forms are micro and macro models. Micro models consist of small entities with separate state and behavior and the overall simulation behavior is generated as a combination of the behaviors of the single parts. Macro models conceptualizes the system as a single entity with a number of state variables and parameters. The state is updated and an output is produced based on the inputs into the simulation. This approach build on the assumption of homogeneity of its parts and space.

The goal of the simulation determines the class of the model and simulation properties with two possible values: *explanation* and *prediction*. A much greater level of realism is required for the latter and it is much harder to achieve. The requirements for explanation of a phenomenon

[5] `www.fipa.org`

are lower, although still challenging. In both cases, proper validation of simulation correctness is required.

Given the complexity of any multi-agent simulation, we focus on those which are easy to extend or which can simulate directly a conflict between two adversaries. Moreover, many of such simulations are designed for military purposes and are thus not publicly available. In following section, we present the most suitable simulation from various domains, which are publicly available.

2.4.2 Related Agent-based Simulations

The simulation framework presented in this thesis was developed in context of another two simulations — Agentpolis and Tactical AgentScout.

Agentpolis[6] (Jakob et al., 2012a) is a fully agent-based platform for modeling multi-modal transportation systems. It comprises a high-performance discrete-event simulation core, a cohesive set of high-level abstractions for building extensible agent-based models and a library of predefined components frequently used in transportation and mobility models. Together with a suite of supporting tools, AgentPolis enables rapid prototyping and execution of data-driven simulations of a wide range of mobility and transportation phenomena.

AgentScout[7] project (Vokrinek et al., 2010) provides a rich simulation platform based on ALite (Komenda et al., 2013), an agent-based simulation toolkit of an urban area where an operation of military forces takes place. The military agents share a common goal and operate in an unknown environment which is abstracted as a graph. There are adversary agents present in this environment aiming for disruption of plans of the military agents.

In transportation domain, the vast majority of the work, focuses on ground transportation (e.g. Boel and Mihaylova, 2006; Seow and Lee, 2010) and, to a lesser extent, on air transportation (e.g. Sislak et al., 2011).

In the maritime domain, applications of similar models are surprisingly scarce. Existing work either focuses on traffic in ports and national, coastal waters (Hasegawa et al., 2004; Kose et al., 2003) or uses high-level equation-based models (Bourdon et al., 2007) unfit for capturing individual-level behavior and inter-vessel interactions essential for modeling maritime piracy. Furthermore, none of the above models is concerned with the security of maritime shipping lanes.

As far as the security angle on transportation systems is concerned, existing simulations focus on modeling activities in and around terminals rather than within transportation networks themselves. This is true both for airport security (Chawdhry, 2009; Wilson, 2005) and port security (Koch, 2007). The spatial, network aspect of transportation security has been touched upon in the work on modeling critical infrastructures (Barton and Stamber, 2000), however, the emphasis there is mostly on other than transportation types of infrastructures. The problem of securing transportation infrastructures and logistical networks has only been studied in the military context (Ghanmi et al., 2011).

[6] `http://merle.felk.cvut.cz/redmine/projects/agentpolis`

[7] `http://agents.felk.cvut.cz/projects/agentscout/`

Focusing on the very phenomenon of maritime piracy, existing work is concentrated primarily in the fields of security studies, international relations and global policy (Onuoha, 2010). Only recently, initial attempts at applying computational modeling and optimization to maritime piracy have emerged but focus exclusively on military aspects of the problem: Bruzzone et al. (2011) model piracy around the Gulf of Aden using the discrete-event simulator PANOPEA. The authors focus on evaluating the efficiency and effectiveness of different Command and Control models; only main actors in the Gulf of Aden are considered and the simulation is not scaled to the Indian Ocean where the merchant traffic model is significantly more complicated.

Tsilis (2011) employs the MANA agent-based modeling framework (Lauren and Stephen, 2002) to identify key factors affecting the escort of vulnerable merchant vessels through the Gulf of Aden. The escorting scenario is modeled on a tactical level, focusing on positioning of individual ships and protection of one group of merchant vessels; this is different from our model which adopts a whole-system perspective and considers the security of maritime transportation system as a whole. The MANA framework is also used by Decraene et al. (2010) to analyze requirements on non-lethal deterrents for defending large merchant vessels against pirate attacks; again, the focus is on the tactical level of modeling a single encounter in detail, rather than the system as a whole.

Slootmaker (2011) describes *Next-generation Piracy Performance Surface (PPSN)* model which employs meteorological forecasts, intelligence reports and historical pirate incidents to predict areas conductive to pirate activity around the Horn of Africa. Hansen et al. (2011) further improve the PPSN model by refining the environment model and adding a probabilistic behavioral pirate model, resulting into the *Pirate Attack Risk Surface (PARS)* model. Both PPSN and PARS models are numerical with only a minor simulation component and are limited to short-term forecasts (several days). They do not directly model real-world behavior and interactions of individual vessels; consequently, their applicability for what-if type of analysis is limited.

Finally, piracy patterns and the effect of countermeasures were also studied using statistical data analysis and data mining (Bowden et al., 2011). The usability of such results for policy design and optimization is limited because the insights gained concern the behavior of the maritime system under current circumstances and are difficult to extrapolate to hypothetical future scenarios.

Chapter 3
Models and Algorithms for Transportation Security

On the design of models capturing the interaction of a transit agent and an intruder with various mobility capabilities.

Essentially, all models are wrong, but some are useful.

– George Box

The strategic interaction between agents moving throughout the transportation infrastructure and an agent trying to harm these transiting agents can be best modeled within the game-theoretic framework. In this chapter, we propose a number of formal models of *transit games*, where the attacking agent is subject to a different sets of constraints on his movement[1]. We consider different utility models which capture more or less precisely the interaction—exact and approximate utilities, expressing correct and approximate probability of encounter respectively. Depending on the utility model, the algorithms design in the next section to find Nash Equilibrium (which is considered solution of the game) are more or less complex and harder or easier to solve.

The algorithms are inspired by current work of McMahan et al. (2003) which introduces iterative algorithms for the computation of Nash Equilibrium using oracles — modules providing the best-response for one player for a given *sub-game* — the game have only a small subset of strategies for both players (the algorithm is described in detail in Section 2.2.2). The algorithm iteratively solves the sub-game and adds the best responses, until no oracle can provide a best response which is not already present in the current sub-game. The solution of the final sub-game is guaranteed to have the same Nash Equilibrium as the complete game. The oracles thus stand at the core of the algorithm and their design is a crucial element in the utilization of this algorithm. Here, a number of oracles is designed to reflect different mobility capabilities and different utility models of the attacking agent.

We also extend the *double-oracle algorithm* by considering additional, sub-optimal oracles providing only *better* responses, thus speeding up the solution. Finally, the implementation of a subset of oracles is described — due to the inherent non-linearity of the problem, we move outside of the typical mixed-integer programming toolkits and utilize classical heuristic algorithms such as A* and a variants of Branch&Bound algorithms.

[1] some of these games are already considered in existing works, we describe them for completeness of the model with appropriate reference.

3.1 Formalization of the Problem

Typical scenarios of an asymmetric conflict in critical transportation infrastructures (Brown et al., 2006) contain two opposing sides, i.e., *players*, having typically antagonistic goals. We focus on area transit and control scenarios where a transiting agent is trying to transit an area and reach its destination while being undetected by a second agent, trying to detect the transiting agent by patrolling in the area.

This general problem can model a wide range of situations, from our perspective, we can be interested in the computation of optimal strategies in following cases:

- **Protection of urban transportation network**, where we are interested in designing an optimal protection strategy to prevent an attacker from reaching sensitive places, such as hospitals, embassies, government buildings or water treatment sources (Jain et al., 2011a).
- **Protection of borders**, where we are interested in designing an optimal border patrol strategy to intercept an unauthorized person crossing the borders.
- **Hostile area transit**, where we are interested in designing a transit scheme to deliver humanitarian or medical help through a hostile area controlled by an adversary.
- **Maritime transit of pirate infested areas**, where we are interested in the design of randomized shipping routes minimizing the probability of merchant vessel hijack by a pirate.

Independently on the scenarios considered above, we will further term the intercepting player *Defender* and the evading player *Evader*, as stated in the Introduction Chapter.

As a large volume of work has already been done in this line of research (see Section 2.1), we focus on a subset of scenarios, where both players are mobile, i.e., both players are able to move through the area/infrastructure. We enrich the model to account for uncertain interceptions, i.e., we introduce uncertainty about the detection of the other player. Finally, it is important to denote that we do not explicitly consider different payoffs for different destinations reachable by the evading player, although it would be trivial to extend the model to account for them. The complexity of the model lies primarily in the richness of interaction of mobile players and in the uncertainty of interception.

3.1.1 Transit Area Representation

In scenarios described above, the environment where both payers interact, is either a graph representing an infrastructure (such as the urban network) or a homogeneous area (such as ocean, forest, desert etc.), where the important parameter is the line of sight. We directly discretize such homogeneous environment to regular grids which we represent as graphs as well.

The graph is directed (allowing cycles and bi-directional edges), possibly with loops. We term the graph *transit graph* $G(N, E)$, where $N = \{1, 2, \ldots, n\}$ denotes a set of nodes represented directly by natural numbers, and $E = \{(i_1, j_1), (i_2, j_2), \ldots, (i_m, j_m)\}$ where $i_k \in N \wedge j_k \in N$ is the set of edges that define a legal movement of players through the transit area/infrastructure. Every edge has a unit length. Two special types of nodes are defined on the transit graph:

- *origin nodes* N_{orig} – the Evader starts its path in any node of this type;

- *destination nodes* N_{dest} – the Evader aims to reach any node of this type;

Uncertain encounters

To provide more expressiveness to the game model, encounters are assumed to lead to interceptions only with a defined location-specific *interception probability* $\rho_l \forall l \in N \cap E$. Interception probability may e.g., reflect the likelihood that the Defender will be able to detect the Evader and/or will be able to physically apprehend the Evader in a given location. The concept is similar to the concept of sensors with a probability of detection explored by Brooks et al. (2009).

For certain encounters, some of the models can be solved using known approaches (summarized in Section 2.1). However, for both mobile players, even the certain encounters scenario is nontrivial and is a subject of this thesis.

3.1.2 Evader Model

The Evader model captures the fact that the Evader aims to transit the area/infrastructure unintercepted from the origin to the destination. The set S_E of all possible pure Evader's strategies is then the set of all walks starting in any origin node and ending in any destination node, with the nodes in between not being an entry or exit node, i.e.,

$$S_E = \{[n_0, \ldots, n_m] | n_0 \in N_{\text{in}} \wedge n_m \in N_{\text{out}} \wedge n_i \in N \setminus \{N_{\text{in}} \cup N_{\text{out}}\} \wedge (n_i, n_{i+1}) \in E\} \forall i = \{1, \ldots, m-1\}\} \quad (3.1)$$

Note that because in general Evader's walks are unlimited, the above set can be infinite. However, as observed by Joseph (2005) the Evader cannot get anything by having cycles in his strategies. We thus consider only paths without the cycles, having the strategy set finite.

3.1.3 Defender Model

The Defender model is slightly more complicated. Typically, the Defender is considered *stateless*, i.e., just being present at some places without any explanation or limitation of the possibility of reaching such places (Joseph, 2005; Jain et al., 2011a; Tsai et al., 2010a,b).

In our model, we explore this variant with uncertain encounters to complement related work, however, our main goal is to propose a model of a mobile Defender having a base, from which he conducts periodical walks throughout the area/infrastructure, having limited resources. This concept captures described scenarios with higher precision, although it complicates the models. We consider three variants of the problem:

1. **fixed-base Defender**, having a fixed position of the base in the area. This model corresponds to pirates starting in harbors, police forces starting every morning in the police station.

2. **mobile-base Defender**, having the ability of moving the base in between the explorations. This model corresponds to pirates with a mobile mothership and a small speed boat or to border patrols with a helicopter moving over forest areas, in which they conduct patrols.
3. **unconstrained Defender**, where we do not pose any constraints on the movement of the Defender, i.e., in every step, he can choose any node/edge to observe. This model corresponds to a satellite-based surveillance where in each step, the satellite can be focused on a single area. In this case, the Defender is not constrained by the base and the strategy allows the highest degree of movement.

3.1.3.1 Static Defender

The static Defender represents a case where static allocations of resources in the infrastructure are considered. Typical cases are, e.g., allocation of checkpoints in the city of Mumbai (Chandran and Beitchman, 29 November 2008) to detect an adversary trying to hit sensitive buildings or an ambush scenario in which an attacker awaits a transiting convoy (Joseph, 2005).

To reflect the static nature of Defender's resources, the model allows the Static Defender to allocate its resources only on a node $n_i \in N \backslash \{N_{\text{in}} \cup N_{\text{out}}\}$.

In general, if the Defender disposes of k resources, his pure strategy is a tuple of k distinct nodes and his strategy space S_D is thus a set of all possible tuples:

$$S_D = \{n_1, \ldots, n_k \ | \forall i \forall j, i \neq j : n_i \neq n_j \wedge n_i \in N \setminus \{N_{\text{in}} \cup N_{\text{out}}\}\} \tag{3.2}$$

3.1.3.2 Fixed-base Defender

In case of a Defender with a static base, the model reflects scenarios inspired by border patrolling, maritime piracy or scheduled city patrols with a fixed position of a station.

The fixed-base Defender is—compared to the static Defender—allowed to move through the graph, however, its movement is restricted by a base—the strategy is a closed path, termed *walk*, starting and ending in the base. The path can contain nodes (including the base) and edges multiple times.

Formally, the set S_D of all possible pure Defender's strategies is the set of all closed walks starting and ending in the base with the length not exceeding L_D:

$$S_D = \{[n_0, \ldots, n_m] | m \leq L_D \wedge n_i \in N \wedge n_0 = n_m = n_b \\ \wedge n_i \in N \setminus \{N_{\text{in}} \cup N_{\text{out}}\} \wedge (n_i, n_b) \in E\} \tag{3.3}$$

Note that if L_D is so small that it does not allow the Defender to cross one of the Evader's walks, then the Evader will have a deterministic strategy that guarantees a safe transit of the graph. The threshold for L_D under which this is the case depends on the position of Defender's base and topology of the transit graph. In contrast to the Evader, who performs his walk only once, the Defender executes his walk repeatedly[2].

[2] Defender does not repeat the walk indefinitely, the game ends once the Evader reaches one of the exit nodes.

3.1.3.3 Mobile-base Defender

This Defender model is motivated by the fact that the base can be mobile, i.e., the base can change its location through time, however, in a much slower pace than the Defender (or his units). The scenario is motivated by mobile platforms for unmanned aerial vehicles or pirates with a skiff as fast mobile units and motherships as mobile bases in huge open waters areas.

The mobile-base Defender is thus not bound to any specific node. The strategy space of the mobile base Defender is enriched for the base-selection, i.e., its set of strategies is the set of all closed walks starting and ending in the same node ($n_0 = n_m$) which is not entry or exit node:

$$S_D = \{[n_0, \ldots, n_m] | m \leq L_D \wedge n_i \in N \wedge n_0 = n_m \\ \wedge n_i \in N \setminus \{N_{\text{in}} \cup N_{\text{out}}\} \wedge (n_i, n_b) \in E\} \quad (3.4)$$

3.1.3.4 Unconstrained Defender

For the case of an unconstrained Defender, we do not pose any constraints on his movement, i.e., he is without a base, he is not limited by the length of the walk and the movement does not have to be continuous, i.e., he can "jump" between places in the area. The motivation is satellite-based surveillance where we can schedule a set of places to be photographed/observed—and the places do not have to be next to each other. A similar scenario arises for a complete sensor coverage of an area (such as an oil field) with hundreds of cameras deployed on place, however there is a limited number of operators observing the streams from the cameras: the system then has to display images only from a small subset of cameras, although he can switch between them.

The unconstrained Defender is not constrained by the movement constraints which require a strategy to form a valid closed walk. The strategy space of the unconstrained Defender is a super-set of the previous mobile base Defender and its strategy is a list of nodes of size L_D:

$$S_D = \{[n_0, \ldots, n_m] | m \leq L_D \wedge n_i \in N \wedge n_0 = n_m \wedge n_i \in N \setminus \{N_{\text{in}} \cup N_{\text{out}}\}\} \quad (3.5)$$

Having the Evader's strategy set model as well as various models for the Defender mobility, we can look at the result of an interaction of these strategies in a form of a utility function.

3.1.4 Utility

Utility theory is used to express preferences of agents about the outcomes of their actions. In this work, we assume utility-theoretic assumptions, such as utility maximizing agents having a linear perception of utility value. Shoham and Leyton-Brown (2008) introduce utility through preferences and lottery and we leave an interested reader to read this book for details. In our model, we consider a utility function to have the form: $U : (s_1, s_2) \to \mathbb{R}$.

Each player has their own utility function and we denote U_D utility of the Defender and U_E the utility of the Evader. All considered game models are zero-sum, i.e., $U_D = -U_E$ we thus introduce only the Defender's utility with Evader's utility being negative.

Interaction of Two Mobile Players

The movement of both players happens simultaneously in synchronized *steps*. During each step, a player can move to an adjacent node or stay in the same node. Both players have the same movement speed and all edges take a single step to traverse. Player's movement through the transit graph can be unambiguously represented by a node *walk*, i.e., a sequence of nodes $w = [n_0, n_1, \ldots, n_k]$[3]; we then denote $|w|$ the length of walk w, and $w[j]$ the j-th node on the walk (for $0 \leq j \leq |w| - 1$).

The following will be required for the definition of utilities. For any finite walk w, we define *infinite walk repetition* $w^{\infty}[i] = w[i \bmod |w|]$ and *shifted infinite walk repetition* $w^{\infty \triangleright m} = w[(i - m) \bmod |w|]$. E.g., for $w = [1, 4, 7]$, we have

index	...	[−2]	[−1]	[0]	[1]	[2]	[3]	[4]	...
w	...	-	-	1	4	7	-	-	...
w^{∞}	...	4	7	1	4	7	1	4	...
$w^{\infty \triangleright 1}$	...	1	4	7	1	4	7	1	...

Encounters

We say two walks w_1 and w_2 have:

- a *node encounter* at node $i \in N$ at step t if $i = w_1[t] = w_2[t]$ (being at the same node at the same time step);
- an *edge encounter* at edge $(i, j) \in E$ at step t if $i = w_1[t] = w_2[t] \wedge j = w_1[t + 1] = w_2[t + 1]$ (traveling the same edge simultaneously in the same direction) or $i = w_1[t] = w_2[t + 1] \wedge j = w_1[t + 1] = w_2[t]$ (traveling the same edge simultaneously in the opposite directions)
- an *encounter* at *location* $l \in N \cup E$ at step t if w_1 and w_2 either have a node encounter or an edge encounter at l at step t.

The *encounter sequence* of two walks w_1 and w_2 is a sequence $[(l_0, t_0), (l_1, t_1), \ldots, (l_n, t_n)]$ where $\forall i \in \{0 \ldots n\}$ w_1 and w_2 have an encounter at l_i at step t_i and $t_i \leq t_{i+1}$, i.e., the order in which the encounter locations appear on walks w_1 and w_2 is preserved; the *encounter location sequence*, denoted as $w_1 \cap w_2$, is then the encounter sequence without timestep indices but with the ordering preserved – $[l_0, l_1, \ldots, l_n]$.

[3] In a slight abuse of common mathematical notation and to emphasize similarity with the array data structure, we use brackets to denote sequences and to index sequence items.

Utility for Static Defender and Mobile Evader

In case of the static Defender and the mobile Evader, the Defender's walk is a single location[4] while the Evader's walk is a sequence of nodes. Having uncertain encounters, i.e., the encounter probability for each location ρ_l, the Defender's utility is computed as a probability of encountering the Evader:

$$U_D = 1 - \prod_{l \in s_E \cap s_D} (1 - \rho_l) \tag{3.6}$$

where $(1 - \rho_l)$ is the probability of the Evader being undetected on location l with Defender present.

Utility for Mobile Defender and Mobile Evader

For the mobile Defender (considering a fixed-base, a mobile-base and unconstrained models) the utility model is constructed by the following reasoning process.

First, given an encounter location sequence I of walks with a specific shift we can express the probability $\pi(I)$ that the Evader will be intercepted by the Defender as

$$\pi(I) = \sum_{i=0}^{|I|-1} p(I[i]) \prod_{j=0}^{i-1} (1 - p(I[j])) \tag{3.7}$$

where

$$p(I[i]) \prod_{j=0}^{i-1} (1 - p(I[j])) \tag{3.8}$$

is the probability that the Defender will not intercept the Evader at locations $I[0], \ldots, I[i-1]$ and will intercept it at location $I[i]$.

To calculate the interception probability $\pi(s_E, s_D)$ of a pair of pure strategies (s_E, s_D), we need to determine all possible encounter location sequences that can result from executing these strategies. Recalling that the Defender has no knowledge on when the Evader enters the transit area, we have to consider all possible mutual shifts of Evader's and Defender's walks; however, because Defender's walk s_D is perpetually repeated, we only need to consider $|s_D|$ shift. The interception probability can therefore be calculated as

$$\pi(s_E, s_D) = \frac{1}{|s_D|} \sum_{i=0}^{|s_D|-1} \pi(I^{\triangleright i}) \tag{3.9}$$

where

$$I^{\triangleright i} = s_E \cap s_D^{\infty \triangleright i} \tag{3.10}$$

For a given pure strategy pair $(s_E, s_D) \in S_E \times S_P$, we now define the Defender's utility $U_D(s_E, s_D)$ as equal to the interception probability, i.e.,

$$u_D(s_E, s_D) = \pi(s_E, s_D) \tag{3.11}$$

[4] for multiple resources, it is a set of locations

Defender mode	Exact utility	Approximate utility
Static	$1 - \prod_{l \in I(s_E, s_D)} (1 - \rho_l)$	$\sum_{l \in I(s_E, s_D)} \rho_l$
Static Base Mobile Base Unconstrained	$\frac{1}{\lvert s_D \rvert} \sum_{k=0}^{\lvert s_D \rvert - 1} \left(1 - \prod_{i=0}^{\lvert I^{\triangleright k} \rvert - 1} \left(1 - \rho_{I[i]} \right) \right)$	$\frac{1}{\lvert s_D \rvert} \sum_{k=0}^{\lvert s_D \rvert - 1} \sum_{i=0}^{\lvert I^{\triangleright k} \rvert - 1} \rho_{I[i]}$

Table 3.1: Defender's utilities

Approximate Utility Model

The utility models defined above are non-linear and could be difficult to optimize. We can also consider simpler linear models, used e.g., by Tsai et al. (2010b), where we simply sum up the encounter probabilities. The approximate utility for static Defender can be then expressed as:

$$U_D = \sum_{l \in I(s_E, s_D)} \rho_l \tag{3.12}$$

and the utility for the Defender with a fixed base, mobile base and the unconstrained Defender is then:

$$U_D = \frac{1}{\lvert s_D \rvert} \sum_{k=0}^{\lvert s_D \rvert - 1} \sum_{i=0}^{\lvert I^{\triangleright k} \rvert - 1} \rho_{I[i]} \tag{3.13}$$

The summary of all considered utility models can be found in Table 3.1 which expresses utility models for all possible combinations of evader's and defender's strategies.

3.2 Solution approach

The models described in previous sections are huge games, containing hundreds of thousands or even millions of strategies even for small games.

To be able to solve these large games, we use a column/constraint generation approach, referred as oracle-base algorithms (i.e., single- and double oracle algorithms, described in Section 2.2.2). These algorithms work on the assumption that the support set of both players in Nash Equilibrium (NE) is exponentially smaller than the strategy space of both players. The goal is to find a minimum set of strategies containing the support set—for one player in case

of the single-oracle algorithm and for both players in case of the double-oracle algorithm. From now on, we will consider w.l.o.g. only the double-oracle algorithm unless stated otherwise.

The algorithm starts from a small sub-game containing at least one strategy for each player; it iteratively adds strategies for both players and checks after each addition whether the Nash Equilibrium of the small sub-game is a Nash Equilibrium of the complete full game. The algorithm can be thus divided into 3 independent modules: (1) sub-game initialization, (2) sub-game expansion and (3) termination check.

Sub-game initialization module initializes the sub-game matrix with a set of strategies for each player. The sub-game should ideally contain most of the strategies which will be in the support set of the final sub-game or strategies which would allow the expansion module to quickly add strategies which are in the final sub-game NE support set. Typically, the game is initialized with a random strategy for each player, as it is in our case.

The sub-game expansion module should be able to enumerate the strategies which will be in the final support set as quickly as possible. This module can add multiple strategies at once (or none if not required) and it can use current sub-game as a guide which strategy to pick. Typically, the expansion module should be able to provide any strategy from the complete strategy set of the full game.

The termination module checks whether the Nash Equilibrium of the current sub-game is the Nash Equilibrium of the complete game.

So far, both the sub-game expansion module and the termination module have been handled by a single algorithm—the best-response oracle. The oracle provided a best-response to the current sub-game (thus expanding the game). If the oracle cannot provide a best-response which is not in the sub-game, the current sub-game's NE is considered the NE of the full game and the algorithm terminates. McMahan et al. (2003) proved that the sub-game's NE is indeed the NE of the full game.

By the decomposition to the sub-game expansion module and the termination module, we can re-use the best-response oracle for the termination check, however, we can enrich the game expansion module for additional mechanisms possibly improving the speed of the NE computation.

In this section, we discuss the extension of the double-oracle algorithm in the form of multiple sub-game expansion modules—an oracle hierarchy. Then, we discuss oracles for each player of each model. We express all oracles as mathematical programs. The issue is the non-linearity of the utility models which prohibits direct utilization of (mixed-integer) linear program solvers, such as CPLEX(CPLEX, 2005). We use our own implementation based on state space search—Branch&Bound for Defender's oracles and A* for Evader's oracles.

3.2.1 Extension of the Double Oracle Algorithm

The original double-oracle algorithm is depicted in Algorithm 2. The goal of the algorithm is to find all strategies which are in the support set of both players for any Nash Equilibrium of the game. The original algorithm uses one oracle for each player to iteratively add strategies to the

sub-game. We can generalize the algorithm by: (1) let the oracle return more than one strategy at once and (2) use more than one oracle for each player.

3.2.1.1 Partially Ordered Oracle Sets

The first modification is straightforward and results in a possibly faster search of the strategy support set. The extension of the algorithm to multiple oracles for each player can speed up the algorithm if the best-response oracles are not computationally fast and take significantly more time to compute a best-response than the computation of the Nash Equilibrium of the sub-game itself (as it is in our case). For each player, we can design a set of suboptimal oracles which are able to provide a only "better"-response of strategies with the trade-off of being fast. We introduce *partially ordered set (poset) of oracles* $\mathbf{\Omega}$ ordering the oracles in increasing computational complexity:

Definition 6. Oracle ordering of $\mathbf{\Omega}$ is defined as $\Omega_i \leq \Omega_j$ if $O(\Omega_i) \leq O(\Omega_j)$.

The effect of this ordering is such that we assume that oracles with a lower computational complexity are faster, however, they may provide a suboptimal response (i.e, not the best response). Note that this fact does not have a direct impact on the number of iterations of the algorithm. The strategy support set does not necessarily contain the best responses to the sub-games in each iteration. Intuitively, we hope that the suboptimal oracles will provide us with most of the strategies of the support set and the computationally intensive best-response oracle will generate a strategy only to check the termination condition.

Thus, to keep the modified double oracle algorithm optimal, we need the best response oracle Ω^* to be in the poset $\mathbf{\Omega}$ to check the termination condition. The extended double oracle algorithm is depicted in Algorithm 3.

Theorem 1. *If pure strategy sets of both players are finite, the Algorithm 3 with posets* $\mathbf{\Omega_1}$ *and* $\mathbf{\Omega_2}$*, containing best response oracles* Ω_1^* *and* Ω_2^* *respectively, finds the Nash equilibrium of the full game.*

Proof. The oracles from both posets $\mathbf{\Omega_1}$ and $\mathbf{\Omega_2}$ iteratively add strategies to the strategy sets of both players. The condition of the presence of the best response oracles on both posets guarantee the optimality of the algorithm, i.e., in the worst case, the oracles add iteratively all pure strategies to the player's strategy sets and the final sub-game equals to the full game. If the algorithm terminated *before* all pure strategies were added, it means that neither Ω_1^* nor Ω_2^* have found a better response. In that case, the termination condition is equal to the originally posed condition on the double oracle algorithm (McMahan et al., 2003)—it is satisfied only if the NE of the sub-game corresponds to the NE of the full game. Hence, this is also true for the extended oracle algorithm.

Algorithm 3 Extended Double-Oracle Algorithm.

```
equilibrium_found ← false
Ŝ1 ← { arbitrary strategy s1 ∈ S1}
Ŝ2 ← { arbitrary strategy s2 ∈ S2}
repeat
  (σ1, σ2) ← compute NE using LP for Ŝ1 and Ŝ2
  added ← false
  for Ω1 in Ω1 do
    s1 ← Ω1(σ2)
    if not (s1 ∈ Ŝ1) then
      Ŝ1 ← Ŝ1 ∪ {s1}
      added ← true
      break
    end if
  end for
  for Ω2 in Ω2 do
    s2 ← Ω2(σ1)
    if not (s2 ∈ Ŝ2) then
      Ŝ2 ← Ŝ2 ∪ {s2}
      added ← true
      break
    end if
  end for
  if not added then
    equilibrium_found ← true
  end if
until equilibrium_found
```

3.2.2 Best-Response Oracles

This section provides a mathematical formulation of best-response oracles for each player for game models proposed in Section 3.1. For the Defender, we have to design a different oracle for each mobility model, for the Evader's oracle, we can reduce the number of oracles to two: one for the static Defender, one for the Defender with any mobility model (i.e., fixed-base, mobile-base, unconstrained).

3.2.2.1 Defender's Oracle for Static Allocations

The best-response oracle for the static Defender provides a best response s_p^*, given Evader's current mixed strategy $\sigma_E = \{x_1, \ldots, x_a, \ldots, x_n\}$ over the Evader's path set S_E by maximizing the following criterion:

$$s_D^* = \arg\max_{s_D} \sum_{s_a \in S_E} x_a \cdot u\left(s_D, s_a\right) \tag{3.14}$$

Uncertain Encounters

For uncertain encounters, the oracle can be formulated as a mixed-integer non-linear program:

$$\max \sum_{s_a \in S_E} x_a \cdot z_a \tag{3.15}$$

$$\text{s.t.} \qquad z_a = 1 - \prod_{k=0}^{K} (1 - c_l \cdot B_l^a) \qquad \forall a \in S_E \tag{3.16}$$

$$\sum_l c_l \leq K \tag{3.17}$$

$$c_l = \{0, 1\} \qquad z_a = [0, 1] \tag{3.18}$$

where K is number of Defender's resources and B_l^a is a constant indicating that location l lies on a-th Evader's path defined as:

$$B_l^a = \begin{cases} \rho(l) \text{ if location } l \text{ on } a\text{-th Evader's path} \\ 0 \quad \text{otherwise} \end{cases} \tag{3.19}$$

$$\tag{3.20}$$

In this program uncertain encounters make the criterion non-linear (Equation (3.16)) if the Defender has multiple resources.

Certain Encounters

For certain encounters, the oracle providing the best response is formulated as a linear mixed-integer program (similarly to Defender's best-response oracle in the RUGGED algorithm (Jain et al., 2011a)):

$$\max \sum_{s_a \in S_E} x_a \cdot z_a \tag{3.21}$$

$$\text{s.t.} \qquad z_a \leq \sum_l c_l \cdot B_l^a \qquad a = 1 \ldots |S_E| \tag{3.22}$$

$$\sum_l c_l \leq K \tag{3.23}$$

$$c_l = \{0; 1\} \qquad z_a = [0, 1] \tag{3.24}$$

Note that for the static Defender with multiple homogeneous resources, certain encounters and the Evader that does not differentiate between the destination nodes, the solution can be found without using the oracles (Washburn and Wood, 1995). We introduce this mathematical program to provide a complete enumeration of all oracles.

3.2.2.2 Defender's Oracle for a Fixed Base

The formulation of mobile Defenders is a non-trivial program based on flows. Additionally, thanks to the uncertainty about the time step in which the Evader enters the graph, the Defender can be *shifted* for any number of nodes on its walk (see Section 3.1.4). For convenience of the

notation, we index the walk by location, i.e., we index *both* nodes and edges on the walk. Each shift (from one node to another) is thus an increment by two. Nodes have then even indices and edges have odd indices.

Uncertain Encounters

We formulate a mixed-integer non-linear program for Defender's best-response oracle for uncertain encounters and for maximum allowed walk length of Ω as follows:

$$\max \quad V_\Omega \tag{3.25}$$

$$V_\Omega = \sum_{s_a \in S_E} x_a \cdot u_a \tag{3.26}$$

$$u_a = \frac{1}{\Omega} \sum_{j=0}^{\Omega-1} {}^{j}_{a}\omega \qquad \forall s_a \in S_E \tag{3.27}$$

$${}^{j}_{a}\omega = 1 - \prod_{l \in s_a} \left(1 - {}^{j}_{a}z_l \cdot \rho_l\right) \qquad \forall s_a \in S_E, \quad \forall j \tag{3.28}$$

$${}^{j}_{a}z_l \geq {}^{j}c^i_l + {}_{a}p^i_l - 1 \qquad {}^{j}_{a}z_l \geq 0 \qquad \forall a \in S_E, \quad \forall l \in L, \quad \forall j \tag{3.29}$$

$$\sum_{e \in out(b)} {}^{j}c^1_e = 1 \qquad \forall j \tag{3.30}$$

$${}^{j}c^\Omega_b = 1 \qquad \forall j \tag{3.31}$$

$$\sum_{e \in out(n)} {}^{j}c^{i+2}_e = \sum_{e \in in(n)} {}^{j}c^i_e \qquad i = 1, \ldots, \Omega - 2, \forall n \in N, \forall j \tag{3.32}$$

$${}^{j}c^{i+1}_n = \sum_{e \in in(n)} {}^{j}c^i_e \qquad i = 1, \ldots, \Omega - 2, \forall n \in N, \forall j \tag{3.33}$$

$$\sum_{l \in L} {}^{j}c^i_l = 1 \qquad i = 1, \ldots, \Omega, \forall j \tag{3.34}$$

$${}^{j}c^i_l = {}^{j+1}c^{i+2}_l \qquad i = 1, \ldots, \Omega, \forall l \in L, \forall j \tag{3.35}$$

$${}^{j}_{a}z_l = \{0, 1\} \qquad \forall a \in S_E, \forall l \in L, \forall j \tag{3.36}$$

$${}^{j}c^i_l = \{0, 1\} \qquad i = 1, \ldots, \Omega, \forall j \tag{3.37}$$

Equation (3.25) is a maximization criterion, rewritten in Equation (3.26) as a sum of products probability x_a of the Evader playing strategy $s_a \in S_E$ which gives the Defender a utility u_a. This utility can be expressed as a sum of interception probabilities over all possible shifts j of the Defender's walk ${}_{a}{}^{j}\omega$ weighted by the length of the Defender's walk Ω (Equation (3.27)). Equation (3.28) — valid for uncertain encounters — introduces a variable ${}^{j}_{a}z_l$ which indicates, if the Evader and Defender are on the same location l (for j-th shift of Defender's walk and a-th Evader's path). We further introduce a variable ${}^{j}c^i_l$ indicating that a location l is i-th on j-th shift of the Defender's path and constant ${}_{a}p^i_l$ set to 1 if location l is i-th[5] on a-th Evader's path (Equations (3.29)). Using ${}^{j}c^i_l$, we can now pose constraints on the Defender's walk: the

[5] More precisely, ${}_{a}p^i_l$ is set to 1, if the location l is also on positions $(i + Omega)$, $(i + 2\Omega)$, etc. on a-th Evader's path.

walk has to start and end in the base (Equations (3.30) and (3.31) respectively) and we further impose typical flow constraints on the Defender's path (Equations 3.32 – 3.34). Finally, we have to constrain the shifts of a walk, to correspond with each other, i.e, i-th location of j-th shift has to be $i+2$-th on $j+1$ shift (Equation 3.35).

Additionally, we can add following redundant equations to constrain the problem more (and solve the problem faster):

$$\sum_{e \in out(b)} \sum_{i=2}^{\Omega} {}^{j}c_e^i \geq 1 \qquad \forall j \tag{3.38}$$

$$\sum_{e \in in(b)} \sum_{i=2}^{\Omega} {}^{j}c_e^i \geq 1 \qquad \forall j \tag{3.39}$$

$$\sum_{e \in out(b)} \sum_{i=1}^{K} {}^{j}c_e^i = \sum_{e \in in(b)} \sum_{i=2}^{K} {}^{j}c_e^i \qquad \forall j \tag{3.40}$$

$$\tag{3.41}$$

i.e., number of edges outgoing from the base have to be greater than 0 (Equation (3.38)), number of edges going to the base has to be greater than 0 (Equation (3.39)) and the number of outgoing and back-going edges has to be equal (Equation (3.40))[6].

Note that we can rewrite the criterion to correspond to the utility formulation (compare with Table 3.1):

$$\max \quad \sum_{s_a \in S_E} x_a \cdot \frac{1}{\Omega} \sum_{j=0}^{\Omega-1} \left(1 - \prod_{l \in s_a} \left(1 - {}^{j}_{a}z_l \cdot \rho_l \right) \right) \tag{3.42}$$

Certain Encounters

For certain encounters, we redefine Equation (3.28) as:

$${}^{j}_{a}\omega = sgn \sum_{l \in s_a} {}^{i}_{a}z_l \qquad \forall s_a \in S_E \tag{3.43}$$

which can be converted to a linear constraint:

$$M \cdot {}^{j}_{a}\omega - \sum_{l \in s_a} {}^{j}_{a}z_l \geq 0 \qquad \forall j, \forall a \tag{3.44}$$

$${}^{j}_{a}\omega = \{0; 1\} \tag{3.45}$$

where M is a large number. The best response for a fixed length Ω — denoted as ${}_{\Omega}s_D{}^*$ — is reconstructed from indicator variables ${}^{j}c_l^i$ and its value is V_Ω. We compute the best response for each $\Omega = 3, \ldots, K$ (the length of the sought Defender's walk) and we pick the one with the highest value V_Ω.

[6] Note that the walk can pass through the base multiple times.

3.2.2.3 Defender's Oracle for a Mobile Base

For the Defender's best-response oracle with a mobile base, we can execute the base oracle $|N|$-times, i.e, for each node, we can get a best response candidate (by setting the node temporarily to be a base) and then pick the best from the candidates.

3.2.2.4 Defender's Oracle for an Unconstrained Movement

The unconstrained Defender's strategy is a sequence of nodes which he observes. To be compatible with the framework above, we add (redundantly) all edges going to every node in the sequence.

Uncertain Encounters

The best-response oracle for the unconstrained movement can be created from the fixed-base Defender's oracle by lifting some of the assumptions on the Defender's walk:

$$\max \quad V_k \tag{3.46}$$

$$V_k = \sum_{s_a \in S_E} x_a \cdot u_a \tag{3.47}$$

$$u_a = \frac{1}{\Omega} \sum_{j=0}^{\Omega-1} {}^{j}_{a}\omega \qquad \forall s_a \in S_P \tag{3.48}$$

$${}^{j}_{a}\omega = 1 - \prod_{l \in s_a} \left(1 - {}^{j}_{a}z_l \cdot \rho_l\right) \qquad \forall s_a \in S_P, j \tag{3.49}$$

$${}^{j}_{a}z_l \geq {}_{j}c^{i}_{l} + {}_{a}p^{i}_{l} - 1 \qquad {}^{j}_{a}z_l \geq 0 \qquad \forall a \in S_E, l \in L, j \tag{3.50}$$

$${}^{j}c^{i+1}_{n} = \sum_{e \in in(n)} {}^{j}c^{i}_{e} \qquad \forall i = 1, \ldots, \Omega - 2, n \in N, j \tag{3.51}$$

$$\sum_{l \in L} {}^{j}c^{i}_{l} = 1 \qquad \forall i = 1, \ldots, \Omega, j \tag{3.52}$$

$${}^{j}_{a}z_l = \{0, 1\} \qquad \forall a \in S_E, l \in L, j \tag{3.53}$$

$${}^{j}c^{i}_{l} = \{0, 1\} \qquad \forall i = 1, \ldots, \Omega, j \tag{3.54}$$

Equation (3.46) is a maximization criterion, rewritten in Equation (3.47) as a sum of products probability x_a of the Evader playing strategy a which gives the Defender utility u_a. The utility can be expressed as a sum of interception probabilities over all possible shifts of the Defender's walk ${}_{a}{}^{j}\omega$ weighted by the length of the Defender's walk Ω (Equation (3.48)). Equation (3.49) — valid for uncertain encounters — uses a variable ${}^{j}_{a}z_l$ which indicates, if the Evader and Defender are on the same location l (for j-th shift of Defender's walk and a-th Evader's path). We again use a variable ${}^{j}c^{i}_{l}$ indicating that a location l is i-th on j-th shift of the Defender's path and constant ${}_{a}p^{i}_{l}$ set to 1 if location l is i-th on a-th Evader's path (Equations (3.50)). Further on, to generalize the concept of movement expressed by a combined node-edge- node-edge- ... movement in base and no-base Defenders, we include also the edges going to a node which

is observed (Equation (3.51)). The strategy has no flow restrictions, however, the Defender can look at most on 1 place at once (Equation (3.52)).

Certain Encounters

The oracle for certain encounters is constructed analogically, by lifting the same assumptions as above and using the same criterion as in case of a fixed-base Defender's best-response oracle.

3.2.2.5 Evader's Oracles

The Evader's oracle provides a best response s_E^* given Defender's strategy set S_D and his mixed strategy $\sigma_P = \{x_1, \ldots, x_n\}$ by minimizing the following criterion:

$$s_E^* = \arg\min_{s_E} \sum_{s_a \in S_D} x_a \cdot u\left(s_E, s_a\right) \tag{3.55}$$

Static Defender

For the Evader, given Defender's static allocation of resources and mixed strategy $\sigma_P = \{x_1, \ldots, x_n\}$ over these allocations, the best response s_E^* is sought in the form of the following mixed integer program:

$$\min \sum_{a \in S_D} x_a \cdot z_a \tag{3.56}$$

$$\text{s.t.} \qquad z_a = \prod_{l \in D} (1 - c_l \cdot B_l^a \cdot \rho_l) \qquad \forall s_a \in S_D \tag{3.57}$$

$$\sum_{o \in \mathcal{O}} \sum_{l \in out(o)} c_l = 1 \tag{3.58}$$

$$\sum_{d \in \mathcal{D}} \sum_{l \in in(d)} c_l = 1 \tag{3.59}$$

$$\sum_{e \in in(v)} c_e = \sum_{e \in out(n)} c_e \qquad \forall n \in N \tag{3.60}$$

$$c_n \geq \sum_{e \in in(n)} c_e \qquad \forall n \in N \tag{3.61}$$

Where B_l^a is set to 1 if Defender's a-th allocation contains location l.

For certain encounters, the Equation (3.57) is substituted with:

$$z_a \geq c_l + B_l^a - 1 \tag{3.62}$$

$$z_a \geq 0 \tag{3.63}$$

$$z_a = \{0, 1\} \tag{3.64}$$

Mobile Defender

For the fixed-base Defender's movement as well as for the mobile-base and unconstrained Defender's movement, the Evader's oracle can be formulated using a single mixed integer program:

$$\min \quad V \tag{3.65}$$

$$V = \sum_{s_a \in S_D} x_a \cdot u_a \tag{3.66}$$

$$u_a = \frac{1}{|s_a|} \sum_{j=0}^{|s_a|-1} {}^{j}_{a}\omega \qquad \forall s_a \in S_D \tag{3.67}$$

$${}^{j}_{a}\omega = 1 - \prod_{l \in s_a} \left(1 - {}^{j}_{a}z_l \cdot \rho_l\right) \qquad \forall s_a \in S_D, \forall j \tag{3.68}$$

$${}^{j}_{a}z_l \geq c^i_l + {}^{j}_{a}p^i_l - 1 \qquad {}^{j}_{a}z_l \geq 0 \qquad \forall a \in S_D, \forall l \in L, \forall j \tag{3.69}$$

$$\sum_{o \in \mathcal{O}} \sum_{e \in out(o)} c^1_e = 1 \tag{3.70}$$

$$\sum_{d \in \mathcal{D}} \sum_{e \in in(d)} \sum_{i=2}^{K} c^i_e = 1 \tag{3.71}$$

$$\sum_{e \in out(n)} c^{i+2}_e = \sum_{e \in in(n)} c^i_e \qquad i = 1, \ldots, K-2, \forall n \in N \tag{3.72}$$

$$c^{i+1}_n = \sum_{e \in in(n)} c^i_e \qquad \forall n \in N, i = 1, \ldots, K-2 \tag{3.73}$$

$${}^{j}_{a}z_l = \{0, 1\} \qquad \forall a \in S_D, \forall l \in L, \forall j \tag{3.74}$$

$$c^i_l = \{0, 1\} \qquad i = 1, \ldots, \Omega, \forall l \in L \tag{3.75}$$

The program is analogous to the Defender's best response oracle. Equation (3.65) is a minimization criterion, rewritten in Equation (3.66) as a sum of products probability x_a of the Defender playing strategy s_a which gives the Defender utility u_a. The utility can be expressed as a sum of interception probabilities over all possible shifts of the Defender's walk ${}_{a}{}^{j}\omega$ weighted by the length of the Defender's walk $|s_a|$ (Equation (3.67)). Equation (3.68)—valid for uncertain encounters—uses a variable ${}^{j}_{a}z_l$ indicating, if the Evader and Defender are on the same location l (for j-th shift of Defender's walk and a-th Evader's path). Variable c^i_l does not depend on the shifts of the Defender's walks, it indicates that a location l is i-th on Evader's path and the constant ${}^{j}_{a}p^i_l$ is set to 1 if location l is i-th on the j-th shift of a-th Defender's path (Equation (3.69)). Using c^i_l, we can now pose constraints on the Evader's path (as in the case of the Defender), i.e, the number of edges outgoing from the origin nodes $o \in \mathcal{O}$ have to be equal to 1 (Equation (3.70)), the number of edges ending in any of the destination node $d \in \mathcal{D}$ has to be equal to one (Equation (3.71)). We further impose typical flow constraints on the Defender's path (Equations (3.72) and (3.73)).

For certain encounters — as in the case of the Defender's best response oracle—the Equation (3.68) can be replaced by Equations (3.44) and (3.45).

3.2.3 Suboptimal Oracles

The sections above described best-response oracles suitable for both the termination module as well as the expansion module. We term these oracles optimal (in the sense that they provide a *best*-response). As it can be seen from the programs above (and as it is proven for a subset of oracles Jain et al. (2011a)), the responses are hard to compute. We can design faster oracles, that provide the responses faster, however, cannot guarantee that the response is a best-response from the complete strategy set and these *suboptimal* oracles cannot be thus used for the termination check (unless we want to trade optimality for speed). However, they can be used for the subgame expansion module to compute fast responses and enumerate the support set faster than the optimal oracles.

Computationally faster oracles can be designed in three ways:

1. We restrict the strategy space to contain only a subset of strategies of the full model (and hope that this subset contains all—or most of—the strategies which are in the final support set) and let the oracle look for a best response in this subset.
2. We provide the oracle with an approximate (i.e., simpler) model of the problem and let the oracle look for the best response for this simpler model (in our case, approximate the computation of the utility and thus linearize the constraints in the mathematical model).
3. We let the oracle look for any strategy which achieves a better utility for the player than the current mixed strategy and let it return the first better strategy it finds.

These three approaches to a faster oracles design can be combined together and design an oracle with an approximate model looking through a subset of strategies and returning the first strategy improving the utility of a player. Of course, the trade-off for the speed of the computation could be reflected in the convergence rate to the Nash Equilibrium.

The utility approximation significantly reduces the complexity of the oracles.

3.2.3.1 Defender's Oracles with Approximate Utility

The Defender's best response oracle for the approximate utility and the static allocation is found by solving the following program:

$$\max \quad \sum_{s_a \in S_E} x_a \cdot \sum_{l \in s_a} c_l \cdot \rho_l \tag{3.76}$$

$$\sum_{l \in L} c_l \leq K \tag{3.77}$$

$$c_l = \{0, 1\} \tag{3.78}$$

where Constraint (3.77) limits the number of Defender's resources to be allocated to at most K.

For the fixed-base, the mobile-base and the unconstrained Defender, the suboptimal oracle with an approximate utility provides a response by optimizing the following criterion:

$$\max \quad \sum_{s_a \in S_E} x_a \cdot \frac{1}{\Omega} \sum_{j=0}^{|\Omega|-1} \sum_{l \in s_a} \sum_{i=0}^{|s_a|-1} {}^{j}c_l^i \cdot {}_{a}p_l^i \cdot \rho_l \tag{3.79}$$

$${}^{j}c_l^i = \{0, 1\} \tag{3.80}$$

The constraints differ for each type of the Defender's strategy. For the Defender with a fixed base and with mobile base, the criterion is optimized subject to constraints posed on the Defender's walk given by Equation (3.30) – (3.35). For unconstrained walks, the constraints are given by Equations (3.51) – (3.52).

3.2.3.2 Defender's Oracles with a Strategy Subset

For the fixed-base and the mobile-base Defender, we design a fast oracle by considering only a subset of possible walks: for each node or edge p reachable from the base, we compute the shortest path from the base $path_{base \to p}$ count number of steps of the final walk left (using the maximal length of walk allowed), construct a loop over a node or walk over the edge *loop* and connect the components of the walk together:

$$walk \leftarrow base + path_{base \to p} + loop + path'_{base \to p} + base \tag{3.81}$$

where $path'_{base \to p}$ is the reversed shortest path from the base to p.

It is crucial to consider both nodes and edges as looping places. The nodes are important to consider as they block multiple edges at once and edges are important as the movement of the Defender can be designed to go *against* the direction of the Evader.

3.2.3.3 Evader's Oracles with an Approximate Utility

Given Defender's static allocations, the Evader's oracle computing with the approximate utility model can be expressed as the following minimization criterion:

$$\min \quad \sum_{s_a \in S_P} x_a \sum_{l \in s_a} c_l \cdot \rho_l \tag{3.82}$$

$$c_l = \{0, 1\} \tag{3.83}$$

which is subject to constraints posed on the properties of the path, given by Equations (3.58) – (3.61).

For the mobile Defender with fixed base, mobile base or unconstrained movement, the Evader's approximate utility oracle minimizes a criterion in the following form:

$$\min \sum_{s_a \in S_P} x_a \cdot \frac{1}{|s_a|} \sum_{j=0}^{|s_a|-1} \sum_{l \in L} \sum_{i=0}^{K-1} c_l^i \cdot {}_{a}^{j}p_l^i \cdot \rho_l \tag{3.84}$$

subject to the constraints posed on the properties of the path, given by Equations (3.70) – (3.73).

3.3 Implementation of Oracles

Due to the non-linear nature of the constraints in the mathematical programs capturing some of the oracles, we cannot directly implement the oracles using standard (Mixed Integer) Linear Programming libraries (such as CPLEX (CPLEX, 2005)). Non-linear programming libraries typically do not guarantee finding the global optimum. We thus implement the oracles using standard Java SDK using branch&bound algorithms and minimum cost path search algorithms (such as A*).

3.3.1 Evader's Oracles

All Evader's oracles can be implemented using a variant of a constructive algorithm based on state-space search, additively constructing a minimum-cost path from any origin node to any destination node. To simplify the origin and destination node selection, we represent all origin nodes by a single node *ORIGIN* and all destination nodes by a single node *DESTINATION*, and we look for a minimum cost path between these two nodes.

We use A* algorithm with a cost function $g(\hat{s}_e) = u(\sigma_D, \hat{s}_e)$, i.e., the cost function is a utility of the partial path, given current mixed strategy of the Defender, σ_D. The heuristic function of A* estimates a minimal possible encounter probability increment from the current node to any destination node. For uncertain encounters, $h = 0$ and the algorithm is thus reduced to the best first search.

For certain encounters (or for approximate utility models assuming the additivity of probabilities of encounter), we can pre-process the graph to better estimate minimum encounter probability from a given node to any destination node possible. For each node, we compute the minimum distance (i.e., the number of nodes on the shortest path) to any destination node. We denote this number as a node layer. I.e., nodes which are neighbors of the destination node are in layer 0, nodes which are neighbors of nodes in layer 0 are in layer 1 etc. In this way, we are able to number all nodes in the graph up to the origin nodes. The layer-based numbering of the nodes is depicted in Figure 3.1. For each node, we compute the probability of the Defender being in that node from its mixed strategy σ_D; we assign a probability of encounter to each node $p^{\sigma_D}(n)$.

We know that any path from any origin node to any destination node has to pass at least once through each layer. In each layer, we find the minimum probability of encounter, denoted as $p^{\sigma_D}_{min}(l) = min\{p^{\sigma_D}(n_1), \ldots, p^{\sigma_D}(n_m)\}_{|layer(n_i)=l}$. The heuristic function for each node is then expressed as:

$$h(n) = \sum_{l=0}^{layer(n)-1} p^{\sigma_D}_{min}(l) \tag{3.85}$$

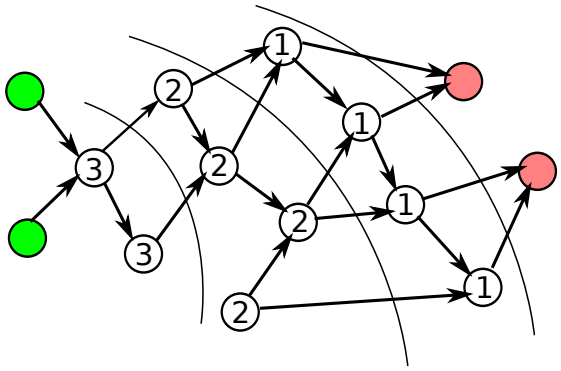


Fig. 3.1: Example of graph with nodes numbered according to their layers. The nodes next to the destination nodes (red) are in the first layer.

3.3.2 Defender's Oracles

In case of Defender's Oracle, the problem is to find a maximum cost path of the length varying from 1 to l, i.e., a path that maximizes the probability of encounter. This problem is NP-hard, a variant of the *longest path problem* (Schrijver, 2003).

3.3.2.1 Best-Response Defender's Oracle

To find a closed walk which maximizes the probability of encounter, we design a Branch&Bound-based algorithm (Cormen et al., 2001) constructing possible closed walks and bounding unfinished ones by bounds computed from the completed closed walks. This algorithm is used directly for the fixed-base Defender and is described in Algorithm 4. First, we pre-process the graph to mark nodes which are reachable from the base. We initiate the best path and a bound by a random walk from the base; we compute the maximum possible utility $maxUtilPerStep$ which can be gained per one step of Defender's walk. We initiate the *openList* with partial paths leading from the base in the form $path_i = \{base, edge(base, neighbour_i), neighbour_i\}$. Then, until the *openList* is not empty, we periodically poll a candidate from the list (sorted by maximum utility of the partial walk constructed). If the candidate is complete, we update the bound, the best solution found so far and we prune the *openList* by removing those partial walks which cannot possibly have the same utility as the bound found so far. If the candidate is incomplete, we generate all possible walks from this candidate (taking into account the bound and maximum length of the walk $maxLength$) and add the children into the *openList*. Once the *openList* is clear, we are guaranteed to have found an optimal closed walk from the base *best*.

We recycle this approach in case of the mobile-base Defender, where we iteratively change the position of the base and use returned best responses for a given base in subsequent iterations as a lower bound.

Algorithm 4 Template for the static-base Defender's Branch&Bound Oracle.

```
graph ← preprocess(graph, maxLength)
best ← randomWalk
bound ← computeBound(best)
maxUtilPerStep ← getMaxUtilPerStep(graph, σ1)
openList ← initPaths(graph)
repeat
  candidate ← openList.poll()
  if complete(candidate) then
    bound ← updateBound(bound, candidate)
    best ← updateBest(best, candidate)
    openList ← prune(openList,amxUtilPerStep)
  else
    children ← generateChildren(candidate, maxUtilPerStep, maxLength, bound)
    openList ← openList ∪ children
  end if
until isEmpty(openList)
return best
```

Algorithm 5 Template for the static-base Defender's Branch&Bound Oracle.

```
graph ← preprocess(graph,σ1)
best ← randomWalk
for length in minLength ... maxLength do
  for place in graph(length) do
    walk ← shortestPath(base,place) ∪ loop(length) ∪ shortestPath(place,base)
    best ← updateBest(best,walk)
  end for
end for
return walk
```

3.3.2.2 Defender's Oracle with Subset of Strategy Space Exploration

The suboptimal Defender's oracle is inspired by the fact that some of the places in the graph are best to be guarded statically; i.e., the suboptimal oracle construct walks which wait at some place; then, the optimal best-response oracle fills walks which utilize the movement capability.

The algorithm for the suboptimal oracle is depicted in Algorithm 5. First, we pre-process the graph to find a set of places which are reachable from the base. Then, for each possible length of Defender's walk and for each place p_i reachable from the base by a walk with this length, we construct the path concatenating (1) the shortest path from the base to p_i, (2) a path of waiting in p_i and (3) the shortest path from p_i to base. The second component of the walk (the loop) differs for nodes and edges. For nodes, we use the loop (if available—if not, we do not have to consider nodes), for an edge $e(source, target)$, we construct the loop by adding periodically $e(source, target)$ and $e(target, source)$. The constructed walk is compared with the best found so far. After all possible lengths and all possible places have been explored, the best path found is returned by the algorithm.

3.4 Evaluation

In this Section, we focus on the evaluation of selected aspects of the solution of the game-theoretic models formalized in Section 3.1 and on properties of the algorithms designed to find solutions for the models. We examine the structure of the solution, i.e., the strategies of both players in Nash Equilibrium of the game. We look at the advantage of Defender's Mobility and the role of the walk length in the quality of the solution. We pause to explore the possibility of the Defender to stop during his walk, quantifying the principal difference between the utilization of VTOL[7] UAVs. and aircraft UAVs. We then explore the properties of oracle-based algorithms, namely the convergence properties and Scalability of both a simple double-oracle algorithm and an extended hierarchical double-oracle algorithm.

3.4.1 Structure of the Solution

For non-trivial cases, both the Evader and the Defender play mixed strategy in the Nash Equilibrium. In a homogeneous environment (i.e., for certain encounters or for encounters with constant interception probability) and for the mobile-base Defender strategies, the resulting Evader's strategies are homogeneous, i.e., the Evader uniformly randomizes over a subset of paths to maximize the entropy of his strategy. In the same environment and for the Defender with a static a base, the Evader's strategy set is influenced by the position of the base, which is discussed in more detail in Section 3.4.3.

Another interesting set of observations can be made about the possibility of the Defender to wait in a node during his walk. This option emulates, e.g., the difference between an airplane-like UAV[8] and a helicopter-like UAV (such as Quadcopters etc.) where the airplanes cannot stop, however, the helicopters can. We will show that the possibility of waiting on a given node could significantly help the Defender.

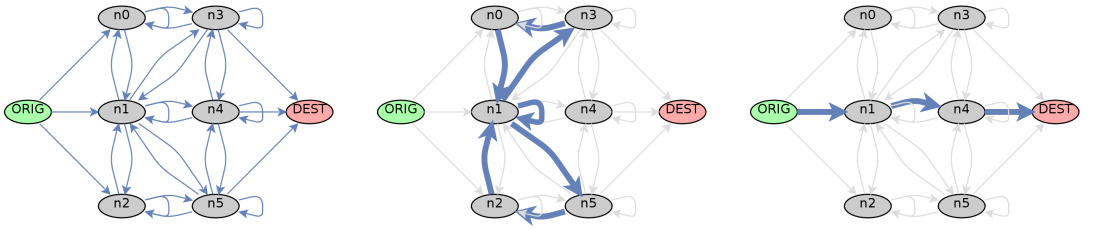
All demonstrations are conducted on rectangular graphs as the structure of the solution is most readable from the graph. Also, the performance of the oracle-based algorithms was the lowest, compared to scale-free graphs or random planar graphs.

3.4.2 Evaluation of Defender's Mobility

An exemplar graph used to demonstrate the advantage of the mobility of both the Defender as well as the base (compared to static Defender allocation resources either on edges or nodes on the graph) is depicted in Figure 3.2 (interception probability is 1 on all places). The Evader has to cross a two-layer graph from left to right. To intercept the Evader, the Defender with static

[7] VTOL – vertical take-off and landing unmanned aerial vehicle.

[8] Unmanned Aerial Vehicle



(a) Graph for demonstration of advantages of the Defender being mobile.

(b) Defender's strategy is composed of three pure strategies: loop in n_1 and two walks.

(c) Evader's best response to such strategy is one of the paths directly crossing the graph.

Fig. 3.2: Demonstration of the advantage of being mobile (with or without a base). In this case, static allocation of one resource would result in expected probability of 1/3 of the Defender catching the Evader (assuming certain encounters). For the mobile Defender, the probability is increased to 3/7, which is approximately 0.4286.

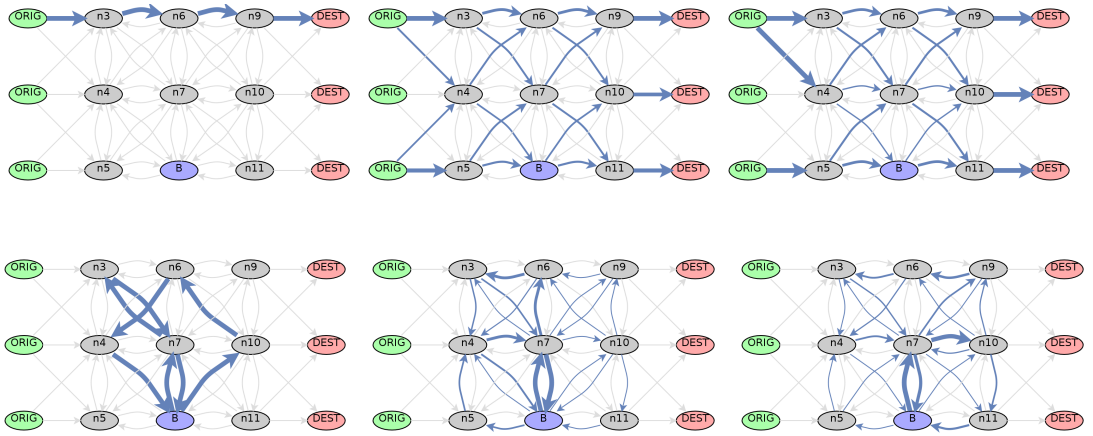
allocations has to equally randomize between three nodes, e.g., $\sigma_D = \{n_0 : 1/3, n_1 : 1/3, n_2 : 1/3\}$ which results in the probability of 1/3 of catching the Evader[9].

If we allow the Defender to move from a static base, he can increase its changes of intercepting the Evader to $3/7 = 0.4286$ (for walk of length $l = 7$). The mixed strategy of the fixed-base Defender is depicted in Figure 3.2b. For such a strategy, the Evader can respond with a pure strategy, which could be any shortest path between origin and destination node (one depicted in Figure 3.2c). Note that in this example, if the Defender is constrained by a base, the base has to be positioned in node n_1 or n_4 for the same length of walk. If we position the base in either of the corners (i.e., n_0, n_2, n_3 or n_5), the length of the walk has to be increased at least to $l = 15$ to achieve the same value. However, for the mobile-base Defender, the position of the base does not matter and with the length $l = 7$, he can intercept the Evader with probability 3/7.

3.4.3 Evaluation of Walk Length

The length of a walk for mobile Defenders reveals some interesting facts about the movement of the Defender. Figure 3.3 examines different Evader's and Defender's strategies for different lengths of fixed-base Defender's walks and Table 3.2 summarizes players' mixed strategy. Note that for a static patrolling policy, the probability of encounter (and thus the game value) for any equilibrium strategy is $V = 0.333$. The presence of the base restricts the Defender,i.e., some nodes in the graph which are farther from the base cannot be visited as frequently as required. However, if enough mobility provided (i.e., the walk can be long enough), it provides the fixed-base Defender with an advantage against a static allocation. This can be observed in Figure 3.4, where, for walk length $l < 9$, the Defender cannot reach the opposite side of the graph and

[9] The Defender has to cover a minimum cut of the graph (mincut) of the graph between the origin and destination nodes—i.e., allocate its resource on each node in the mincut with equal probability.



(a) Defender's walk length $l = 9$. Game value $V = 0.25$.

(b) Defender's walk length $l = 11$. Game value $V = 0.352$.

(c) Defender's walk length $l = 13$. Game value $V = 0.357$.

Fig. 3.3: Evader's and Defender strategies (upper and lower row respectively) for different base-constrained Defender's walk length (9, 11 and 13) for left, middle, and right column respectively on a rectangular graph with width = 3 and length = 3, no loops, certain encounters.

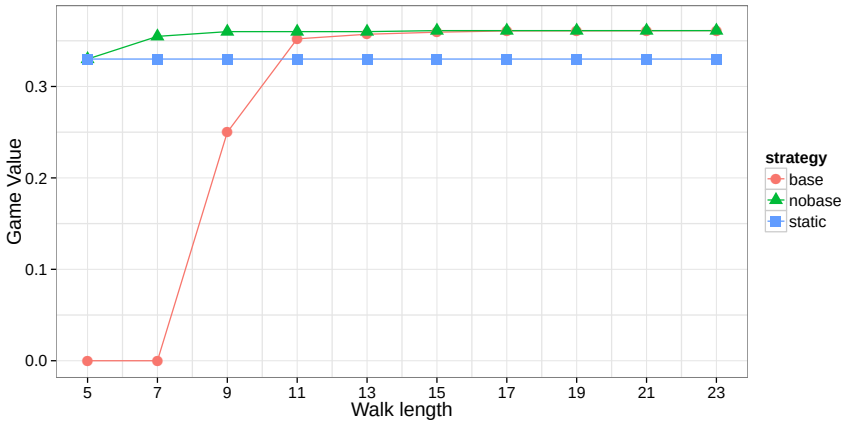


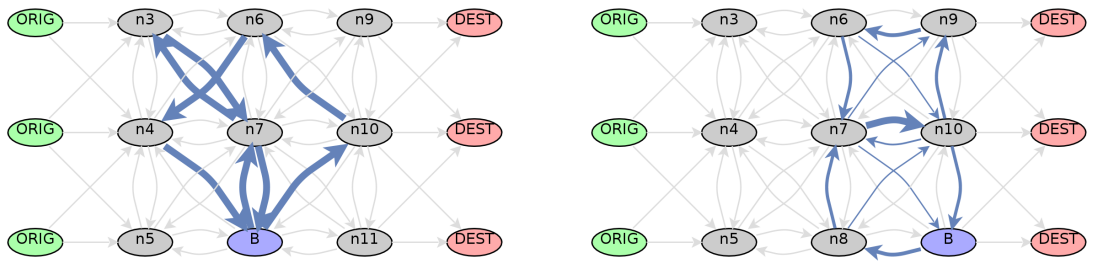
Fig. 3.4: Dependency of the value of the game on graph displayed in Figure 3.3 on the length of Defender's walk. The value of the game converges to a stable value (in this case, $V = 3.6111$) as the length of the walk is increased.

the game value is 0. For the length $l = 9$, the Defender can visit nodes on the opposite side of the graph, however, not as frequently as required, the game value is thus lower than for a static allocation ($V = 0.25$). However, if we allow longer walks, the Defender can leverage the movement to increase the probability of an encounter ($V = 0.3611$ for $l = 23$).

The solution for the Defender with a mobile base is always superior to the solution for the base-constrained Defender (as it can be observed in Figure 3.4). When we compare Defenders' strategies for walk of length $l = 9$ (depicted in Figure 3.5), we can see that the fixed-base Defender can only touch the opposite side of the graph and not spend enough time to catch the Evader; on the other hand, the mobile-base Defender can position the base as he likes and can thus cover all required nodes and edges appropriately.

Table 3.2: Comparison of player strategies when changing the length of the walk.

Walk length	9	11	13
D's SS size	2	10	10
E's SS size	1	7	9
Game value	0.25	0.352	0.357
D's Strategy	$\{n8; n7; n3; n7; n8;\}0.5$, $\{n8; n10; n6; n4; n8;\}0.5$	$\{n8; n7; n9; n6; n4; n8;\}0.107$, $\{n8; n4; n7; n8;\}0.046$, $\{n8; n10; n6; n4; n7; n8;\}0.065$, $\{n8; n5; n4; n7; n8;\}0.210$, $\{n8; n7; n3; n4; n7; n8;\}0.041$, $\{n8; n7; n10; n9; n7; n8;\}0.032$, $\{n8; n7; n6; n3; n7; n8;\}0.253$, $\{n8; n7; n6; n3; n4; n8;\}0.178$, $\{n8; n7; n10; n8;\}0.009$, $\{n8; n7; n10; n11; n8;\}0.052$	$\{n8; n7; n9; n6; n3; n7; n8;\}0.241$, $\{n8; n5; n7; n3; n4; n7; n8;\}0.080$, $\{n8; n7; n10; n11; n8;\}0.089$, $\{n8; n5; n4; n7; n4; n7; n8;\}0.214$, $\{n8; n7; n10; n9; n7; n11; n8;\}0.026$, $\{n8; n7; n10; n6; n4; n7; n8;\}0.026$, $\{n8; n7; n10; n9; n6; n4; n8;\}0.080$, $\{n8; n7; n10; n6; n3; n7; n8;\}0.053$, $\{n8; n10; n6; n3; n4; n7; n8;\}0.133$, $\{n8; n5; n4; n7; n8;\}0.053$
E' Strategy	$\{n0; n3; n6; n9; n12;\}1.0$	$\{n0; n4; n6; n10; n13;\}0.154$, $\{n2; n4; n8; n11; n14;\}0.140$, $\{n2; n5; n8; n10; n13;\}0.140$, $\{n1; n3; n7; n11; n14;\}0.154$, $\{n2; n5; n8; n11; n14;\}0.028$, $\{n0; n3; n6; n9; n12;\}0.225$, $\{n2; n5; n7; n9; n12;\}0.154$	$\{n0; n3; n6; n9; n12;\}0.190$, $\{n2; n5; n8; n10; n13;\}0.023$, $\{n0; n4; n7; n10; n13;\}0.095$, $\{n2; n5; n7; n9; n12;\}0.142$, $\{n0; n4; n8; n10; n13;\}0.071$, $\{n0; n4; n6; n10; n13;\}0.142$, $\{n2; n5; n8; n11; n14;\}0.166$, $\{n0; n3; n7; n11; n14;\}0.142$, $\{n0; n4; n8; n11; n14;\}0.023$



(a) Fixed-base Defender.

(b) Defender with a mobile base.

Fig. 3.5: Comparison of Defender's strategies for fixed-base and mobile-base mobility modes.

The mobility of the base gives a significant advantage when the walks are short, however, if given long enough walk, both Defender modes converge to the same value.

3.4.4 Evaluation of Waiting/Pausing Ability

We demonstrate the difference between the Defender with the ability of waiting over any node and the Defender without such ability, demonstrating the difference between mobile resources with and without such capabilities (i.e., airplane vs. helicopter). We do not need to modify pro-

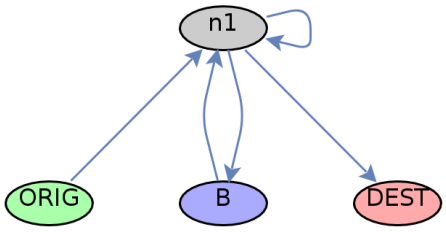


Fig. 3.6: An example of a graph where the Defender leverages its ability of waiting i given node. If the Defender can wait in $n1$, he intercepts the Evader with probability reaching 1 as the length of walk is increased to infinity. For the Defender without the waiting ability, the probability cannot be more than 0.5.

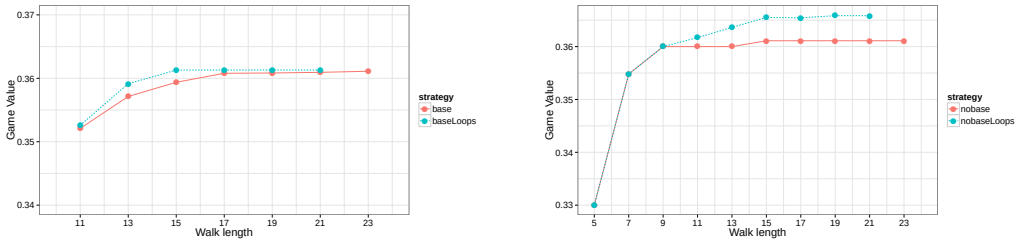
posed algorithms, we modify the graph structure where we allow loops (edges $e(n_i, n_i)$ starting and ending in the same node). An example of such a graph is depicted in Figure 3.6. In this graph, there is a single node through which the Evader can reach the destination node from the origin node. The Defender starts in the base which is connected to a single node n_1 between the origin and destination. We can optionally add a loop edge on n_1 emulating the ability to wait.

Without the ability to wait, the Defender periodically moves between the base and n_1, reaching probability of 0.5 to catch the Evader. With the loop at n_1, the Defender can wait and the probability of catching the Evader is proportional to the amount of time the Defender can spend in n_1, which is given by the maximum allowed walk length l. The probability of the Evader being caught is $p = (l - 3)/(l - 1)$. For $l \to \infty$, the probability of the Evader being caught $p \to 1$ (compared to the probability of 0.5 by being caught by a Defender without the waiting ability).

This example demonstrates the superiority of the Defender with the waiting ability (it can do no worse than the Defender without the waiting ability, as in the worst case, the waiting-able Defender does not have to wait and thus, e.g., lose time). The transfer of this result into the real world has to be done carefully, as the graph could possibly represent a richer environment where the Defender which cannot wait can turn around over the area represented by the node etc. If we look at larger rectangular graphs used in subsection 3.4.3 to evaluate the structure of the player strategy sets, we quantify the differences in Figure 3.7. Even though there are differences in the value of the game, the difference is under 1% for both the Defender with a mobile and static base.

3.4.5 Evaluation of Double Oracle Algorithms

We evaluate the simple variant of the Double Oracle algorithm (SDO)—as proposed by McMahan et al. (2003)—with a best-response oracle for each player, as well as the hierarchical Double Oracle (HDO) algorithm with multiple oracles for each player, i.e, for the Defender, we utilize (if possible) the oracle with a subset of strategies (Section 3.2.3.2) and for the Evader, we use the approximate utility oracle (Section 3.2.3.3). We explore two most interesting versions of the proposed game model: (1) the fixed-base Defender and (2) the mobile-base Defender. The static



(a) Game values for the fixed-base Defender.

(b) Game value for the mobile-base Defender.

Fig. 3.7: Comparison of different Defender mobility models on rectangular graph(Figure 3.3)—with and without the ability to wait in nodes (dashed blue and solid red line respectively). The difference in general larger graphs is marginal.

version of the Defender is explored in greater detail in joint work with Jain et al. (2011a), the unconstrained version of the Defender is not scalable even on simplest graphs (i.e., limited by 8GB RAM and 2 hours, the largest graph instance was a rectangular graph of size 3x4 and Defender's walk length of 5).

We are interested in convergence properties of the algorithms (Section 3.4.5.1) as well as scalability of the algorithms (Section 3.4.5.2).

3.4.5.1 Convergence and Runtime Breakdown

We demonstrate the convergence properties of the algorithm on a rectangular graph of the width $w = 4$,the length $l = 10$, with loops and with the mobile-base Defender allowed walk of the length $l = 13$. We examine the convergence of the simple double oracle with the best-response oracle (Figure 3.8) and with a hierarchy of oracles for the Defender (Figure 3.9). The optimal solution was found in case of the SDO algorithm in 118 seconds and in case of the HDO algorithm in 34 seconds.

First, let us observe the convergence of the algorithm to a Nash Equilibrium, which is depicted in Figure 3.8 for SDO and in Figure 3.9 for HDO. The red and blue lines depict the value of strategies provided by the oracles in each iteration of the algorithm, the black line denoted as *CoreLP* depicts the value of a Nash Equilibrium of the sub-game.

The dotted and dashed lines depict upper and lower bounds for the final game value in the Nash Equilibrium of the full game respectively (highlighted by the green ribbon). Note that we can use the bounds in i-th iteration to estimate maximum error when playing a strategy from the NE of the sub-game in i-th iteration. Given that the resulting value will be always between the bounds, the player's strategy lose no more (w.r.t. the exact game value) than the difference between the bounds. This fact can be used to terminate the algorithm prematurely with quality guarantees. In our case, for the SDO, we can end the algorithm after 54nd and 64th iteration (out of 73) to achieve maximum error of 10% and 5% respectively, saving potentially 10% resp. 6% of total computation time.

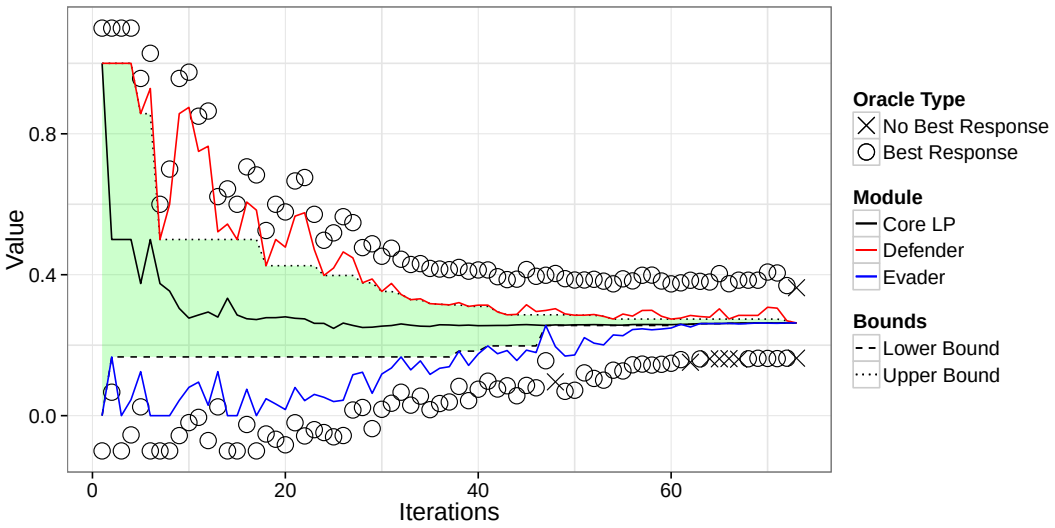


Fig. 3.8: Convergence of SDO algorithm with mobile-base Defender on a rectangular graph of width $w = 4$, length $l = 10$.

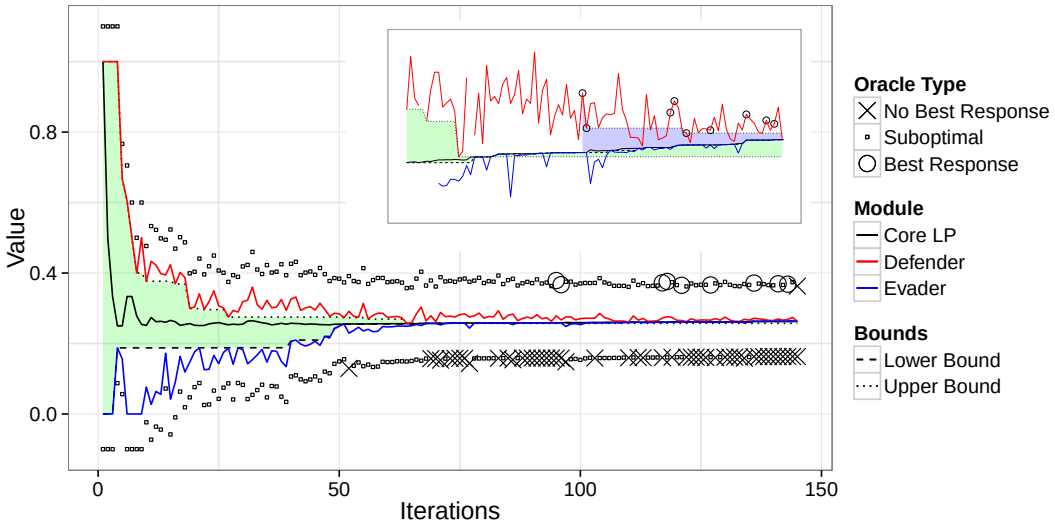


Fig. 3.9: Convergence of HDO algorithm with mobile-base Defender on a rectangular graph of width $w = 4$, length $l = 10$. Number of iterations is higher than in case of the SDO algorithm, however, the algorithm is faster.

In case of the HDO algorithm, we cannot guarantee the maximum error during the algorithm run, as the final game value does not have to lie between the bounds provided by the suboptimal oracles (as depicted in the detail of Figure 3.9): the bounds cross each other because the Defender's oracle provides only suboptimal responses. In this case, the bounds provided by any oracle (denoted by the green ribbon) cannot be taken into account. However, whenever an optimal best-response oracle provides an answer, we can use the best-responses to establish proper

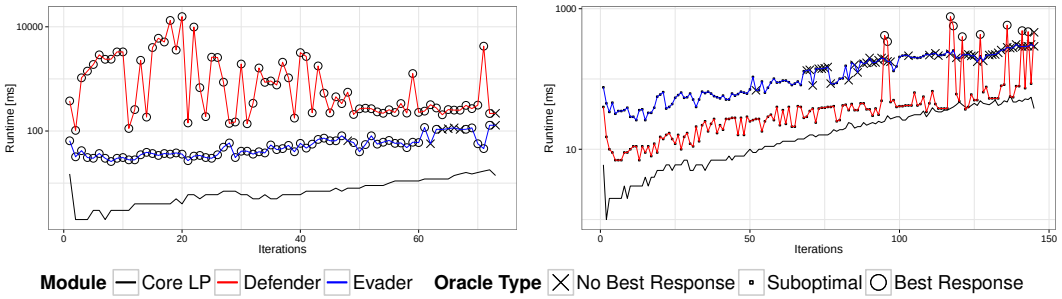


Fig. 3.10: Runtime breakdown of the SDO (left) and HDO algorithm (right).

bounds—this is denoted by the blue ribbon in the detail of respective Figure. This uncertainty in the value of the complete game is the trade-off for the speed of the suboptimal oracle.

Markers above and under convergence lines denote the type of oracle, which has provided the solution (or if no improving solution was found). In case of SDO algorithm, we can observe that most of the time, oracles of both players provided a response not contained in the current sub-game. However, as the strategy space of the Defender is richer, the Defender's Oracle has to provide more strategies. In case of the HDO algorithm, we can observe that the suboptimal oracles can provide response in most cases and that the optimal oracles are not used frequently (and thus serve as a termination check algorithm). For the Evader, the suboptimal oracle is equal to the best-response oracle.

The runtime breakdown of both algorithms is depicted in Figure 3.10. We can see that in case of the SDO algorithm (Figure 3.10 left), the Defender's oracle consumes most of the time and it is thus vital to provide a fast suboptimal alternative, which could increase the runtime—which is done in the HDO algorithm (Figure 3.10 right) where the runtimes of all modules are comparable. The linear program solving the Nash Equilibrium in each stage is fast, however, scales linearly with the number of iterations.

Finally, we explore the size of the support set of both players during the algorithm run (Figure 3.11). As the sub-game increases in each iteration, the ratio of strategies in the support set falls. This means that the oracles generate strategies which are less likely to be in the final strategy set. As expected, the HDO algorithm generates more strategies not present in the final support set, compared to the SDO algorithm. The analysis of the size of the support set could be used to indicate the quality of the oracles (i.e., how good they are at estimating strategies of the final support set).

3.4.5.2 Scalability of Double Oracle Algorithm

We evaluate the double oracle algorithm on a set of rectangular graphs, similar to those depicted in Figure 3.2a. We have tried irregular graphs, such as random scale-free graphs, rectangular graphs with a missing ratio of nodes and edges. The rectangular graphs are the most difficult for the algorithms to compute. The graph is defined by its *width*, the length of the graph is

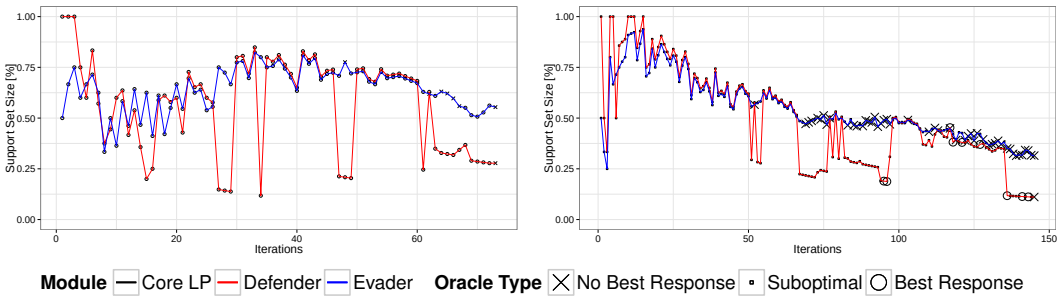


Fig. 3.11: Size of the strategy support sets for both players during the SDO (left) and HDO (right) algorithm execution.

set to $length = 2 \cdot width + 1$. We randomized the interception probabilities on each node and edge, which are sampled from a uniform distribution $U(0, 1)$. The origin nodes are on the left size of the graph, the destination nodes are on the right side of the graph. The base is set to the position $(length - 1/2, \lfloor width/2 \rfloor)$. We varied both the size of the graph (parameterized by the $width$ parameter) and the length of the Defender's walk. We allow one processor on computing nodes with 64bit Java 7 and maximally 6 GB of RAM.

The scalability of the algorithm for the fixed-base Defender is depicted in Figure 3.12. The runtime of the algorithm is exponential both in the size of the graph and in the length of Defender's walk. The number of iterations increases with both the size of the graph and with the length of the Defender's walk, however, the number of iterations depends heavily on the nature of interception probabilities. For equal interception probabilities on all edges and nodes in the graph, the number of iterations is the highest. Structuring the interception probabilities helps in the convergence of the algorithm.

An interesting observation can be made about the oracles themselves. The Evader's oracle scales exponentially with the size of the graph (as the number of paths in the graph depends combinatorially on its size) however, are independent on the length of Defender's walk. The Defender's oracle scales exponentially with the size of the graph as well as with the length of the walk, however, the scalability is worse when increasing the length of the walk. For very short walks (i.e., 5, 7 and 9), the oracle's performance is dominated by the implementation overhead, not by the complexity of the task (which can be observed on the respective graph in Figure 3.12).

A notable phenomenon arises in the situation, where the length of the walk limits the Defender so that he can reach some places only with a small portion of his walk (i.e., touch one side of the graph). In that case, the Evader picks his paths to lead through that area and the Defender cannot intercept the Evader with high enough probability. In that case, the number of iterations is lower (observable on the iteration charts on Figure 3.12).

For the mobile-base Defender scenarios, the performance of the Defender's oracle should be decreased linearly in the number of nodes/edges, however, the performance of other components should stay the same (Figure 3.13). An interesting phenomenon arises in the number of iterations which increase linearly with the size of the graph, which is not true for the fixed-base Defender (due to the phenomenon described in previous paragraph). Again the Evader's oracle

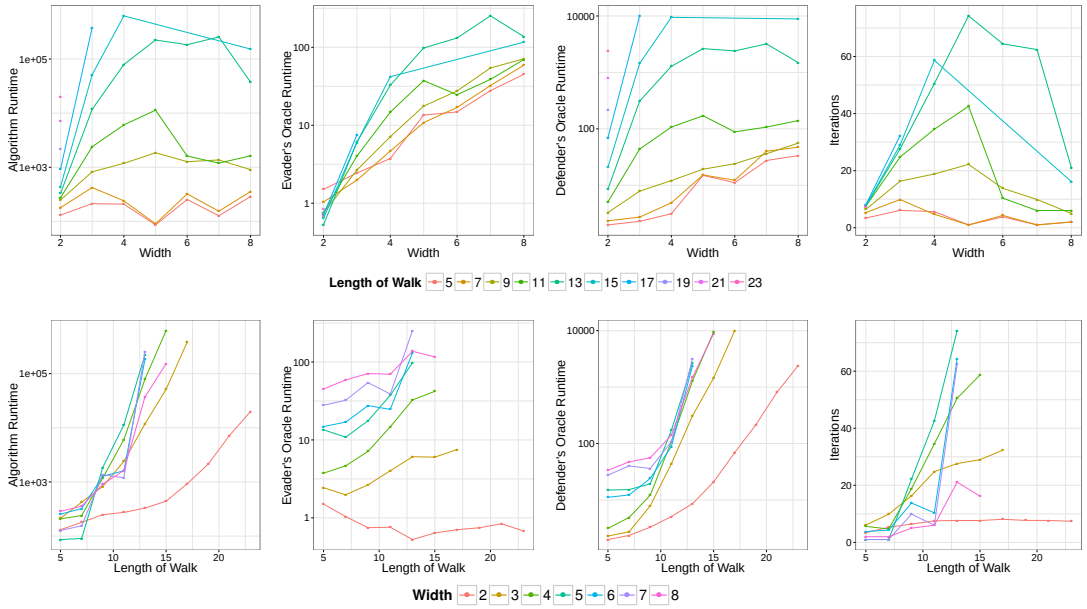


Fig. 3.12: Scalability of the SDO algorithm on rectangular graphs with uncertain encounters for the **fixed-base Defender** while varying the *width* of the graph and the length of the Defender's walk.

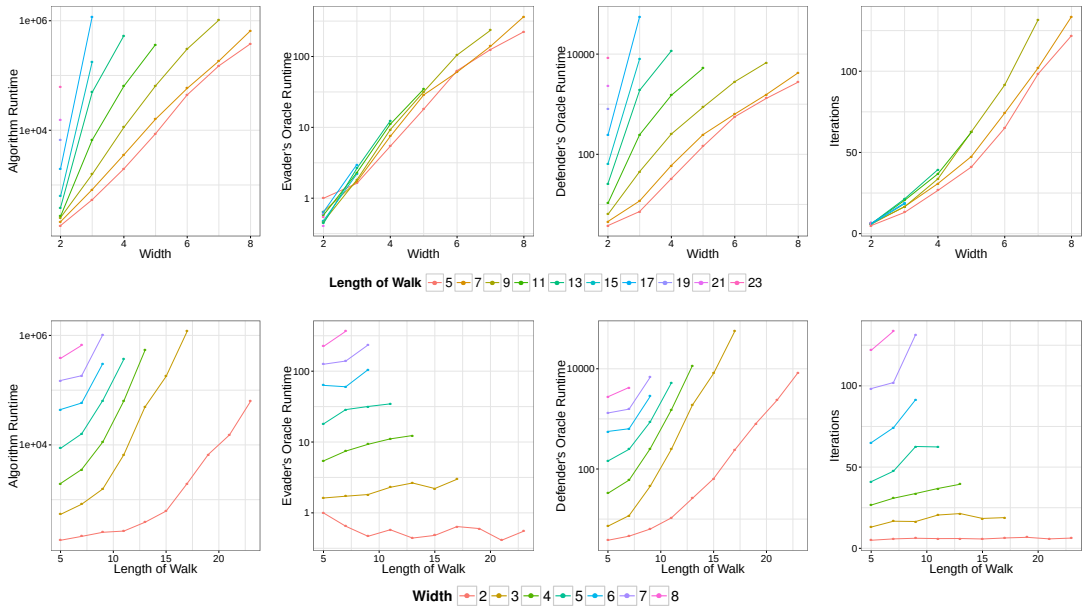
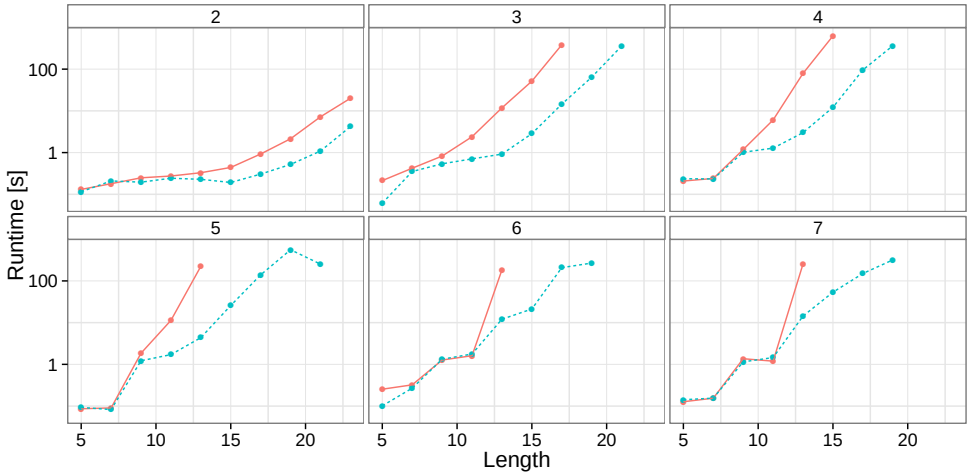
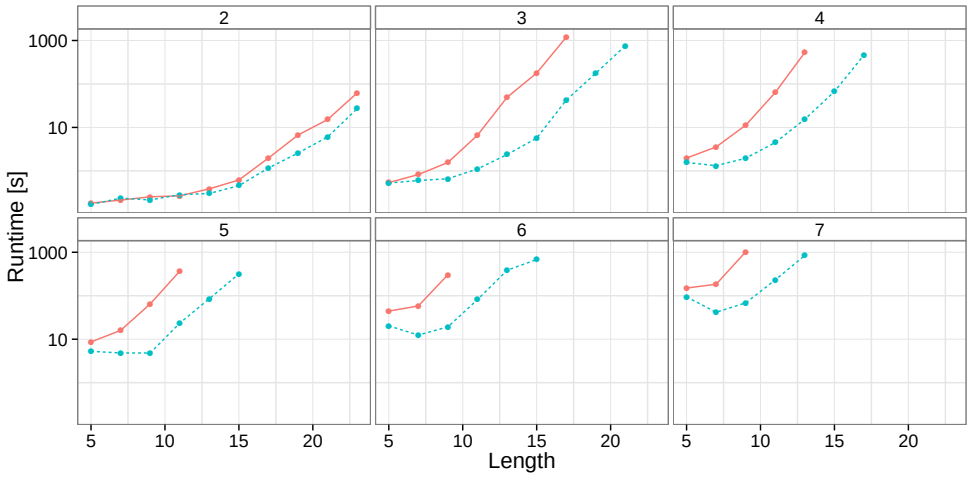


Fig. 3.13: Scalability of the SDO algorithm on rectangular graphs with uncertain encounters for the **mobile-base Defender** while varying the *width* of the graph and the length of the Defender's walk.

computation time scales exponentially with the size of the graph, however, it is independent on the length of Defender's walk.



(a) Performance comparison for **fixed-base** Defender.



(b) Performance comparison for **mobile-base** Defender.

Fig. 3.14: Comparison of runtime (log scale) of Simple Double Oracle algorithm (red full line) and Hierarchical Double Oracle Algorithm (blue dashed line). The graphs are faceted by the *width* of the graph.

To compare the performance of SDO and HDO, we limit ourselves on rectangular graphs and observe the total runtime of both algorithms (depicted in Figure 3.14). We plot only those values, where at least half of experiments of **both** algorithms finished in two hours of computation time. We can observe a steady advantage of the hierarchical double oracle algorithm. The difference is in average five times faster compared over all instances with a differing length of Defender's walk and with a differing graph size, however, for larger instances HDO is faster in order of magnitudes compared to SDO.

3.5 Summary

We have proposed a formal game-theoretic model of a randomized transit through guarded areas. This model can be either used for the design of routes minimizing the probability of encounter or for design of strategies suitable for the defending agent aiming to maximize the probability of encounter, as we compute Nash Equilibrium of the model in which the resulting strategies are optimal (with respect to the utility model) for both players.

Additionally, we have proposed different mobility models which take into account different guarding capabilities of the defending agent. Some of the models can be reduced to existing ones, as in case of the static strategy (Joseph, 2005; Jain et al., 2011a), the mobility models accounting for the base are novel and extend the existing state-of-the-art. Unfortunately, they pose significant computational difficulties due to the non-linearity of utility function (arising from the uncertainty of encounters), which—combined with large strategy spaces for both players—poses a non-trivial problem for Nash Equilibrium computation.

We tackle the NE computation problem by using oracle-based algorithms (McMahan et al., 2003) and design a set of state-of-the-art oracles providing both optimal and suboptimal best-responses—strategies for both players which are used iteratively during the NE computation. Additionally, we broaden the concept of oracles and propose an extension of the double-oracle algorithm for oracle hierarchies, where the suboptimal oracles can be used to significantly speed up the solution without the loss of optimality guarantees.

We have evaluated the game-theoretic model with respect to the solution structure and solution properties, such as the impact of Defender's mobility, the length of defender's walk and the possibility of waiting on one place, focusing mainly on models with base, as they are the main contribution of the thesis. The results show that resulting strategies are not trivial, however, a rule of thumb can be extracted for both the Evader and the Defender: (1) The Evader should randomize fully over possible paths almost uniformly in case of homogeneous environment and with not significantly constrained Defender, slightly preferring routes farther from the base; (2) the Defender walks have to intercept every possible Evader's route at least in one node. The positioning of the base in a bottleneck of the area has a significant advantage, especially when the walk length is restricted. For very long walks (with respect to the width of the area bottleneck), the mobility of the base is not important, for short base, the mobility of the base is vital. The mobility of the Defender has a positive impact on its expected game value, as well as the ability to wait on place.

The scalability of the algorithms differs for different Defender models when varying the size of the graph as well as the length of Defender's walk. Both parameters influence the runtime exponentially in case of the mobile-base Defender; for the static-base Defender, the runtime dependency of the algorithm on the length of Defender's walk is exponential, the dependency on graph width is not trivial and depends on other graph properties. We are able to solve problems for graphs with the size up to approximately hundred of nodes and walks of the length 15–20. The unconstrained model scales very poorly, as the problem is highly under-constrained and no suitable heuristics was found to navigate the algorithm to an optimal solution.

The convergence rate of the double-oracle algorithm is not favorable for ϵ-Nash computation—typically, we can cut-off the algorithm after two thirds of expected iterations (i.e., save one third

of the computation time) to get $\epsilon = 5\%$ of the expected game value. The convergence rate of the hierarchical double-oracle algorithm is better, achieving $\epsilon = 5\%$ after one third of the iterations, however, theoretically, without a final termination check with optimal oracles, we cannot guarantee the final game value.

The formalization of the area transit problem as a security game was published in 2010 (Vaněk et al., 2010) and extended in the following papers (Vaněk et al., 2011; Vaněk, 2011). The extension of double-oracle algorithms was published in 2012 (Vaněk et al., 2012).

The methods described in this chapter are utilized for urban security (Jain et al., 2011a) where we have developed an algorithm for optimal placement of checkpoints in an urban network. Additionally, we slightly modified the presented concept to design an optimal packet sampling to maximize the probability of a malicious packet detection in computer networks (Vaněk et al., 2012b).

Chapter 4
Optimization of Transit Grouping

On the design of algorithms for an optimal grouping of transiting agents.

The Best for the Group comes when everyone in the group does what's best for himself AND the group.

– John Nash

Game-theoretic models of the interaction of two mobile players bring a lot of insight into the structure of the solution as well as the complexity of the computation of the optimal strategy. The scalability of the algorithms is limited even for two-player game models. The scalability of the models with respect to the number of the players is low. We can assume either complete cooperation on either side—we can model the situation as a two player game each of which disposes of multiple units (each of which disposes of the complete strategy set)—or we can assume independent players and solve n-player general sum game in the form of a nonlinear complementarity problem (Shoham and Leyton-Brown, 2008) which is difficult to solve even in the case of the possibility of enumeration of all strategies of the players.

We thus approach the problem from a different perspective, we step-out of the game-theoretic framework and we search for optimal grouping of evading players[1] which have to transit the area in the same time frame. We then consider each group to play an individual game against an unknown adversary. We are primarily motivated by the situation in the Indian Ocean where a number of merchant vessels sail daily through the piracy affected waters and try to avoid an encounter with a pirate ship, fitting the model of a fixed-base or mobile-base Defender. The crucial property of this problem is the low saturation level of the pirate, i.e., the pirate can hijack at most one ship at a time. Which implies that if there are multiple vessels transiting the area, they can form a group to transit together, possibly choosing a route planned using the game-theoretic model. If the group is encountered by a pirate, the effect of the grouping is two-fold: (1) as the pirate can attack only one vessel, the unattacked vessels stay unharmed even after the pirate encounter; (2) the group provides an additional protection in the form of a better positioning of the weakest vessels and better alertness and awareness of the situation around the group.

The problem of effective grouping is not trivial. The vessels arrive continuously to the area and their speeds differ, often significantly. Additionally, some of the vessels are not that vulnerable and even though they would prefer not to encounter a pirate, they are not willing to wait for other vessels to form a group with. These requirements lead to a multi-criterion formulation of the problem.

[1] As we are primarily motivated by the maritime domain and we are thus interested more into scale-up in the number of evaders.

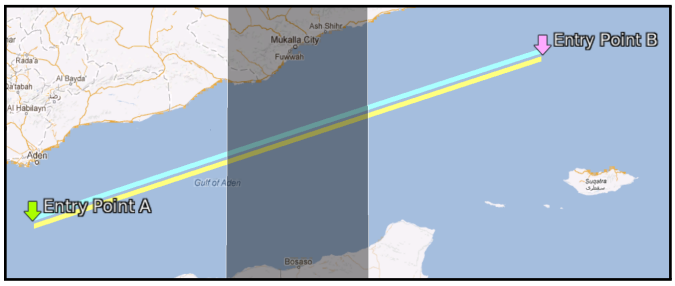


Fig. 4.1: IRTC Corridor in the Gulf of Aden. Ships enter the gulf area from both sides. The most dangerous area in the middle (denoted by the dark zone) is best to be crossed at night.

We will design an algorithm which groups transiting units together and takes into account the constraints given by the transit area (such as entry and exit points), the speed of individual vessels as well as their risk-aversion parameter. We are directly inspired by currently employed group transits of the Gulf of Aden and our abstract formulation allows to optimize grouping schemes for randomized transits as well as for a narrow corridor transit.

4.1 Problem Description and Motivation

This problem is heavily inspired by current transit methods employed in the Gulf of Aden where merchant vessels are grouped into a number of groups according to a *Group Transit Scheme* Intertanko (2009): the Gulf of Aden is a narrow area north off the coast of Somalia with dense merchant traffic in both directions, mostly transporting goods and oil from Asia to Europe. Current spike in Somali-based piracy poses a serious threat to merchant ships transiting the area of the Indian Ocean and the Gulf of Aden – hundreds of hijack attempts have been reported from 2008 till today and tens of ships have been hijacked every year.

4.1.1 International Recommended Transit Corridor

International naval forces were deployed in the area to protect the merchant ships and International Recommended Transit Corridor (IRTC) was established to align the traffic into two lanes—separating the East bound and the West bound traffic—for an easier protection (see Figure 4.1). The East bound lane begins at the Entry Point (EP) A and is oriented along a straight line course of 72 degrees. The West bound lane begins at the Entry Point B and is oriented along a straight line course of 252 degrees Intertanko (2009). The most dangerous area for the transit is approximately in the middle of the corridor (dark area in Figure 4.1).

Table 4.1: Gulf of Aden group transit schedule.

Speed Level	Entry Time for EP A	Entry Time for EP B
10 kn	04:00	18:00
12 kn	08:30	00:01
14 kn	11:30	04:00
16 kn	14:00	08:30
18 kn	16:00	10:00

4.1.2 Group Transit Schemes in the IRTC

In August 2010, a group transit scheme was introduced to further reduce the risk of pirate attacks (illustrated in detail in Intertanko (2009)). Gulf of Aden Group Transits are designed to group ships into different speed groups in order to exploit additional protection and assurance of traveling in a group. Each group is defined by a *speed level* (i.e., published speed at which all ships belonging to the group sail) and *entry time* at which all the ships belonging to the group have to be at the Entry Point. There is one transit per day for each speed group (shown in the Table 4.1).

The entry times for different speed groups to enter the IRTC are calculated so that the groups pass through the most dangerous area at night and they ensure that all ships, regardless of speed, are together at dawn. The group transit scheme thus groups the ships on two tiers: first, the ships are grouped according to their speed and second, the groups are grouped again at the most dangerous area in the Gulf of Aden to transit the area together. This allows the military forces to best position their assets in the area so as to protect ships against piracy and to provide assistance in case of attack.

4.1.3 Group Transit Schemes for Randomized Transits

Group transit schemes were originally proposed by maritime authorities for a single narrow corridor transit, however, the concept, can be directly reused for grouping of agents prior the game-theoretic randomized routing through adversary-controlled areas: typically, all vessels follow the same route when sailing between the same pair of harbors. Randomization allows them to select a different route each time to minimize the probability of a pirate attack; the grouping mechanism allows them to follow this randomized route together with other vessels sailing to the same port, again decreasing a probability of being attacked.

The same constraints on speed distribution of vessels in one group can be posed (as in case of IRTC GTS), some constraints (on the aggregation point) can be left out. The following section formalizes the problem of grouping in a rectangular area (where the length is significantly longer

than the width) and proposes different constraint sets suitable for different modes of a group transit scheme.

4.2 Group Transit Scheme Optimization

The dynamic group transit scheme optimization poses a complex problem as it tackles the grouping problem on a scale of individual ships and thus takes into account constraints imposed by the spatial distribution and different capabilities of the approaching ships. Additionally, we have to deal with a—possibly infinite—stream of approaching ships transiting the corridor. The latter issue can be solved in two ways: (1) the design an algorithm which continuously creates groups and adds ships on the fly (using, e.g., a rolling horizon approach) or (2) to cluster approaching ships into clusters and compute group transit separately for each cluster. We will focus on the latter approach, motivated by the specifics of dangerous areas transits. The vessels approach the dangerous areas in similar times (due to the fact that some parts of the areas are best to be transited at night) and are thus clustered to some degree prior the area transit[2]

4.2.1 Problem Abstraction

Figure 4.2 depicts the abstraction of the environment for the dynamic group transit formation: the ships transit the area from left to right, reaching the *approach zone* first. Only the ships in the approach zone (of length L_a) are considered for grouping[3]. The groups are established at the Entry point A and they follow the corridor (of length L). Additionally, inside the corridor, there is an *aggregation point* π, in which all the groups have to meet. Moreover, the aggregation point has a specific time of the day U assigned, at which the groups have to arrive at the point. The last two conditions place additional constraints on the problem and can be relaxed in similar models without such requirements.

To express the advantage of a ship being in a group, we use a term *risk*. If a ship does not take part in the group transit, it is facing an increased risk of hijack, because it does not obey the required time of transit of the most dangerous area or it does not sail at the recommended group speed (or both). We model the increased risk with a *risk aversion* $R_j \in [0, 1]$ parameter, set individually for each ship. For $R_j = 0$, the ship does not gain anything by being in a group, for $R_j = 1$, the ship suffers maximum penalty for being left out of the GTS.

[2] similar situation naturally arises for the Gulf of Aden transit: typically, vast majority of the ships arrive from or continue its journey to the Strait of Suez. Strait of Suez employs its own traffic system, structuring all ships transiting the strait into groups which transit the strait once per day. For details, see Strait of Suez Traffic System website: `http://www.suezcanal.gov.eg/sc.aspx?show=13`. This means, that the daily transit of ships through the Strait of Suez forms a cluster for which we design the dynamic group transit scheme.

[3] The length of the approach zone determines the number of ships for the group transit scheme computation and its length has a significant impact on the size of the problem (and thus on the ability to compute the optimal solution).

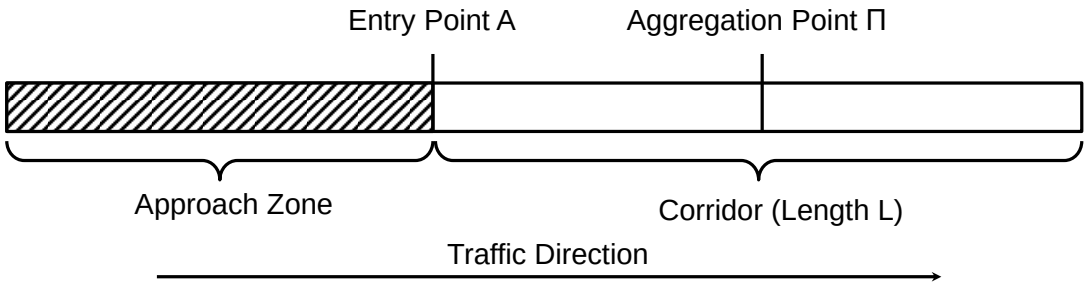


Fig. 4.2: Abstraction of the dynamic group transit setting.

Table 4.2: Notation used in mathematical model of the dynamic GTS.

Symbol	**Definition**
$s \in \mathbb{N}$	number of speed levels
$n \in \mathbb{N}$	number of ships in the approach zone
Π	coordinates of the aggregation point [GPS]
$\nu \in \mathbb{R}^+$	minimum approach speed of any ship [kn]
$\mu \in \mathbb{N}$	minimum group size
$\tau \in \mathbb{R}^+$	current time (typically set to 0)
$\Delta V \in \mathbb{R}^+$	maximum allowed speed difference in one group [kn]
A	coordinates of the entry point [GPS]
$P \in \mathbb{R}^n$	position of ships (P_j is position of the j-th ship) [GPS]
$V \in \mathbb{R}^n$	speed of ships (V_j is speed of the j-th ship) [kn]
$T \in \mathbb{R}^n$	the earliest time the ship can arrive to the entry point A [h]
$R \in [0,1]^n$	risk value, expresses the risk aversion of a ship (R_j is risk aversion oftje j-th ship)
$U \in \mathbb{R}^m$	vector of admissible times to be at the aggregation point Π w.r.t. the minimum approach speed ν and current time τ [h]
$L \in \mathbb{R}^+$	length of the corridor [nm]

4.2.2 Formal Model

The parameters and notation used throughout the section are summarized in the Table 4.2. We formulate the problem of dynamic group transit as a bi-objective optimization problem with two objective functions: *delay* and *risk*, which are subject to a set of constraints on the schedule properties and ship abilities:

$$\min \quad (delay(x), risk(x)) \tag{4.1}$$

$$\text{s.t.} \quad x \in X \tag{4.2}$$

where X is a set of feasible solutions defined by a set of constraints. We scalarize the multi-objective criterion into a single linear combination of the two functions, weighted by γ parameter (see Section 2.3.4.2 for details):

$$\min \qquad \gamma \cdot \mathcal{R} + (1 - \gamma) \cdot D \tag{4.3}$$

where the delay function D has two components $D = D_a + D_t$; D_a is the delay caused by the lower speed of the ships in the approach zone and D_t is the delay caused by lower speeds of grouped ships when transiting the corridor. $\mathcal{R}$ is the overall risk aggregated over all ships. The solution is not unique due to the counter-going objective functions, as delay and risk are inversely proportional; we thus look for a set of Pareto-optimal solutions—i.e., the Pareto front—by varying the $\gamma \in [0, 1]$ parameter.

We define the terms in the criterion 4.3 as:

$$D_a = \sum_{j \in n} \omega_j \tag{4.4}$$

$$D_t = \sum_{i} \sum_{j \in n} \sum_{k \in m} U_k \cdot y_{ijk} \tag{4.5}$$

$$\mathcal{R} = \sum_{j \in n} R_j \cdot (1 - \sum_{i \in s} x_{ij}) \tag{4.6}$$

where ω_j is the entry time of j-th ship at the entry point A (capturing the approach delay D_a), y_{ijk} is an indicator variable set to 1 if the j-th ship in the i-th group would sail through the aggregation point Π at time U_k (capturing the transit delay D_t). The equation (4.6) expresses the risk as a sum of individual risks of all ships which are not assigned to any group (variable x_{ij} indicates that j-th ship is assigned to i-th group and the expression $(1 - \sum_{i \in s} x_{ij})$ is 1 if and only if j-th ship is not assigned to any group).

Criterion (4.3) is optimized subject to a number of constraints capturing structure of the grouping mechanism. First, we impose a set of constraints for correct grouping, expressed as:

$$\sum_{i \in s} x_{ij} \leq 1 \qquad \forall j \in n \tag{4.7}$$

$$\sum_{j \in n} x_{ij} \geq \mu - M \cdot a_i \qquad \forall i \in s \tag{4.8}$$

$$\sum_{j \in n} x_{ij} \leq 0 + M \cdot (1 - a_i) \qquad \forall i \in s \tag{4.9}$$

$$x_{ij} \in \{0, 1\} \qquad a_i \in \{0, 1\}$$

Constraint (4.7) specifies, that each ship can be at most in one group, Constraint (4.8) restricts the group size to be greater than or equal to a pre-specified minimum group size μ and Constraint (4.9) allows the group to be of size 0 (thus having no ship assigned). The binary variable a_i indicates that only Constraint (4.8) or (4.9) can be active at the same time for i-th group and M is a large number.

To capture the requirement of group formation at the entry point A, we introduce variable w_i capturing entry time of i-th group with the following constraints:

$$\omega_j \geq w_i - M(1 - x_{ij}) \qquad \forall i \in s; \forall j \in n \qquad (4.10)$$

$$w_i \geq -(1 - x_{ij}) \cdot M + \tau + T_j \qquad \forall i \in s; \forall j \in n \qquad (4.11)$$

$$w_i \leq (1 - x_{ij}) \cdot M + \tau + \frac{|P_j - A|}{\nu} \qquad \forall i \in s; \forall j \in n \qquad (4.12)$$

$$w_i \in \mathbb{R}^+ \qquad \omega_j \in \mathbb{R}^+$$

Constraint (4.10) links the entry time ω_j of the j-th ship to the i-th group, i.e., if the j-th ship is in the i-th group, then (due to the minimization criterion) $\omega_j = w_i$. Constraint (4.11) expresses the fact, that the group cannot be established at the entry point A earlier than every ship reaches the entry point A (relative to the current time τ), posing a lower bound restriction on w_i. For j-th ship, the earliest arrival time T_j can be pre-computed as $T_j = |P_j - A|/V_j$[4]. Constraint (4.12) poses an upper bound restriction on w_i by assuring that no ship violates the minimum approach speed requirement (given by ν)[5].

Finally, we incorporate the restriction given by the requirement to aggregate the groups at the aggregation point Π:

$$\sum_{k \in m} y_{ijk} \cdot U_k - w_i \geq \frac{|\Pi - A|}{S_j} \cdot x_{ij} \qquad \forall i \in s; \forall j \in n \qquad (4.13)$$

$$\sum_{k \in m} y_{ijk} \cdot U_k - w_i \leq \frac{|\Pi - A|}{S_j - \Delta V} + M \cdot (1 - x_{ij}) \qquad \forall i \in s; \forall j \in n \qquad (4.14)$$

$$\sum_{k \in m} y_{ijk} = 1 \qquad \forall i \in s; \forall j \in n \qquad (4.15)$$

$$y_{ijk} \in \{0, 1\}$$

As stated, variable y_{ijk} indicates that the j-th ship in i-th group will be present at the aggregation point Π at time U_k. Constraint (4.13) states that the speed of a group cannot exceed the speed of any ship which belongs to this group (i.e., the time needed for the i-th group to reach Π is given by $\sum_{k \in m} y_{ijk} \cdot U_k - w_i$ and the time needed for the j-th ship to reach Π is given by $|\Pi - A|/S_j$). Constraint (4.14) restricts the spread of the speeds of ships in one group to be at most ΔV. Constraint (4.15) states that only one time of passing Π is admissible for any group.

The equations above fully capture the problem, however, we can add additional redundant constraints to speed-up the solution process:

$$x_{ij} \neq x_{il} \quad \text{iff} \quad |s_j - s_l| > 2 \cdot \Delta V \qquad \forall i \in s; j, l \in n; j \neq l \qquad (4.16)$$

$$x_{ij} \neq x_{il} \quad \text{iff} \quad et_j > lt_l \quad \text{OR} \quad et_l > lt_j \qquad \forall i \in s; j, l \in n; j \neq l \qquad (4.17)$$

i.e., no two ships can be in one group, if the difference of their speed is greater than $2 \cdot \Delta V$ and no two ships can be in one group if the earliest time et_j of one ship to arrive to A is greater than the latest time lt_l when the other ship has to leave A (given by the minimum

[4] The distance between two points given in GPS coordinates is computed to be in nautical miles to correspond with other units.

[5] It is trivial to account for different minimum approach speeds for each ship. For each ship, we can introduce ν_j variable and directly use it in constraint (4.12).

approach speed requirement) or vice versa. The times are computed as $et_j = dist(P_j, A)/s_j$ and $lt_j = dist(P_j, A)/\nu$, where $dist(P_j, A)$ is orthodromic shortest path between P_j and A.

4.2.3 Group Transit Scheme Relaxations

Having the full problem formulation, we can relax or restrict any of the constraints to customize the dynamic GTS. Variations with minor modifications as well as with more fundamental ones—removing restrictions posed by the aggregation point Π and/or by the approach buffer (i.e., the groups are assembled directly at the entry point A)—have been considered.

4.2.3.1 Mandatory grouping

We modify Constraint (4.7) to account for a mandatory assignment of every ship into any group, i.e., $\sum_{i \in s} x_{ij} = 1$. Having this restriction, the risk summand in the criterion function is redundant and can be left out. However, the problem cannot be always feasible, due to the requirements of the minimum approach speed (Constraint (4.12)) and maximum speed difference in one group (Constraint (4.14)). These constraints have to be either relaxed or the program has to be solved multiple times while decreasing the minimum allowed approach speed ν and/or increasing maximum speed difference ΔV.

4.2.3.2 Group size limit

We can additionally limit the size of the group to be at most η[6] by introducing a constraint $\sum_{j \in n} x_{ij} \leq \eta$. In this case, the number of groups should be proportionally increased to create enough groups and not to be penalized for the risk of ships that cannot be placed in any group because of this constraint.

4.2.3.3 No Aggregation Point

By relaxing constraints (4.13) – (4.15), we do not pose any aggregation requirements on the group transit scheme and the groups are established at the entry point A as soon as possible. Additionally, all groups sail at the group speed equal to the speed of the slowest ship of the group. To reflect these facts in the mathematical model, we consider explicit group speeds which are not linked to the aggregation point. We reformulate the transit delay D_t to:

$$D_t = \sum_{j \in n} \sigma_j \cdot L \tag{4.18}$$

[6] Where $\eta \geq \mu$. When setting the $\eta = \mu$ we get groups of the exact size.

where L is the length of the transit corridor and σ_j is the inverse transit speed[7] of j-th ship in a group, subject to the following constraints:

$$\sigma_j \geq g_i - M \cdot (1 - x_{ij}) \qquad \forall i \in s, j \in n \tag{4.19}$$

$$g_i \cdot L \geq x_{ij} \cdot \frac{L}{s_j} \qquad \forall i \in s, j \in n \tag{4.20}$$

$$g_i \cdot L \leq \frac{L}{(s_j - \Delta V)} + (1 - x_{ij}) \cdot M \qquad \forall i \in s, j \in n \tag{4.21}$$

$$g_i \in \mathbb{R}^+ \qquad \sigma_j \in \mathbb{R}^+ \tag{4.22}$$

Constrain (4.19) links the speed of the j-th ship to the inversed speed of the i-th group g_i, constraint (4.20) restricts the group speed g_i to be lower than the speed of the slowest ship in the group and constraint (4.21) allows one group to have ships with maximum speed difference ΔV.

4.2.3.4 No Approach Zone

For some dynamic group transit schemes, we do not have to consider the approach zone, i.e., we consider all ships to be at point A at the beginning of the grouping, similar to the Convoy Scheduling Problem formulation. We can then relax constraints (4.11) and (4.12). However, it is recommended to leave variables w_i in constraints (4.13) and (4.14) to allow groups to start their route with a delay w_i. Otherwise, if the meeting times U_k are too sparse, a solution may not be found.

4.2.3.5 No Aggregation Point, No Approach Zone

Finally, we can relax the restriction on the aggregation inside the corridor and consider the ships to be ready at point A. The delay caused is given only by the transit delay D_t in the form of Equation (4.18), which is restricted by constraints (4.19) – (4.22). Together with constraints (4.7) – (4.9), they form a complete constraint set for this problem[8].

4.3 Evaluation

We evaluate the quality of solutions of the Group Transit Scheme (GTS) optimization problem. We use both synthetic and real-world data and we are interested in the structure of the solution, a relative improvement against the current group transit scheme, and in the scalability of algorithms for dynamic grouping. The algorithms were evaluated on a Quad-core 64-bit PC with

[7] We define variables as inverse to keep the mathematical formulations linear. The final speed of j-th ship a in a group is equal to $\frac{1}{\sigma_j}$. Term $\sigma_j \cdot L$ expresses directly the corridor transit time of j-th ship.

[8] We do not have to consider group delays w_i before transit, as the transit is not restricted by the aggregation times U_k.

4GB available RAM; implementation was done in Java 1.7 and we used CPLEX 12.3 to solve the mathematical programs.

We compare 4 variants of the algorithm designed in Section 4.2.

1. *AggregationApproach* – considering constraints of the original problem statement (aggregation at point Π and approach zone considered).
2. *Approach* – considering only approach constraints, as described in subsection 4.2.3.3 (no aggregation point Π; ships are spread through the approach zone).
3. *Aggregation* – considering only aggregation constraints, as described in subsection 4.2.3.4 (aggregation at point Π; all ship starting at point A).
4. *None* – considering no aggregation and no approach constraints, as described in subsection 4.2.3.5 (no aggregation point Π; all ship starting at point A).

We look at the structure of the optimal dynamic group transit. We further explore the Pareto frontier and evaluate the scalability of the algorithm with respect to main algorithm parameters. Finally, we compare the dynamic group transit to the currently deployed fixed schedule.

We generate synthetic scenarios where ships are spread uniformly in the approach zone which should imitate the hardest possible conditions (i.e., the worst case) for successful grouping. Note that in real-world, the ships would be often significantly clustered and will be grouped farther from the entry point A, thus the solutions would be significantly better (in terms of number of ships grouped). We generate ship speeds from a uniform distribution, again emulating worst case possible. If not stated otherwise, tested values of parameters are set by default to: length of the approach zone $L_a = 600$ nm, number of ships $n = 25$, number of groups $s = 5$, minimum approach speed $\nu = 8$ kn, maximum speed difference in a group $\Delta V = 2$ kn and minimum group size $\mu = 2$.

4.3.1 Structure of the Solution

The structure of the solution differs significantly when computed by each algorithm for default parameter setting and risk weight $\gamma = 0.5$.

The solution is displayed on two figures for each algorithm (see Figure 4.3). We can observe that for all algorithms except *None*, some ships are left ungrouped (red squares in plots). In case of the *AggregationApproach* algorithm, the ships which are either slow or very close to the entry point A cannot slow too much to meet the minimum approach speed constraint. For the *Approach* algorithm, not all ships can be grouped because of the minimum approach speed constraint as well.

Some groups in solutions computed by *AggregationApproach* and *Aggregation* algorithms have slower speed than the speed of the slowest ship in the group. This is caused by the restriction of a specific set of times of arrival at the aggregation point Π.

Solutions computed by *Aggregation* and *None* algorithms neglect the arrival positions of the ships, considering all ships to be at the entry point A.

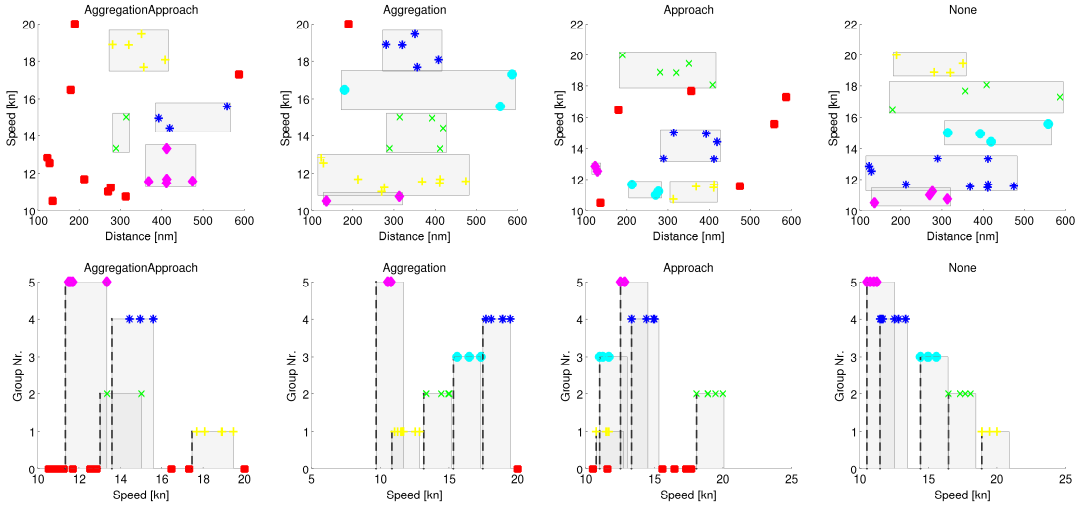


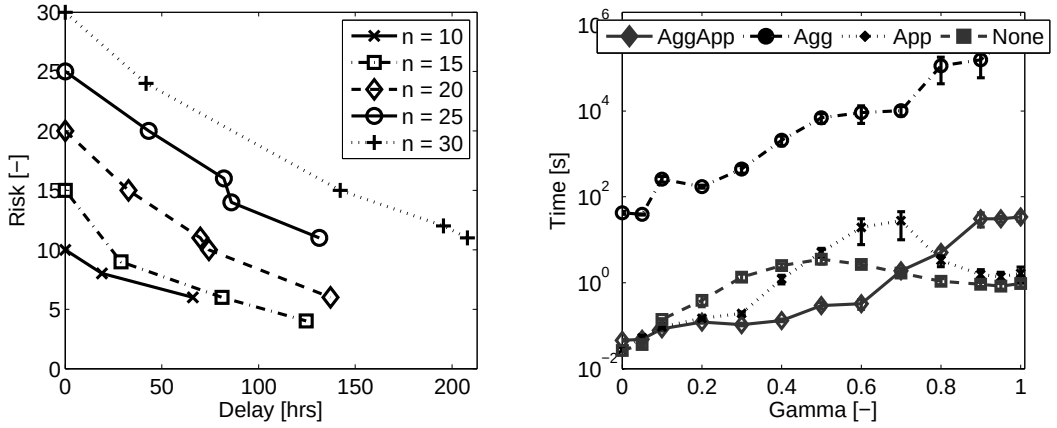
Fig. 4.3: Structure of Solutions for different constraint sets. The upper row of graphs show grouping in speed and distance from entry point dimensions. For *Aggregation* and *None* algorithms, the distance dimension is not used. The lower row shows group speeds (dashed lines) and their range (i.e., maximum speed difference in one group.). The ships which are not assigned to any group are denoted as red squares.

4.3.2 Pareto Frontier Evaluation

In the optimization problem defined by a Criterion (4.3), we weigh two functions through a parameter γ. While varying γ, we reach different solutions with different risk and delay caused by the dynamic GTS. Figure 4.4a captures Pareto frontiers computed by *AggregationApproach* algorithm for default parameter values for $n = \{10, 15, 20, 25, 30\}$ ships. The Pareto frontier curves are almost linear with non-smooth transitions due to the binary property of group membership. The scalability of algorithms with respect to γ is described in the next Section.

4.3.3 Scalability

The performance of the algorithm depends on the number of parameters: the number of ships, the number of groups, risk aversion coefficient, maximum speed difference in one group, minimum approach speed and minimum group size. Here we focus on the first three parameters as they have the greatest impact on the speed of the algorithm. We have generated 100 random instances of the problem for each parameter setting and averaged the computation time needed to find a solution. The dependency of solution time on the number of ships and the number of groups is depicted in Figure 4.5. For all 4 algorithms, we are able to find solutions for up to 20 ships and 4 groups. For the *Aggregation* algorithm, when increasing the number of ships or number of groups, some generated problem instances are not solvable within 4GB of RAM.



(a) (b)

Fig. 4.4: (a) Pareto frontiers for number of ships $n = \{10, 15, 20, 25, 30\}$. Solutions which are not Pareto efficient, would lie *above* the curve, as the criterion is minimized. (b) Scalability of the algorithms with respect to risk weight coefficient γ. Note the significantly lower performance of the *Aggregation* algorithm.

For the current IRTC transit constraints, the *AggregationApproach* algorithm is able to find solution for 25 ships and 5 groups in under 10 minutes in average. The largest problems instances solved in hundreds of minutes by *AggregationApproach* were with 30 ships and 6 groups, however, in some cases, the memory was the bottleneck and the solution could not be found. The maximum solvable problem size is heavily dependent on the solver of the program.

The scalability of the algorithms with respect to the risk weight coefficient γ is captured in Figure 4.4b. We varied γ from 0 to 1 for a setting with $n = 20$ ships and $s = 5$ groups, while having all other parameters fixed at default values. For larger risk weights (i.e., $\gamma > 0.9$), the time needed to find a solution does not vary significantly anymore, as the risk summand in the criterion outweigh the delay summand and if a ship can be in a group, the algorithm places the ship in a group.

Algorithms without aggregation constraints peak for intermediate values of γ (*Approach* for $\gamma = 0.7$ and *None* for $\gamma = 0.5$). After this peak, when increasing γ, the computations time is lower again, converging to a constant value. Algorithms with aggregation constraints (*AggregationApproach* and *Aggregation*) do not have the property described above and converge to a constant value.

4.3.4 IRTC Case Study

To compare the dynamic GTS with current GTS, we measured the delay caused and the number of ships left ungrouped (by either not satisfying the minimum approach speed constraint or – in case of the current GTS– being alone in the group) for both the grouping transit schemes. We have set the minimum approach speed $\nu = 8$ for both schemes. For dynamic GTS, we have set the risk weight coefficient to $\gamma = 1$ to force maximum grouping with possibly increased

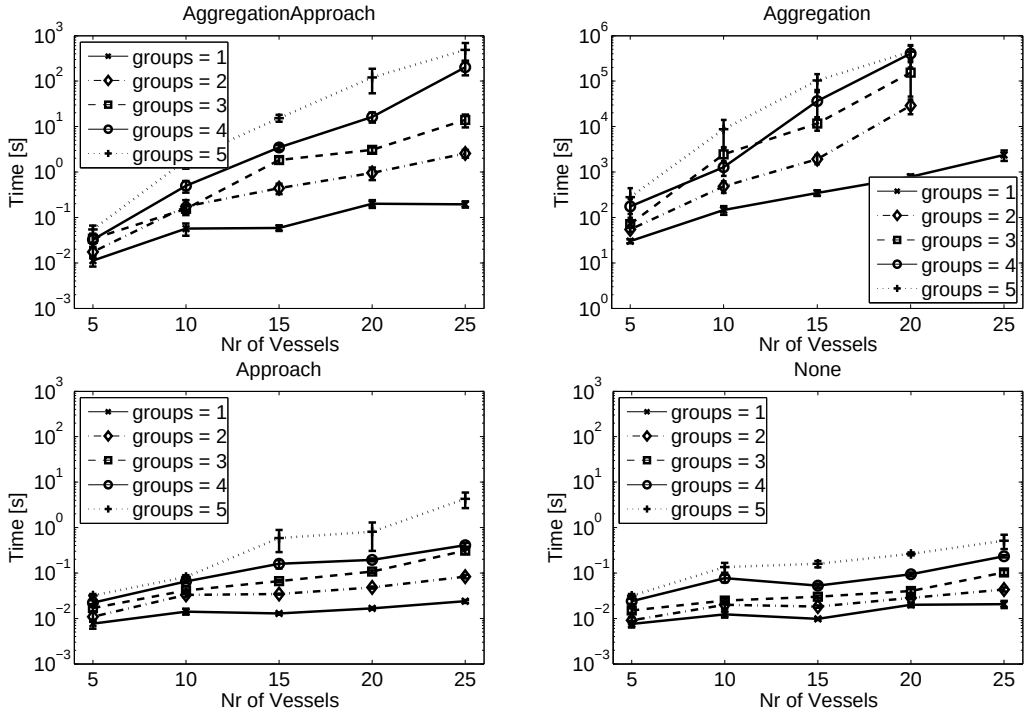
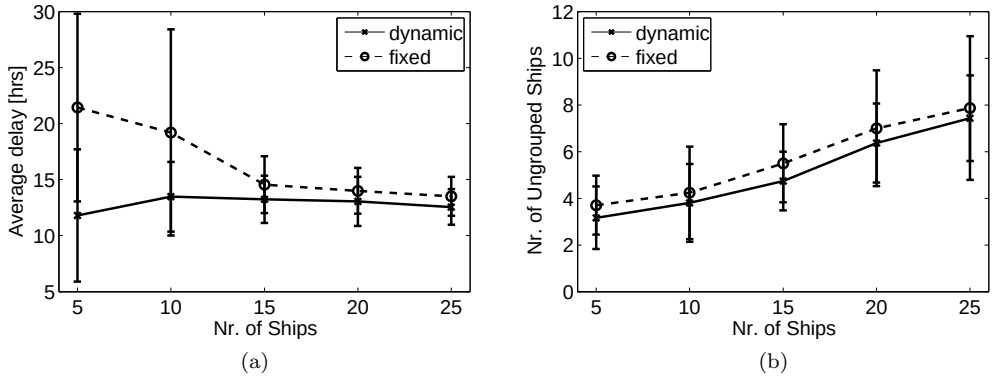


Fig. 4.5: Scalability of Algorithms while modifying number of ships and number of groups. For the *Approach* algorithm, the y-axis is in different range. Not that the error bars depict standard error.



(a) (b)

Fig. 4.6: (a) Comparison of delays caused by the currently deployed GTS and by the dynamic GTS (lower is better); (b) comparison of number of ungrouped ships for fixed and dynamic GTS (lower is better). The error bars depict standard deviation.

delay, the number of groups to be created to $s = 5$ and the maximum speed difference in one group to $\Delta V = 2$ to be equal the currently deployed fixed GTS. We have varied the number of approaching ships from 5 to 25 and run 100 samples for each setting.

The results are depicted in Figure 4.6. We can observe, that the delay caused by the dynamic GTS is lower for less ships, however, as the number of approaching ships increases, the delay

caused by both groupings is comparable, as all groups are filled and given constraints given by the aggregation point Π, the delay caused is converging to a stable value. The average time saved by the dynamic grouping is approximately 1 hour, equaling to approximately 8% of the transit time[9]. In Figure 4.6b we can compare the average number of ungrouped ships for both schemes. Dynamic GTS has steadily lower number of ungrouped ships (which is preferable). For high number of ships, the ungrouped ships typical do not meet the minimum approach speed requirement.

4.4 Summary

Due to inherent scalability problems in the number of independent players in a game-theoretic model, we step outside the game-theoretic framework and propose grouping schemes which aggregate transiting agents into groups which transit the problematic area together. The groups are then considered to be a single agent playing the game. To find optimal grouping for a set of agents with differing attributes, we inspire ourselves with group transit schemes currently deployed in the Gulf of Aden and we design a set of grouping models taking into account both speed and risk-aversion of (i.e., the willingness to transit the area in a group) the transiting agents.

We propose a state-of-the-art grouping model with different constraint sets, considering area entry points and aggregation points. We model the problem as a bi-objective optimization problem and propose a linear mixed-integer program with a scalarized criterion function which is able to capture trade-off between the delay caused and the willingness to transit the area alone.

The structure of optimal groups of the grouping problem is non-trivial, most frequently grouping ships with speeds in the beginning and in the middle of the approach zone and ships which are further from the entry point of the area (especially in case of an aggregation point constraint). The algorithm scales to tens of vessels, where the number of vessels influences the runtime of the algorithm the most—the dependency is exponential, which is expected, given the NP-hardness of the problem.

Finally, we compare the delay caused and the number of vessels grouped by our grouping mechanism with the currently deployed fixed grouping scheme for the Gulf of Aden transit—we show the superiority of our approach in both the delay caused and the number of vessels grouped.

The first, static model of group transit optimization in corridors placed in pirate infested areas was proposed in 2011 (Hrstka and Vaněk, 2011). The dynamic version of the problem was first described in 2013 (Vaněk et al., 2013b) later extended to different constraint sets (Vaněk et al., 2013a).

[9] Observe rather high standard deviation for 5 and 10 ships which is caused by a low number of groups created. In many cases, there is only a single group; for a single group, the delay can be either high or low, depending on the speed level of the group. For higher number of ships, higher number of groups is created and the delay averages out over all speed levels of all the groups.

Chapter 5
Agent-Based Model of Maritime Piracy

On the design of the agent-based model of maritime traffic

> *A good simulation, be it a religious myth or scientific theory, gives us a sense of mastery over experience. To represent something symbolically, as we do when we speak or write, is somehow to capture it, thus making it one's own. But with this appropriation comes the realization that we have denied the immediacy of reality and that in creating a substitute we have but spun another thread in the web of our grand illusion.*
>
> *–Heinz R(udolf) Pagels*

Having designed mathematical models capturing the game-theoretic interaction between two agents, algorithms allowing to get a solution in the form of a strategy and optimized grouping mechanism to cope with multiple independent players, the next step is to validate its robustness and suitability for the real-world deployment. This step is often neglected in similar problems (Jain et al., 2010b; Pita et al., 2009) however, we consider this step to be crucial before the deployment in real-world conditions, where not all assumptions of the mathematical model are met.

We thus propose an additional step between abstract mathematical model and the real-world deployment: simulation-based validation. The simulation can bridge the gap between a rigorous (however expression-constrained) mathematical model and an uncertain, unpredictable (and expression-rich) real world. Agent-based simulations are especially suitable for this approach—they provide a rich set of tools to capture the realistic behavior of involved actors and incorporate solutions from the mathematical model.

We develop AgentC, an agent-based simulation of the maritime traffic in piracy infested areas which is trying to emulate realistic behavior of all actors involved. We then implement agents with decision making processes derived from the game-theoretic model—however, the agents act in a more realistic environment where not all conditions—e.g., discrete time, synchronous movement, equal speeds etc.—assumed by the mathematical model are met. This allows us to quantify deviations from the theoretical guarantees given by the mathematical models and assess the robustness of the model.

5.1 Problem Description

Maritime traffic simulation can serve to many purposes: analysis of traffic properties (such as congestion, transit scheme effectiveness, quality of routing/route planning), trace generation (synthetic data generation) and others. In our case—even though the scope of use of the designed

simulation is very broad—we use the simulation for purposes of a simulation-based validation: we want to assess the quality (such as robustness and effectiveness) of the mathematical models designed in previous sections in a more realistic environment.

5.1.1 Why Agent-Based Simulation?

Klügl (2009) summarizes advantages of agent-based simulations (ABS) compared to traditional approaches—such as a macroscopic simulation (Van Dyke Parunak et al., 1998), event-based simulations (Davidsson, 2001), queuing networks (Allen, 1990), petri nets (Murata, 1989), cellular automata (CAs, 2007) and others—as follows: (1) ABS is suitable for dynamic and flexible local interaction which can capture the heterogeneousness of the environment and actors (and their populations), (2) ABS captures perfectly locality a situatedness (together with constraints arising from these concepts) of actors, (3) ABS is able to capture decision-making processes on different levels (such as asset allocations vs. reaction on a pirate attack), (4) ABS can capture partial group-based interaction (utilized by merchant vessels) and others.

5.1.2 Agent-based Modeling Methodology

We employ an *individual-centric* modeling approach, in which the behavior of the modeled system is represented at the micro-level of individual vessels. Vessels are modeled as autonomous agents (Russell et al., 2010) capable of moving freely within the navigation boundaries of ocean waters while interacting with the maritime environment, other vessel agents and other actors (such as shipping operators or traffic coordinators).

Based on the literature and discussions with domain experts, we identified *merchant vessels*, *pirate vessels* and *navy vessels* as main vessel classes which are therefore explicitly represented in our model as vessel agents. For most of the time, each vessel agent pursues its individual goals, however, there are situations where multiple vessel agents interact—such interactions are either non-cooperative (such as pirate attacks or navy warship counter-pirate interventions) or cooperative (such as merchant vessel agents' requests for help to navy vessel agents). Vessel agent interactions play a critical role in the dynamics of maritime piracy and make the agent-based, micro-simulation approach vital for accurately modeling the effect of piracy on maritime transportation, primarily because it allows capturing phenomena and providing the detail of analysis not attainable with macro-level equation-based methods (Van Dyke Parunak et al., 1998).

In line with the agent-based modeling approach, the model for each class of vessels consists of an individual *vessel behavior model* and a *vessel population model*. The vessel behavior model represents the executable behavior of an individual vessel agent; such behavior can depend on *vessel parameters* assigned to each individual vessel. The vessel population model specifies how many vessels of each class are generated and how the values of vessel parameters are assigned.

The choice of vessel parameters is based on relevant literature (Bruzzone et al., 2011; Tsilis, 2011; Decraene et al., 2010) and was discussed with domain experts; their influence (i.e., importance) was also explored using sensitivity analysis (described in Section 5.3.2).

5.1.3 Simulation Scope

To capture all essential parts of the maritime piracy phenomenon, we have to consider all actors involved, together with the environment influencing their behavior and strategies. We decompose the model into following parts:

1. **Environment model** – captures all structures representing data about the environment, such as coastal lines, areas of navigable waters, weather conditions and sea conditions and transiting schemes deployed in the area (such as corridors and grouping schemes).
2. **Merchant traffic model** – captures the population of merchant vessels sailing through the area and individual behavior in the form of a route planner and actions related to piracy activity.
3. **Piracy model** – captures the areas of pirate activity, their main hubs and the population as well as the decision making process when choosing an attack area and interaction with other agents.
4. **Navy operations model** – captures the allocation mechanism of assets in the area and the decision making process when a pirate activity is reported

These models can be parameterized to be configurable for various scenarios or there can be multiple different models for each module, which can be exchanged when scenarios differ significantly (and the difference cannot be captured within the parameters of one model).

The simulation emulates situation in the Indian Ocean, focusing on the phenomenon of maritime piracy as of 2011 (all data sources were gathered for this year). We have calibrated the simulation on these data-sets, the simulation can be re-calibrated for differing datasets if required.

The game-theoretic setting, i.e., merchant vessel randomized route planner for a predefined area (or set of routes) and pirate strategic behavior, is implemented in form of different modules of merchant traffic and piracy models. The simulation scope is limited—we do not employ the naval assets model and we let only one or a few vessels to interact.

To construct the agent-based simulation, we have followed the simulation design guidelines described by Klügl (2009) (see Section 2.4 for details).

Note on Naming

In the military domain, merchant vessels are denoted as MV, pirate vessels—recently mainly operating as a group of one mothership and multiple accompanying speedboats—are denoted as pirate attack group (PAG) and navy vessels as coalition forces (CF). We denote the vessel classes as M, P and N respectively. Furthermore, although we use the term vessel and vessel agent rather interchangeably, we use the latter if we want to stress the behavioral aspect of vessel

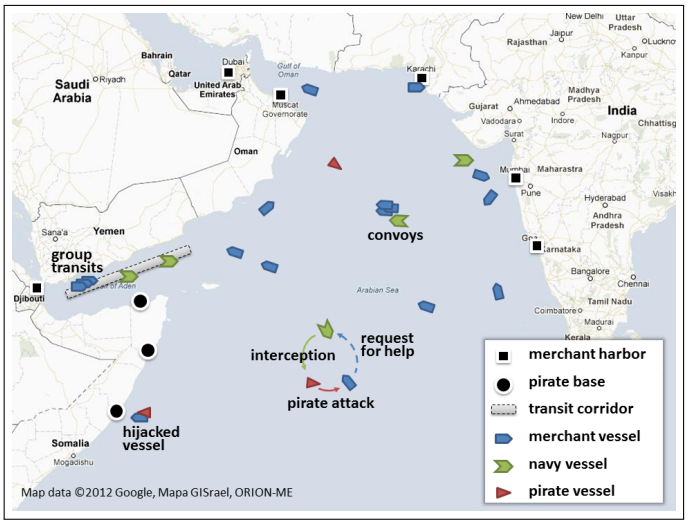


Fig. 5.1: Key actors, activities and environmental features represented in the AGENTC model of piracy affected waters.

description. With some simplification, vessel agent can be viewed as the shipmaster controlling a respective vessel.

5.2 Domain Model

The AGENTC model represents the movement and other activities of selected categories of vessels in piracy-affected waters of the Indian Ocean (we focus on piracy with origins in Somalia, affecting the Gulf of Aden, the Arabian Sea and the West Indian Ocean). A visual overview of the model and its constituent entities is given in Figure 5.1.

5.2.1 Maritime Environment Model

The environment model represents the physical maritime environment in which the vessels operate. It consists of two principal components:

- *geography, bathymetry*—represent the geography of the maritime environment in terms of a set of polygons representing land masses, shallow waters and other obstructions that limit navigability. It also contains locations of ports and anchorages used in merchant shipping and pirate activities.
- *weather*—represents the environmental conditions affecting the behavior of modeled vessels, specifically, wave height, wind speed and currents. Wave height plays an important role in pirate's decision making; currents and wind slightly alter routes of small vessels (e.g., pirate boats).

5.2.2 Merchant Shipping Model

Merchant vessels are large ocean-going vessels carrying cargo over long distances between world's major ports. Merchant vessels are the primary targets of pirate attacks. In order to be useful for what-if analysis of different counter-piracy measures, strategies and policies, the merchant traffic model has to produce realistic traffic patterns even for the situations which diverge from the current status quo in terms of pirate operations and the configurations of piracy countermeasures deployed. The merchant traffic model therefore cannot solely mimic current real-world merchant vessel routes but it has to be capable of generating realistic shipping traffic from more fundamental principles. We therefore adopt the approach of separating the modeling of transportation demand from traffic routing. Demand is considered given and fixed while the routes are generated dynamically, based on the assumption that merchant vessel agents maximize their utility and take the most advantageous route possible. Such an approach is widely used in ground transportation modeling, its application to global maritime shipping, however, is novel.

5.2.2.1 Merchant Shipping Origin–Destination Matrix

The demand for merchant transportation is represented in terms of an *origin-destination matrix (O-D matrix)* which specifies the volume of merchant traffic between world's major ports. The O-D matrix is used to generate origins and destinations for individual merchant vessel voyages and the voyage planning module is then used to find optimum routes connecting voyage endpoints.

Unfortunately, in contrast to ground traffic modeling, no data explicitly and completely capturing the merchant shipping O-D matrix is available; we were therefore forced to estimate the matrix from several partial sources. We have extracted the most important ports in and near the observed area from CI-online database and Ports&Ships[1] portal. We then estimated the O-D matrix by fitting generated traffic to known real-world traffic densities (see Section 5.3.3).

5.2.2.2 Voyage Planning

A fundamental part of the merchant vessel operation is voyage planning. Voyage planning is primarily used in the model of merchant vessels to generate realistic merchant traffic from the merchant shipping O-D matrix; however, it is also used in the operation of the other two vessel classes.

Voyage planning is modeled as an optimization problem of finding an optimal route between two points on a sphere, given vessel-specific route optimality criterion, a set of constraints imposed by geographical obstacles and physical properties of the vessel, and a spatial piracy risk function (the latter only when voyage planning is performed for merchant vessels). Geographical obstacles are represented by spherical polygons, each defined by a list of points. The route optimality criterion function $\mathcal{L}$ is expressed as a weighted sum of the route length and the aggregate risk along the route (computed as the sum of risk values on n route segments):

[1] `http://www.ci-online.co.uk`, `http://ports.co.za/`

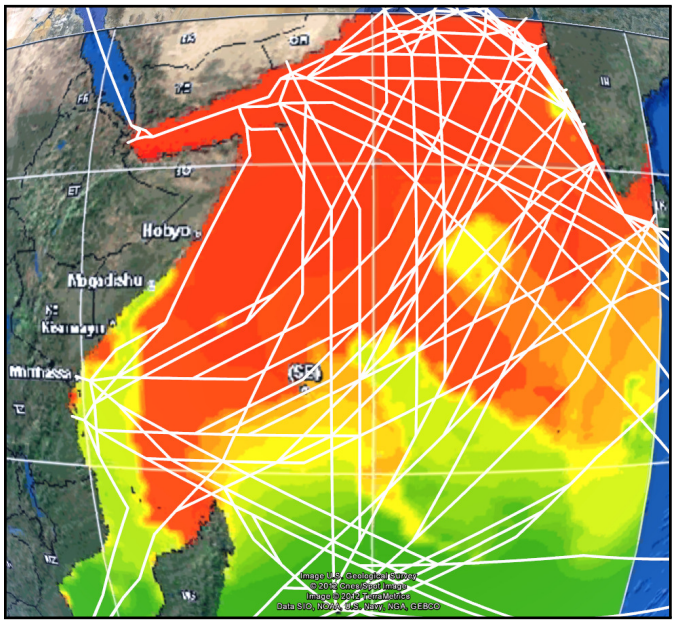


Fig. 5.2: PARS model for October 1st, 2011, provided by NATO Shipping centre—green, yellow, red colors correspond to low, medium and high risk of attack respectively. Plans are generated for each O-D matrix entry with risk aversion coefficient $\alpha = 0.5$.

$$\mathcal{L} = \sum_{i=0}^{n} \left((1-\alpha)\, l_i + \alpha r_i l_i \right) \tag{5.1}$$

where l_i is length of the i-th segment in nautical miles and r_i is risk value on the i-th segment, having values from interval $[0, 1]$. The weight in this sum is the *risk aversion* coefficient $\alpha \in [0, 1]$, capturing how a respective merchant vessel agent weights risk against the length of the route[2]. The risk aversion coefficient can be set individually for each merchant vessel agent based, e.g., on the level of on-board security, vessel cruising speed or the value of its cargo.

We formalize the problem as path-finding on a graph: we construct a *spherical visibility graph* from the spherical polygons by adding an edge between any two polygon vertices connectible by a geodesic[3] which does not intersect any polygon. The spatial risk function is represented by a discrete cell-grid called *piracy risk map*, assigning a piracy risk value in the interval $[0, 1]$ to each cell according to piracy threat-level maps provided by, e.g., the NATO Shipping Centre[4], synthetically generated, or provided by the user.

An optimal vessel route is then computed using the A* algorithm (Russell et al., 2010) with orthodromic distance[5] heuristics and with the cost function equal to the merchant vessel criterion function (5.1). For illustration, vessel routes generated by the route planner between all pairs

Table 5.1: Merchant vessel parameters

Parameter	Values	Description
Origin	port id	Origin port of vessel voyage
Destination	port id	Destination port of vessel voyage
Docking time	[0, 3] days	Docking time of a merchant vessel
Cruising speed	[10, 20] kn	Vessel travel speed unless participating in a group transit
Ship size	[30, 250] m	Size of the ship
Alertness	[0, 60] hr^{-1}	Frequency of checking for an approaching pirate

of ports considered in the AgentC model are shown in Figure 5.2. Risk aversion coefficient $\alpha = 0.5$ and a risk model obtained from the NATO Shipping Centre were used.

5.2.2.3 Merchant Vessel Population Model

The merchant vessel population model instantiates a population of merchant ships of size $\#M$ with a realistic distribution of key vessel attributes, i.e., speed and size. It then samples the merchant shipping O-D matrix in order to assign each vessel a port of origin and a destination port to reach. All merchant vessel attributes specified by the population model are listed in Table 5.1. 2000 merchant vessels with speed and size distribution taken from a data-set of 2700 real-world vessel samples[6] were used in the simulation to replicate real-world shipping density in the Indian Ocean in Year 2011.

5.2.2.4 Merchant Vessel Agent Behavior Model

The behavior of a simulated merchant vessel is straightforward. Given a pair of origin–destination ports at the beginning of the simulation, the merchant vessel agent invokes the route planner to plan vessel's voyage, taking into account corridors and group transit schemes along its route. The merchant vessel then sets on cruising along the route. After the destination port is reached, a new port is sampled from the O-D matrix and a new route is planned. This basic behavior is interrupted if the vessel is attacked by a pirate vessel, in which case the merchant vessel agent reports attack to nearby merchant vessel agents, notifies the closest navy vessel agent and employs self-defense measures modeled by alertness and awareness parameters (see Section 5.2.5). Activities and transitions of the merchant vessel agent behavior are depicted in Figure 5.3.

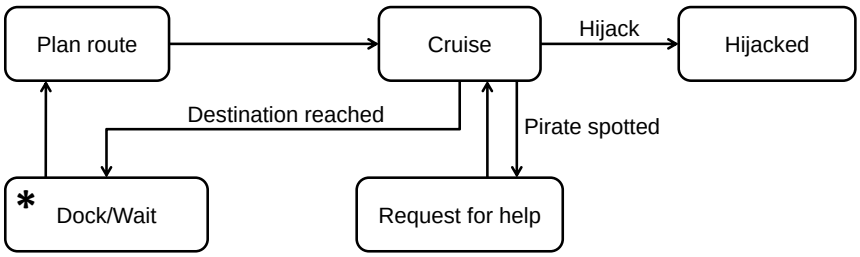


Fig. 5.3: Merchant vessel agent behavior model. The entry point is the *Dock/Wait* state. After docking in a port, the merchant vessel plans a route and cruises to its destination. If a pirate is spotted, a request for help is sent. In case the vessel is hijacked, the hijacked state is terminal and the merchant vessels is under the control of the pirate.

5.2.3 Navy Operations Model

Navy vessels represent military vessels operating in piracy-affected waters and capable of using force to deter and disrupt pirate activities. AgentC focuses on modeling the part of Navy vessel operation consisting in providing assistance to vessels subject to pirate attacks. It does not model active search and area patrolling operations, although such extensions can easily be incorporated in the model assuming data about such operations (which are typically classified) are obtained.

5.2.3.1 Navy Vessel Population Model

Navy vessel population model instantiates $\#N$ navy vessel agents with specified deployment locations. The deployment locations can be specified manually by a human expert or obtained as a result of an optimization process (see below). All navy vessel parameters are listed in Table 5.2.

5.2.3.2 Navy Vessel Behavior Model

The basic behavior of a navy vessel comprises staying in its deployment location waiting for possible distress calls from nearby merchant vessels threatened by pirates. If a distress call is received, the navy vessel agent responds by dispatching a helicopter (if available) and by moving at its cruising speed to the attacked merchant vessel, trying to intercept the attack. Once the response has been completed, the navy vessel returns to its original deployment position. More

[2] The α coefficient is scalarization constant for bi-objective optimization function as in case of the grouping problem in Section 4.2.

[3] The shortest path between two points on a surface of a sphere.

[4] NATO Shipping Centre website: `http://www.shipping.nato.int`

[5] The shortest distance between any two points on the surface of a sphere.

[6] The data-set is a subset of the Vesseltracker (`http://www.vesseltracker.com`) database.

Table 5.2: Navy vessel parameters

Parameter	Values	Description
Helicopter	Y/N	Presence of helicopter on board the navy vessel
Patrolling location	GPS Coords.	Area at which the navy vessel is located and from where it can respond to nearby pirate attack
Action radius	[100, 200] nm	Distance on which the navy vessel reacts to distress calls
Response speed	[20, 30] kn	Speed at which the vessel sails to intercept pirate attack
Helicopter Speed	[140, 170] kn	Speed of the on-board helicopter

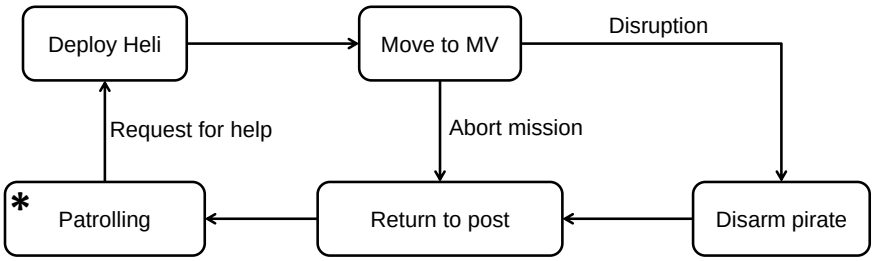


Fig. 5.4: Navy vessel agent behavior model. The entry point is the *Patrolling* state. The navy vessel reacts on a request for help by deploying a helicopter (if available) and cruises towards the merchant vessel. If the navy vessel or the helicopter arrives before the merchant vessel is hijacked, the pirate is disarmed and the vessel (and helicopter) returns to its assigned location. The intervention terminates unsuccessfully if the pirate successfully completes the hijack of the merchant vessel.

details about navy vessel involvement in pirate attacks are given in Section 5.2.5. The behavior model of the navy vessel agent is depicted in Figure 5.4.

5.2.3.3 Navy Vessel Location Assignment Model

Similarly to the merchant traffic model, the navy operations model cannot solely replicate existing real-world deployment locations, but it needs to be able to take into account hypothetical merchant traffic flows in diverse evaluated what-if scenarios. We have therefore developed a navy vessel location assignment algorithm which takes the number of vessels and the density of merchant traffic as input and produces navy vessel deployment locations. The choice of locations aims to maximize the proportion of merchant traffic that lies within the response radius of deployed navy vessels. Technically, the problem can be reduced to a standard set coverage problem which, which is NP hard, however, greedy algorithms are able to achieve the best approximation by a polynomial algorithm (Lund and Yannakakis, 1994). We use an iterative

Table 5.3: Pirate vessel parameters

Parameter	Values	Description
Home anchorage	base id	Base from which the pirate vessel embarks and to which it returns
Cruising speed	[8, 14] kn	Normal speed when traveling long distance between the base and a target location
Pursuit speed	[25,30] kn	Speed during the attack on a merchant vessel
Endurance	[7, 21] days	The number of days the pirate vessel can stay at sea
Visibility radius	[5, 12] nm	Maximum distance of a merchant vessel which the pirate can spot
Attack time	30 min	Duration of attack attempt
Cool-down time	[1, 4] hr	Time needed for recovery after an unsuccessful attack
Navy knowledge	[0, 1]	Probability of knowledge about navy vessel position
Hijack prob. ρ_u	[0, 1]	Probability of successful hijack of a merchant vessel cruising at 10 nm unaware of the pirate attack
Hijack prob. ρ_a	[0, 1]	Probability of successful hijack of a merchant vessel cruising at 10 nm aware of the pirate attack

greedy algorithm to determine navy vessel locations—in each iteration, a new vessel is placed at a position that maximizes the amount of merchant traffic covered within the action radius of all already allocated navy vessels, until all navy vessels are placed.

5.2.4 Pirate Activity Model

Pirate vessels range from small skiffs up to large motherships acting as a floating base from which speedboats are launched to attack. We model piracy at the level of individual pirate attack groups which are represented by a single pirate vessel agent having its home anchorage and operating in and around main shipping lanes, where it attempts to attack, board and hijack passing merchant vessels.

5.2.4.1 Pirate Population Model

The pirate population model is currently simple and is only used to generate $\#P$ pirate agents and to assign them their *home-anchorage* parameter. The assignment is based on reported estimates of the number of pirate attack groups operating from each known pirate anchorage. All parameters of the pirate are listed in Table 5.3.

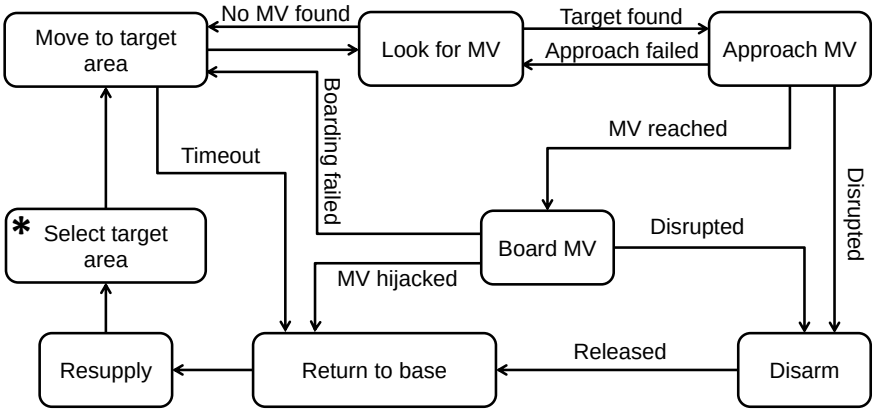


Fig. 5.5: Pirate vessel agent behavior model. The entry point is the *Select target area* state. After the pirate reaches the target area previously selected, it looks for a suitable merchant vessel to approach and hijack. The hijack attempt can be unsuccessful, interrupted by a navy vessel—in which case the pirate is disarmed and sails home—or successful, in which case he sails with the merchant vessel to its home base.

5.2.4.2 Pirate Vessel Agent Behavior Model

The pirate vessel agent behavioral cycle consists of three stages:

1. *Cruising*—the pirate vessel moves directly to its selected target area and looks for a suitable merchant vessel to attack. If the pirate vessel agent spots a navy vessel, it steers away temporarily. When the pirate vessel reaches the target area, it moves at a low speed and changes its course randomly from time to time.
2. *Attack*—If a suitable merchant vessel is spotted, the pursuit starts (described in detail in Section 5.2.5).
3. *Recuperation*—After a successful attack or when running out of supplies (*endurance* parameter), the pirate agent navigates back to its home anchorage. After an unsuccessful attack, the pirate recovers (*cool-down time* parameter) and looks for a merchant vessel again.

Activity diagram of pirate vessel agent behavior is depicted in Figure 5.5. Note that we do not model the economic aspect of piracy, such as ransom negotiation and other processes taking places after a hijacked vessel is brought to shore.

5.2.4.3 Target Attack Area Selection Mechanism

A key part of the pirate vessel decision-making is choosing its target area where it will look for a merchant vessel to attack. Again, in order to have the pirate activity reflect the simulated scenario, the target area cannot be predefined based on attack locations currently observed in the real world but needs to be determined dynamically from more fundamental principles. AgentC therefore implements a mechanism that determines attack locations dynamically from the assumption that the pirates aim to maximize their utility and choose such locations which

maximize their chance of a successful hijack (with some exploration). Pirate's target area is selected based on the following inputs: weather conditions, merchant traffic density (depicted in Figure 5.7a) and (partial) knowledge about navy vessel positions.

In the first phase, a subset of regions with acceptable weather conditions[7] is selected. In the second phase, navy vessels positions—if known—are combined with merchant traffic density map and an area is randomly selected with a probability proportional to its expected reward.

5.2.5 Pirate Attack Model

Pirate attack is a complex interaction between all three classes of vessels; we therefore provide its standalone description that complements the description of the attack from the perspective of individual vessel agent behaviors. Parameters directly influencing the course and the outcome of the attack are depicted in Table 5.4 and are a subset of parameters of individual vessel classes, except the M *awareness* binary parameter which is well-defined only during the attack phase. All interactions taking place during a pirate attack are depicted in Figure 5.6. The attack consists of three phases:

1. *pre-attack/approach*—this phase begins after a merchant vessel is spotted by a pirate vessel agent; the pirate vessel agent starts a pursuit at its pursuit speed. The merchant vessel agent checks for an approaching pirate vessel several times per hour (parameterized by the alertness parameter capturing the probability of spotting an approaching pirate). If a pirate vessel is spotted during its approach, the awareness parameter is set to true, meaning that the merchant vessel is not taken by surprise by the attacking pirate and can deploy self-defensive countermeasures. Furthermore, upon spotting the attack, the attacked merchant vessel agent broadcasts a distress call and notifies nearby merchant vessel agents about the danger (their awareness is then set to true). If there is an idle navy vessel within the navy vessel action radius, the navy vessel responds by moving towards the attack. If the navy vessel carries an on-board helicopter, it dispatches the helicopter to prevent the pirate from hijacking the merchant vessel.
2. *attack*—the pirate vessel agent attempts (repeatedly, for a time period, specified by the *attack-time* parameter) to board the merchant vessel and seize control. The probability of success depends on the speed s of the merchant vessel, its alertness a and subsequently on its awareness. The average probability of a merchant vessel being hijacked without any navy vessels present in the model can be computed as $p_h = a \cdot p_a(s) + (1 - a) \cdot p_u(s)$, where $p_a(s)$ and $p_u(s)$ are the probabilities of hijacking an aware and unaware merchant vessel, respectively, traveling at speed s. The hijacking probabilities are linear functions (with threshold) of merchant vessel speed: $p_a(s) = max\{0, (2 - s/\nu)\rho_a\}$, $p_u(s) = \max\{0, (2 - s/\nu)\rho_u\}$, where ρ_a and ρ_u are base probabilities specifying the probability of hijacking a merchant vessel cruising at $\nu \geq 10$ kn (minimum cruising speed of a merchant vessels in our model is 10 kn).

[7] Based on the correlation of attack frequencies and weather conditions in 2011, we have estimated the acceptable wave height for piracy operations to be up to 1.25 m.

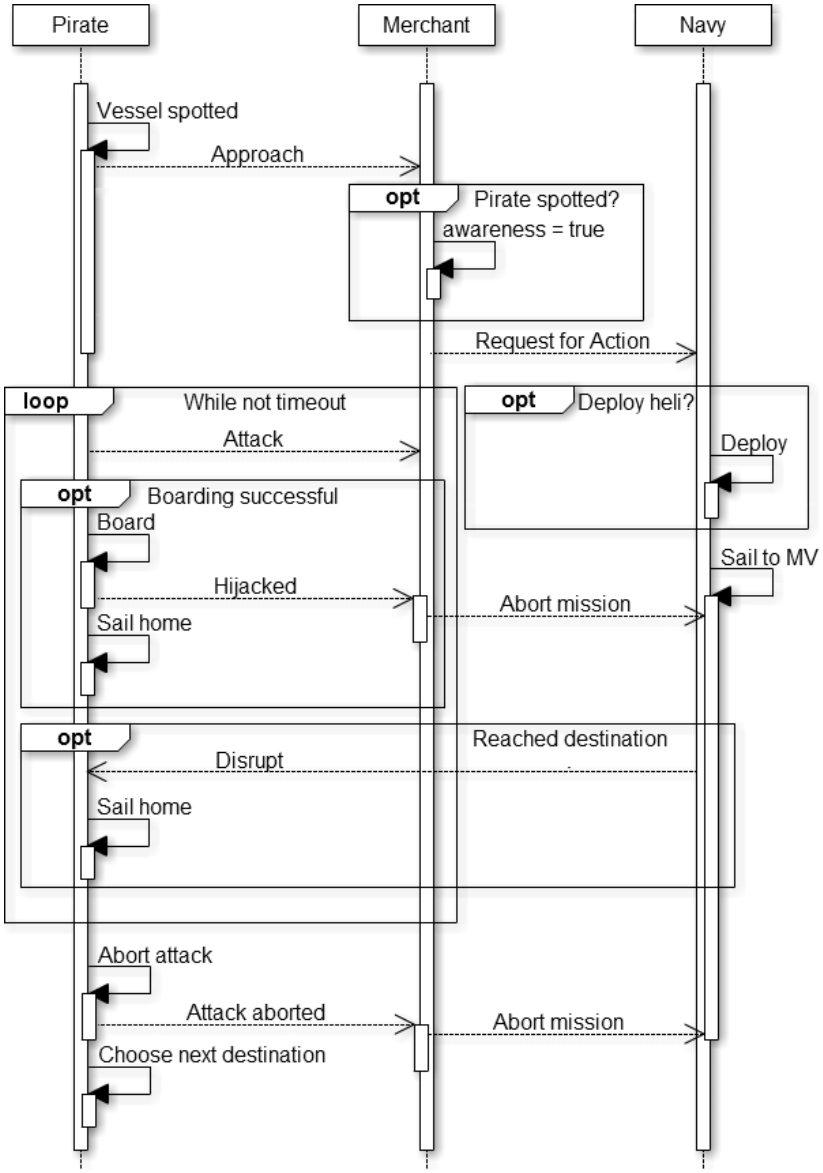


Fig. 5.6: Sequence diagram of the pirate–merchant–navy vessel interaction during a pirate attack. The attack takes a predefined amount of time and is terminated either by the successful hijack of the merchant vessel or by the arrival of a navy vessel or its helicopter.

3. *post-attack*—if the attack is successful, the hijacked vessel is taken to the pirate's home anchorage; if the attack is aborted by the pirate (after attacking unsuccessfully for a period given by pirate's *attack time* parameter), the merchant vessel continues its voyage according to the original plan; the pirate vessel recovers from the attack for a specified period of time (*cool-down time* parameter) and then it looks for another target to attack. If the attack is interrupted by the navy, the pirate is disarmed and sails to its home anchorage without further trying to attack any merchant vessels.

Table 5.4: Parameters affecting the outcome of a pirate attack.

Parameter	Values	Description
M Cruising speed	[10, 20] kn	Cruising speed of the merchant vessel
M Alertness	[0, 60] hr^{-1}	How often the merchant vessel checks for pirate presence in its vicinity
M Awareness	Y/N	Merchant vessel's knowledge about an approaching pirate (element of surprise)
P Visibility radius	[5, 12] nm	Maximum distance at which a merchant vessel can be spotted
P Pursuit speed	[25,30] kn	Cruising speed of the pirate
P Attack time	30 min	Maximum time for which the pirate attacks a merchant vessel
P Hijack prob. ρ_u, ρ_a	[0, 1]	Probability of hijack of (un)aware merchant vessel
N Helicopter	Y/N	Presence of helicopter on board the navy vessel
N Action radius	[100, 200] nm	Navy vessel distress call response radius
N Helicopter speed	[140, 170] kn	Speed of navy vessel's on-board helicopter
N Cruising speed	[20, 30] nm	Speed of a navy vessel

5.2.6 Piracy Countermeasures Model

Merchant and navy vessels can engage in various piracy countermeasures designed to increase the security of passage through piracy-affected waters. Most of such measures require the cooperation between multiple vessels and can be viewed as multi-agent coordination mechanisms that augment standard, single-agent vessel behaviors. Based on discussions with the maritime security community, we support the following operational piracy countermeasures in the AgentC model:

- *Recommended transit corridors*, which concentrate merchant traffic along a defined route connecting a sequence of waypoints. Such concentration of traffic facilitates protection from navy vessels; however, it also makes targeting transiting vessels easier for pirates. The corridors are modeled as extensions of the merchant voyage planner.
- *Group transit schemes*, which coordinate the timing of merchant vessel transit so that vessels pass high-risk piracy areas in groups. This improves mutual awareness and facilitates navy response; however, it makes the transit take longer as vessels have to follow a predefined schedule and may have to reduce their cruising speeds to match the speed of their respective transit group. The group transit schemes are modeled as an extension of the voyage planner; they assign time marks to a subset of waypoints in a plan and the merchant vessels is then required to be at those waypoints at given time.

Table 5.5: Piracy countermeasures considered, with sets of parameters by which they are specified.

Countermeasure	**Parameters**
Transit corridor	Sequence of GPS waypoints
Navy vessel deployment	Set of locations
Group transit scheme	corridor, Speed levels, transit schedule (per speed level)
On-board security team	Alertness of merchant vessels

- *Navy vessel deployments*, which deploy navy vessels in strategic locations from where they can provide assistance to nearby merchant vessels in case of a pirate attack. We consider only stationary deployments (see Section 5.2.3).
- *On-board security teams* consists in deploying armed security personnel on-board of vessels transiting high-risk areas capable of deterring attackers and denying them access to the vessel. This countermeasure is currently modeled by the alertness and awareness parameters of the merchant vessel.

Each countermeasure is parameterized by a set of parameters (see Table 5.5). Except for route randomization, all above measures are currently actively used, although convoy schemes are operated rather sporadically by national navies on an ad-hoc basis. The usage of transit corridors and group transits is currently limited to the Gulf of Aden.

5.2.7 Simulation Model Outputs

By its very nature, the agent-based micro-simulation model allows recording and evaluating, at different levels, the multitude of information about the behavior of the modeled maritime transportation system.

At the lowest level, each simulation run produces a detailed log of all events generated by vessel agents and their interactions among themselves and with the modeled maritime environment. One year of simulated maritime traffic generates hundreds of thousands of events recording vessel locations in time, state-transition events (e.g., *destination-reached* event, *target-area-selected* event etc.) and events generated during vessel interactions (e.g., *pirate-spotted* event, *pirate-attack* event, *pirate-attack-disrupted* event and many others). Events logs can be used for studying micro-level behaviors involving one or more individual vessels. They can be also used for an on-line visualization of individual simulation runs, which is crucial for the presentation and face validation purposes.

Micro-level event logs are aggregated in time and/or space into meso-level spatio-temporal output reports describing the occurrence of specific event or a set of events over time or in geographical space. A particularly important type of the output report at this level are *density maps* which capture the frequency of occurrence of a specific event in a grid-discretized space (as

an example, see Figure 5.8a for a density map of pirate attacks). Meso-level reports are useful for understanding how a certain phenomena is geographically distributed, how it is changing in time or both.

Finally, at the highest-level, event logs and spatio-temporal reports are aggregated into simple numerical statistics summarizing the activity in the piracy-affected waters. Both security-related and operational output quantities are evaluated. The former includes *pirate attack count*, i.e, the number of all attacks over a given time period (e.g., a year), and *attack success ratio*, i.e, the number of successful attacks (i.e., hijacks) divided by the number of all attacks; the latter includes *average transit distance* and *average transit duration*, which can be used to estimate operational shipping costs. The high-level statistics are used to gain the insight into the global behavior of the modeled system under different circumstances and are also crucial for model calibration.

5.3 Model Calibration

The AgentC model contains a wide range of parameters that needs to be specified. Most of these parameters were set based on the consultation of domain sources and experts. There are, however, also parameters which significantly affect the behavior of the model and for which no reliable sources exist—the values of these parameters were therefore determined through the calibration against the real-word data.

In the following sections, we describe the calibration and validation process employed. Specifically, we describe the calibration methodology selected, the sensitivity analysis, the calibration of the merchant traffic sub-model, and the calibration and the validation of the complete model.

5.3.1 Calibration Methodology

The purpose of the calibration step is to set the values of key model parameters so that the behavior of the model most closely reflects the behavior of the real system; this closeness of the model is measured in terms of several fitness criteria. Due to limited supply of computing resources, we performed greedy iterative calibration—in each iteration, we chose a subset of parameters most influencing the fitness criteria, found the optimal value of these parameters and fixed them in subsequent iterations. To further speed up the calibration, we used different calibration fitness metric in each step. The ordering of steps and the choice of calibrated variables and fitness metrics in each step was based on the results of the sensitivity analysis. Depending on the standard deviation of the selected fitness criterion, we executed between 50 and 100 simulation runs (with a different random generator seed) for each model configuration (i.e., each combination of model parameters – see Table 5.6).

In order to compare the spatial outputs of the model, we used spatial *success rate (SR) curves* as described by Chung and Fabbri (2003). Spatial success rate curves give a concise account of the performance of a spatial model. Specifically, the curve specifies what percentage of space a

given spatial model needs in order to cover a given percentage of real-world occurrences of the event of interest (e.g., pirate attacks). The smaller the area required to cover a given percentage of the events, the tighter the fit and the better the model. The SR curve is constructed by discretizing the modeled area into a finite-size cells and then sorting the cells according to the relative frequency of the event occurring in the cell (e.g., relative frequency of pirate attacks). The SR percentage value for x percent of covering cells is determined by taking the x percent of the highest-frequency cells and counting the percentage of events occurring within these cells. SR curves can be modified in order to be used for comparing spatial models against spatial event density maps rather than sets of discrete events. In this case, coverage is counted as the fraction of the overall mass[8] of the density map covered by a given proportion of highest-ranking cells. We further define the *SR curve index* as the percentage of the area under the SR curve with respect to the overall rectangular plot area (i.e., the area below and above the curve). The interpretation of SR curves is illustrated in the captions of Figures 5.7 and 5.8. To our knowledge, this is the first use of SR curves for the calibration of simulation models.

5.3.2 Sensitivity Analysis

Before the actual calibration, we performed the sensitivity analysis in order to understand how the variation of key model parameters affects model outputs: (1) attack distribution, (2) attack frequency and (3) attack success ratio. Table 5.6 summarizes the sensitivity of each output to the variation of the given model parameter, measured in terms of the coefficient of variation[9] of a given output variable when varying a given model parameter. For each of the model outputs, we have selected the most sensitive parameters which were then varied while the rest of the parameters remained fixed.

Table 5.6: Coefficients of variation for each criteria and model parameter. The bold-faced values correspond to parameters which were varied when calibrating the model for each criterion

Parameter	**Attack Dist.**	**Attack Freq.**	**Hijack Ratio**
#N	**0.15**	0.24	0.32
#P	0.046	**0.74**	0.041
P Visibility radius	0.052	**0.26**	0.11
M Alertness	0.053	0.075	**0.20**
P Hijack prob. ρ_a, ρ_u	0.057	0.078	**0.16**
P Navy knowledge	**0.1**	0.085	0.14

[8] Where the mass of a density map is the sum of density values in all cells. Note that we assume the space is discretized into finite-sized cells.

[9] Coefficient of variation is defined as a ratio of standard deviation and mean: $c_v = \frac{\sigma}{\mu}$.

Algorithm 6 OD matrix Calibration

```
1: max, step
2: SRindex ← ∞
3: AMVER ← AMVER density map
4: ODmatrix ← U(ones)
5: best ← 0
6: changed ← true
7: while changed do
8:    changed ← false
9:    for entry ∈ U(ODmatrix) do
10:      while entry < max do
11:         entry ← entry + step
12:         SRindex ← (AMVER, ODmatrix)
13:         if SRindex < best then
14:            entry ← entry − step
15:            break
16:         else
17:            best ← SRindex
18:            changed ← true
19:         end if
20:      end while
21:   end for
22: end while
```

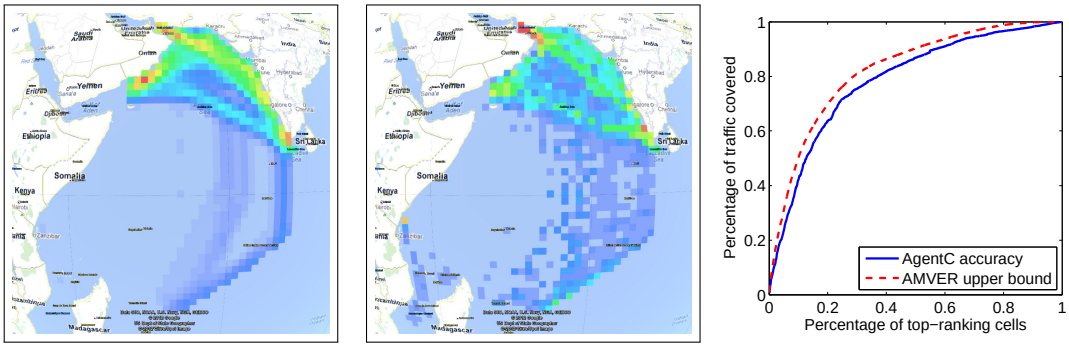
5.3.3 Merchant Traffic Sub-Model Calibration

In the first calibration step, we calibrated the merchant traffic sub-model of the AgentC model, i.e., the model consisting solely of merchant vessels. The calibration involved estimating all entries in the O-D matrix (see Section 5.2.2.1) together with the risk aversion parameter $\alpha \in [0, 1]$ (see Section 5.2.2.2). We used a canonical average risk map modeling the piracy risk as a function of the distance from main pirate anchorages for risk-aware routing. We used the SR curve index between the simulated merchant traffic density and reference 2011 traffic density map provided by AMVER[10] (see Figure 5.7b) as the calibration fitness metric in this step.

The calibration of the O-D matrix required setting the number of voyages between all pairs of 20 major world ports, i.e., altogether almost 400 entries so that the generated merchant traffic matches well the reference AMVER traffic density map. Such a number of parameters coupled with non-linearity of the fitness metrics (the SR index) made the optimum solution intractable; we therefore employed a greedy approach with random restarts to determine locally optimum values for each origin-destination pair.

The algorithm is depicted on Algorithm 6. The O-D matrix is initialized with ones in entries over the diagonal. Then, for each entry in the upper triangular matrix, the algorithm tries to increase its value as long as the SR index is increasing, until no entry can be increased. Then, the procedure is repeated for the created OD matrix by subtracting *step* from every entry if it increases the SR index. The randomization part is in the ordering of the entries, which are accessed on the 9-th line of the algorithm. The O-D matrix calibration was repeated 100 times

[10] Automated Mutual-Assistance Vessel Rescue System (AMVER), `http://www.amver.com`.



(a) (b) (c)

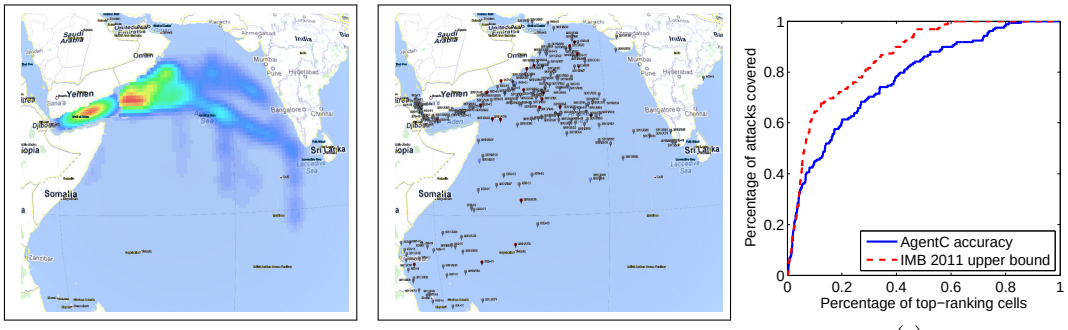
Fig. 5.7: Merchant traffic sub-model calibration. (a) Density map for merchant traffic sub-model. (b) Reference AMVER 2011 traffic density map. (c) SR curves for the merchant traffic sub-model (blue solid line) and the AMVER density map (red dashed line). The red SR curve of the AMVER model captures the theoretical upper-bound achievable for a given spatial resolution of the model: 20% of the AMVER top ranking cells cover 70% of the AMVER traffic; 20% of the AgentC merchant traffic sub-model top ranking cells cover approximately 64% of the AMVER traffic.

for each risk aversion coefficient $\alpha = \{0, 0.1, \ldots, 1\}$. The best fit was achieved for $\alpha^* = 0.6$; the resulting traffic density map for α^* and the associated SR curve are given in Figures 5.7a and 5.7c, respectively. Minor discrepancies can be observed around Kenyan and Tanzanian coast and in the Mozambique channel; overall, the fit is very good.

5.3.4 Complete Model Calibration

Following the calibration of the merchant traffic sub-model, we calibrated the complete model containing all three categories of vessels. The complete model was calibrated to fit the situation in the Indian Ocean in 2011, where there were 181 attacks (source: IMB 2011 reports), from which 28 were hijacks (15.4% hijack success rate). Even though some of the attacks are unreported and thus the IMB 2011 reports are incomplete, it is to our best knowledge the most comprehensive report source. The calibration consisted of the following steps:

In the calibration process, we need to compare the output of the model to the reference real-world data to assess the accuracy of the model. We thus constructed 2D regular rectangular grid G defined over the observed area with a predefined cell size. A grid serves as an underlying discretization structure to create density maps $G(E)$: 2D histograms, storing a value in each cell and thus capturing spatial distribution of various phenomena E (e.g., hijacks, traffic density) which are described by a list of spatial events $E = e_1(lat_1, lon_1), ..., e_n(lat_n, lon_n)$ (created from data sets or from the simulation output). Each event $e_i(lat_i, lon_i)$ is mapped onto the grid G by either increasing the value of a cell c which contains the point (lat_i, lon_i) (direct mapping) or mapped also to a set of cells surrounding c (smoothed mapping, taking e.g., into account the spatial uncertainty of the event).



(a) (b) (c)

Fig. 5.8: Complete model calibration—attack spatial distribution fitting. (a) Density map for the complete model. (b) IMB 2011 Reports. (c) SR curves for the complete model (blue solid line) and IMB 2011 reports (red dashed line). The red SR curve for IMB 2011 reports was measured by transforming the incidents into a density map on which the SR curve was measured. This IMB density map thus serves as a theoretical upper-bound 20% of the densest cells in the IMB density map covers approximately 72% and 20% of the AgentC model density map covers 61%.

A density map $G(E)$ can be compared with another density map $G(\bar{E})$ using a number of metrics for comparing two histograms or images, e.g., Kullback-Leibler divergence $KL(G(E), G(\bar{E}))$ (Kullback and Leibler, 1951) or Symmetric Image Ratio $SIR(G(E), G(\bar{E}))$ (Yang and Mueller, 2008). We use the second metric which can account for zero values of cells. The SIR value is in the interval [0,1] where higher values correspond to more similar density maps.

Additionally, a similarity of a density map $G(E)$ with an event set $\bar{E}$ (which is, e.g., too sparse to create an accurate density map) can be quantified by integrating the area under a *prediction-rate curve* (Chung and Fabbri, 2003).

5.3.4.1 Attack Spatial Distribution Fitting

First, the complete model was calibrated with regards to the spatial distribution of pirate attacks. The number of navy vessels $\#N$ $(0 - 500)$ and pirate's *P-navy-knowledge* were the calibrated variables; the SR curve index (see Section 5.3.1) between the attack density map produced by the model and the IMB 2011 reports was used as the fitness metrics. The best fit was found for $\#N = 50$ and *P-navy-knowledge* $= 0.4$. The attack density map produced by the model and the SR curve for the best values of calibrated parameters are given in Figure 5.8, along with the reference IMB 2011 reports.

Two directly observable discrepancies between the AgentC model and the situation in 2011 can be observed: the attacks in the AgentC model are concentrated in the East Arabian sea, not spreading to the North. Additionally, due to the sparse AgentC merchant traffic in the West Indian Ocean, there are no attacks in that area, however, the AgentC pirates are attacking the traffic under the southern tip of India, which did not happen in 2011.

5.3.4.2 Pirate Attack Frequency Fitting

Second, the complete model was calibrated with regards to the overall number of attacks. The number of pirates $\#P$ (0–5) and pirate's *P-visibility-radius* (5–12 nm) were the calibrated parameters; the fitness metric was a difference between the overall number of attacks produced by the model and the reference real-word value of 181 based on the IMB 2011 reports. The best fit was obtained for $\#P = 2$ and *P-visibility-radius* $= 6$ nm, producing on average 182 attacks with the standard deviation of 16.1.

5.3.4.3 Pirate Attack Success Ratio Fitting

Finally, the complete model was calibrated with regards to the attack success ratio. Two parameters influencing the outcome of the merchant vessel-pirate interaction were calibrated: *M-alertness*$\in [0, 1]$ and *P-base-hijack-probability* $\rho_a, \rho_u \in [0, 1]$ (defined in Section 5.2.5); the fitness metric was the difference to the reference success ratio based on IMB 2011 report statistics, was 0.15. Best fit was obtained for *M-alertness*$= 0.5$, $\rho_a = 0.2$ and $\rho_u = 0.5$, estimating the probability of a hijack with navy vessels ($\#N = 50$—fixed in the previous calibration phase) to be 0.15 and without any navy vessels to be $p = 0.35$. The probability of a pirate being disrupted by a navy vessel is then 43%, according to our model—this is an example of an insight which cannot be directly inferred from the collected attack reports alone.

5.3.5 Validation

Face validation was performed repeatedly throughout the model development process. We consulted experts and officials from the industry, government and military, including International Maritime Organization, U.S. Naval Research Lab, U.S. Naval Postgraduate School and several maritime security providers. The feedback received on structural walkthroughs and visualized simulation runs confirmed structural and behavioral plausibility of the proposed model.

Unfortunately, due to lack of the data on the behavior of the maritime transportation system under varying circumstances (e.g., the exact number of pirate attack groups and/or deployed naval warships), we were unable to statistically validate the model. The model should not therefore be treated as reliable for quantitative prediction and should be used for gaining qualitative insights only.

5.4 Summary

We have developed an agent-based model of the maritime traffic in piracy infested areas in the form of a step-based simulation with three main agent classes taking part in the problem: merchant vessels, pirates and naval patrols. We have integrated various data sources to capture most relevant layers of the environment, such as geographical boundaries, weather conditions and

transit schemes deployed in the area. We decompose the agent model into a population model—parametrized by a set of parameters configuring complete population of one agent class—and an individual behavior model—parametrized by a set of parameters configuring an individual behavior of each agent, together with its decision-making process. This decomposition allows us to implement various behaviors and strategies for each agent, being it an intelligent route planner for merchant vessels or game-theoretic spot selection for pirates.

The scope of the simulation is much wider than a single purpose of the evaluation of interaction of two (or more) agents. The simulation can be calibrated on given dataset (e.g., pirate activity and transit scheme for year 2011) and a sophisticated what-if analysis can be conducted to reveal potential impact of, e.g., different allocation naval patrols, the introduction of additional transit corridors, the modification of existing grouping schemes etc. The what-if simulation module (Vaněk et al., 2012a; Vaněk et al., 2013c) is out of scope of this thesis, however, the calibration is briefly described to provide an overview of main methods, parameters, and metrics used.

The computational model was developed throughout the span of four years, with periodical technical reports for the first three years (Jakob et al., 2009, 2010b, 2011b), additionally, we have published increments on the Autonomous Agents and Multi-agent Systems conference (Jakob et al., 2010a, 2011a, 2012b) and simulation-focused workshops (Vaněk et al., 2012a) periodically. Additionally, first results were published as a journal article (Vaněk, 2010), and periodical overviews of the framework have been reported to the community in imapcted periodicals (Jakob et al., 2011c; Vaněk et al., 2013c).

Chapter 6
Simulation-based Validation

On the evaluation of the robustness of the proposed models using the agent-based simulation of maritime piracy.

A mind is a simulation that simulates itself.

–Erol Ozan

We have stated in Chapter 5 that a necessary step between the mathematical model solution and the real-world deployment is a validation in a richer environment, which is closer to the real world, however, at the same time, which allows us to implement the solutions proposed by the mathematical models. In our specific case, the role of this link will be played by the agent-based simulation, enabling the implementation of the game-theoretic strategies into a rich simulated reality of maritime traffic and the evaluation of robustness and properties of the strategies in a more realistic environment, thus allowing us to estimate suitability for the real-world deployment.

6.1 Comparison of Models' Fidelity

The level of abstraction and parameterization of the model which is used to provide a solution to a defined problem strongly correlates with model's accuracy and robustness. Table 6.1 summarizes the main differences between the real-world problem of maritime piracy—i.e., the phenomenon of merchant vessel hijacks by pirate boats—and its representation using the game-theoretic model or simulation framework. The phenomena are divided into following categories, which are inspired by the agent-based approach to world representation: (1) environment model parameters, (2) agents' physical model, (3) agents' mental model, (4) agents' interaction model.

The space-time continuum of the real-world is represented in the game-theoretic model by a graph with nodes and edges and units steps between the nodes, the simulation formalizes the space as a 2D surface of the sphere and (as it is step-based simulation) the time is discretized into predefined quanta.

The visibility of another player depends in reality on many factors, mainly on the properties of the environment, such as the sea state, the aerial visibility/fogginess and possibly on the detection equipment. And the visibility decreases non-linearly with the distance of both agents. In the game-theoretic framework, the agents have to be at the same edge or node to detect each other; the environmental conditions can be represented by the interception probability parameter. In case of the simulation, the visibility is defined by a radius, where inside of the radius, the visibility is set to 1 and to 0 outside of the radius. The probability of the interception is in

Table 6.1: Parameters of the game-theoretic and simulation model compared to the real-world.

Parameter	**Real-World**	**Game-theoretic Model**	**Simulation**
Space	3D space	graph	globe surface
Time	continuous	discrete unit-length steps	discrete unit-time steps
Visibility	function of distance of vessels/state of environment	edge/node	radius
Interception prob.	result of complex micro-interaction	probability-based	probability-based
Position of base	shore/floating	node in graph	shore/floating
Player's Model	3D object	point in graph	3D box
Player's Mobility	speed+acceleration	unit-step speed	speed
Player's Reasoning	behavioral	rational	rational/adaptive
Player's Knowledge	experience-based	perfect/none	perfect/partial
Action Execution	stochastic execution	deterministic exec.	deterministic exec.

reality a result of complex micro-interaction between both players, given physical constraints of the vessels (berth size, maneuverability etc.) and deterrence/attack abilities of people involved. In the game-theoretic model as well as in the simulation framework, we represent this micro-interaction by an interception probability, which—in case of the simulation—is computed using the sea-state, merchant ship speed and awareness of the merchant ship parameters. The interception probability is closely related to the mobility model of agents, where in reality, the vessels sail at differing speeds and we have to take into account acceleration models as well as the maneuverability of the (especially) larger vessels to properly predict interception probability. In the game-theoretic model, the ships are represented as points, the maneuverability is unrestricted and we consider an equal speed for both agents, transiting one edge per time-step. In case of the simulation framework, the ships are represented by 3D boxes, however, the maneuverability is unrestricted as well; we can, however, consider differing speeds for both agents.

The positioning of the pirate's base is on the shore, possibly far from the main shipping lanes or—in case of pirate's mothership utilization—in the open ocean. In case of the game-theoretic model, the base is part of the graph, i.e., the base is positioned in one of the nodes. In the simulation framework, we can specify the position of a base anywhere (i.e., on the shore or in the ocean, representing a mothership).

An important (if not crucial) distinction of the models lies in the representation of the decision-making process of the agents. In the real-world, we can assume that the merchant ships are rational and use significant computation resources as well as mental effort to design a route which would decrease possible risk of attack (however, given resource constraints, costs of shipping and insurance premiums). On the other hand, the pirates have, based on first-hand

anecdotal evidence (Hutchins, 2013), their reasoning process more or less simplified; additionally, cultural and religious factors take a substantial part in the decision-making process. They posses neither complex computational infrastructure nor the decision-making knowledge to compute an optimal strategy. Additionally, both players are subject to uncertainty about the other player's payoffs, strategy space and reasoning process.

The game-theoretic framework simplifies greatly many of the real-world aspects described above, mostly the space-time representation and the reasoning process of the attacking player. The game-theoretic framework, by its nature, assumes utility-optimizing agents with more or less rational behavior, unlimited computation resources (and time), the decision being made based on the player's utility captured by the game model itself and a perfect knowledge about the game and about the other player reasoning process as well (even though there are some models considering an uncertainty in knowledge or the game being played (Kiekintveld et al., 2011, 2013)).

The simulation framework stands in-between the mathematical abstraction provided by the game-theoretic model and does not aim to take role of a model able to *provide* solutions, its primary aim is to *verify* solutions designed by simpler, more abstract models. The reasoning process of the simulated pirate (when verifying the solution computed for the merchant ship) has to reflect the reasoning process of the real-world pirate to some degree (i.e., the pirates have some mechanism for knowledge representation); however, we can assume worst-case scenarios, such as perfect knowledge of past routes of merchant vessels or faster speed of the pirate for any sea state.

6.2 Experiment Design

In the first phase, we design a test-case that tries to imitate the abstract game-theoretic model as much as possible. Second, we perturb some parameters—such as the relative speed of vessels or visibility range—and violate some of the assumptions made—such as the number of vessels participating in the transit and knowledge representation of the pirate.

We fix some of the parameters of the simulation to remove biases being caused by them: (1) we set the probability of hijack to zero, i.e., every pirate attempt is unsuccessful, which mitigates decreasing number of merchant vessels throughout the simulation execution and we set the interception probability to 1 in the game-theoretic to simplify results interpretation, (2) we fix pirate's reaction after an attack: the pirate sails home without attacking anyone to account for game end and strategy reset; merchant vessels continue their journey normally (and cannot be caught again before arriving to the destination). We set pirate's speed to 15 nm and we set the merchant vessel's speed to 15 nm in scenarios with equal speeds and sample the merchant vessel's speed from a uniform interval $U(10, 20)$ when measuring the impact of relative speed.

We do not directly deal with the synchronization of the agents, i.e., the simulation is not able to provide sufficiently strong sychronization device—in the game-theoretic model, all edges have an equal length and the agents traverse one edge per step, which is the elementary unit of time. In the simulation framework, a simulation step corresponds to a parameterized time quantum, set to 1 minute of simulated time. Even when the agents start at the same time (which

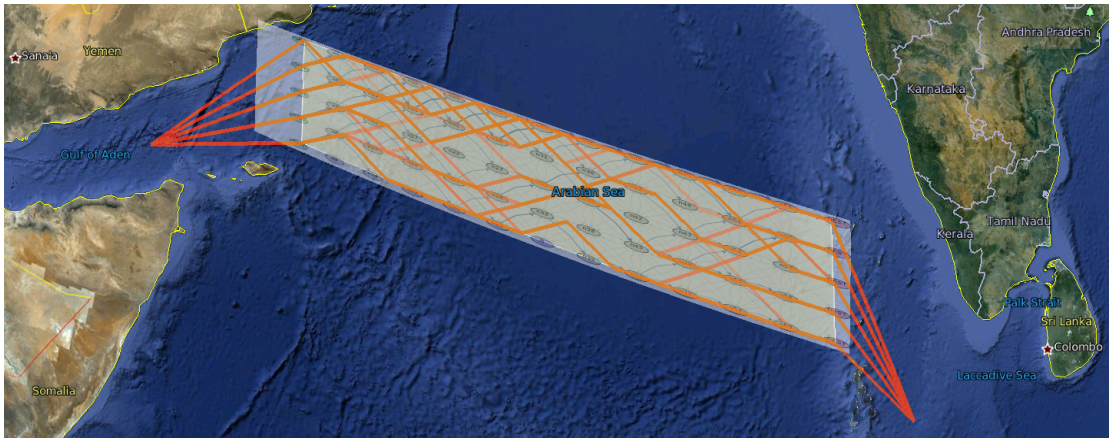


Fig. 6.1: Placement of the game-theoretic model in the simulation.

is not desirable) with the same speed, their movement de-synchronizes after a few steps in sense of graph node visits—the edges do not have equal lengths as we discretize the space using a rectangular grid[1].

For the game-theoretic settings, we use five different game-theoretic graphs with width $w = \{4, 5, 6, 7, 8\}$ with respective walk lengths $maxLength = 4 \cdot w + 1$ which are displayed on Figures B.1 – B.5 in the Appendix B respectively.

We overlay the graph used in the game-theoretic model over the earth surface. The area is defined as a rectangular polygon with 180 nautical miles width, placed over the Indian Ocean spanning from the Gulf of Aden to the Maledives, covering the main shipping lane from south-east Asia to Europe (see Figure 6.1). The vessels follow trajectories sampled from their mixed strategies—they follow lanes through the graph (i.e., the polygon), transiting the Indian Ocean. Origin and Destination nodes are interchangeable, we allow the vessels to sail from both ends of the graph.

In the game-theoretic setting, the game ends after the transport transits the area or is attacked/captured. In the simulation framework, we simulate a certain time frame, in which both the merchant vessel as well as the pirate operate repeatedly; i.e., after the area transit or an attack, both the vessels continue in their strategy execution. We simulate three months of the simulation time and we count the number of transits of all vessels, the number of attacks and the number of pirate's journeys. Parameters varied throughout the evaluation are described in each subsection in the experiment setup.

Expected Biases of the Simulation

We assess some of the expected biases of the simulation prior the simulation execution compared to the real world behavior. These biases have to be taken into account when interpreting results:

[1] Possible solution using hexagonal grids would prohibit the agents to follow straight line, making the traversal of graph—an important ability of the pirate—longer than necessary.

1. simulation-finiteness bias—the simulation runs for a limited amount of time. The simulation can theoretically terminate after the merchant vessel is attacked but does not finish its journey—i.e., the ratio of attacks vs. journeys is not correct.
2. simulation-start bias—at the start of the simulation, the vessels start at their starting points. If the pirate is much slower, compared to the merchant vessel, the hijack probability drops, as the pirate can spend the majority of the simulation around the base.
3. contingency bias—the pirate sails directly to the base after the attack, lowering the frequency of attacks (there is no contingency planning in the game-theoretic model included). This bias rises due to the simulation time-frame, which is different from the game-theoretic time frame.
4. time-discretization bias—the discretization of time gives rise to situations where the merchant vessel's trajectory crosses the circle given by the visibility radius of the pirate within one simulation time step and thus is not noticed by the pirate which leads to (very slight) decrease of attack frequency.

6.3 Statistical Parameters of the Simulation

The simulation framework is stochastic, driven by a set of probability distributions used for the sampling of agent's behavior parameters (such as the waiting time in a dock), for decision-making (e.g., which strategy to select) and for resolving interactions between the agents (e.g., visibility range). This section aims to evaluate the level of stochasticity when modeling the game-theoretic interaction in the simulation, evaluate resulting distributions of key observed variables and compute estimation on the number of samples needed to provide conclusive results.

Experiment Setup

We use a simple scenario which varies two parameters—the visibility radius and the speed of the merchant vessel—to analyze the level of stochasticity, given following random variables: (1)the dock waiting time of the merchant vessel, sampled uniformly from $U(0, 100)$ hours and (2) the strategy, sampled from the probability distribution over simple paths in the graph (computed by the game-theoretic model). It represents the smallest amount of stochasticity required for reasonable game-theoretic model evaluation.

We observe two variables: the number of transits of the merchant vessel and the number of attack attempts on the merchant vessel. From these two variables, we compute the final variable: the attack frequency, which we compare with the expected value of the game.

We set the visibility radius to $r = \{1/0.8, 1, 1/1.2, 1/1.5, 1/2\}$ of the edge length and the speed of merchant vessel to $s_{MV} = \{10, 12, 15, 18, 20\}$. We thus get 25 parameterizations of the scenario and we run 500 instances of each.

Results

Figure 6.2 shows the distribution of samples in attempts/transits space. The data can be fitted neither with Poisson nor with Binomial distributions.

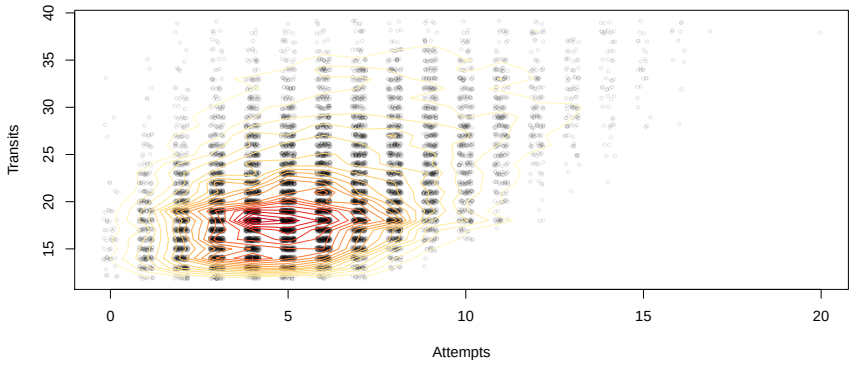
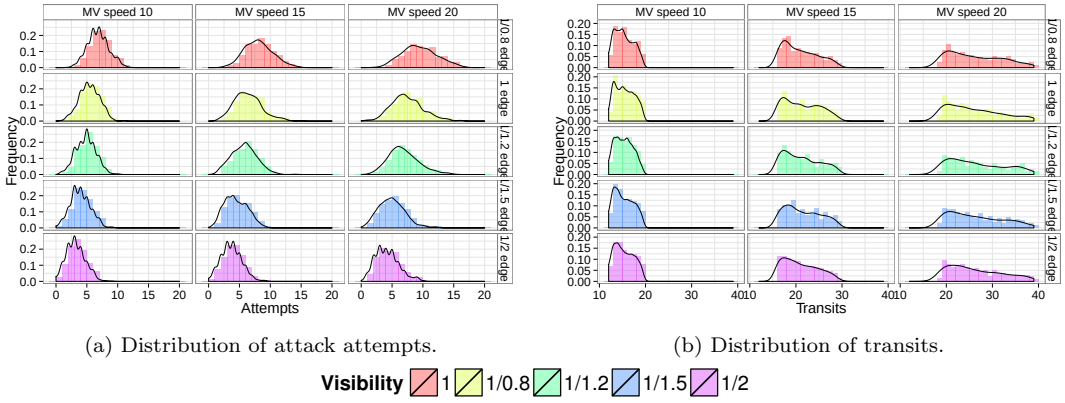


Fig. 6.2: Distribution of samples in attempts/transits space w.r.t. two basic parameters – speed of the merchant vessel and the visibility range. The samples are jittered to accent their density.



(a) Distribution of attack attempts.

(b) Distribution of transits.

Fig. 6.3: Distribution of key measured values w.r.t. the perturbation of visibility and merchant vessel speed.

A closer look, decomposing the output variables by the merchant vessel speed and visibility, reveals elementary distributions when fixing all parameters to a single value (See Figure 6.3a for the distribution of the number of attack attempts while varying visibility and the merchant vessel speed and Figure 6.3b for the distribution of the number of transits while varying the same parameters). The distribution of attack attempts should be influenced mostly by the underlying strategy sampling variable (an be closer to Poisson distribution), the distribution of transit is influenced mostly by a uniform sampling of waiting time of the merchant vessel and should be thus close to a uniform distribution as well. Differences from the expected distributions can be explained by the imprecise pseudo-random generator of Java (Knuth, 2006), the impact of the strategy sampling on the number of transits (where some sampled paths are shorter than others) or by the simulation biases.

We further explore the distribution of the attack frequency on Figure 6.4, which cannot be fitted with any standard distribution as well. The distribution results from the combination of

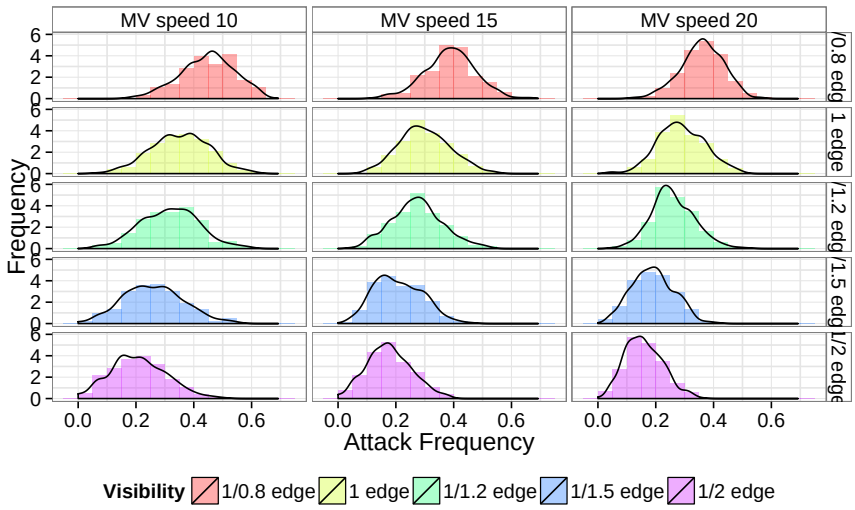


Fig. 6.4: Distribution of attack frequency (i.e., ratio between number of attacks and number of transits) w.r.t. perturbation of visibility and merchant vessel speed.

number of attack attempts and the number of transits and thus combines distributions of those two variables.

The standard deviation varies from $\sigma_{min} = 0.065$ to $\sigma_{max} = 0.10$ and mean varies from $\mu_{min} = 0.16$ to $\mu_{max} = 0.45$. To achieve a relative standard error of the mean $RSEM \leq 5\%$ and assuming the worst case (i.e., standard deviation of 0.10 and mean of 0.16), we have to conduct $N_{max} = (\frac{\sigma_{max}}{SEM})^2 = 175$ experiments per parameter set. When assuming the best case, we have to conduct approximately $N_{min} = 10$ experiments. Due to constraints on computational resources, we conduct $N = 100$ experiments per parameter set in every scenario, unless stated otherwise, relative standard error of the mean $RSEM_{avg} = 3\%$ in an average case (in a scenario with the variation of the merchant vessel speed and visibility and two random variables involved).

6.4 Validation of Game-theoretic Transit and Grouping Mechanisms

This section summarizes a set of scenarios focused on the evaluation of the robustness of the proposed mathematical model. Each scenario modifies one or more parameter of the problem and quantifies the results. The results should not be used for an estimation of the exact value (i.e., the probability of an attack), however, for a demonstration of trends and sensitivity of the observed variable when perturbing parameters of the problem

6.4.1 Space/Time/Visibility Abstraction Validation

In this scenario, we aim to calibrate the visibility parameter to account for a different space-time abstractions in game-theoretic and simulation setting.

Experiment Setup

The representation of space and time is fixed in the simulation—the space is continuous and the time is step-based. The design of the simulation core prohibits a direct modification of these representations.

The vessels are not synchronized—the merchant vessel waits in the origin node to start its journey at a random time. Additionally, the edges do not have a unit length (the diagonal edges are longer) which, even when the vessels would start at the same time, would introduce an additional dis-synchronization during the transit (note that even for other space tilings such that a triangular tiling where all the edges have the same length, the curvature of the Earth would dis-synchronize the movement of the vessels ultimately).

The visibility parameter is set to the same value for both vessels, however, plays an important role for the pirate vessel, which detects and attacks merchant ships in this visibility radius. The visibility can be interpreted in the following ways:

1. Real-world visibility—simulates the curvature of the earth and the visibility by naked eye—approximately 7 nautical miles for 2-meter high vessel.
2. Full-edge visibility—as soon as both vessels start sailing against each other on the same edge, they see each other. Here, another issue arises: the vessels oversee each other on two distinct edges, when close together. This, however, could account for spatial spread of the node.
3. Part-of-edge visibility—this setting represents a compromise between a local and full-edge visibility, i.e., we assume that the vessels can see each other only on the same edge at a certain distance.

For each graph, we have varied the visibility $v = \{1/2, 2/3, 4/5, 1, 4/3\}$ of the length of the vertical edge.

Results

Figure 6.5 shows the result for each graph, having the width from $w = 4$ to $w = 8$. The dotted lines represent a theoretic expected attack frequency. We can deduct, that the best correspondence with the probability of encounter predicted by the game-theoretic model is achieved for the visibility equal to the length of the edge in the graph.

We thus set the visibility parameter to the length of a vertical edge in the evaluated graph for subsequent scenarios.

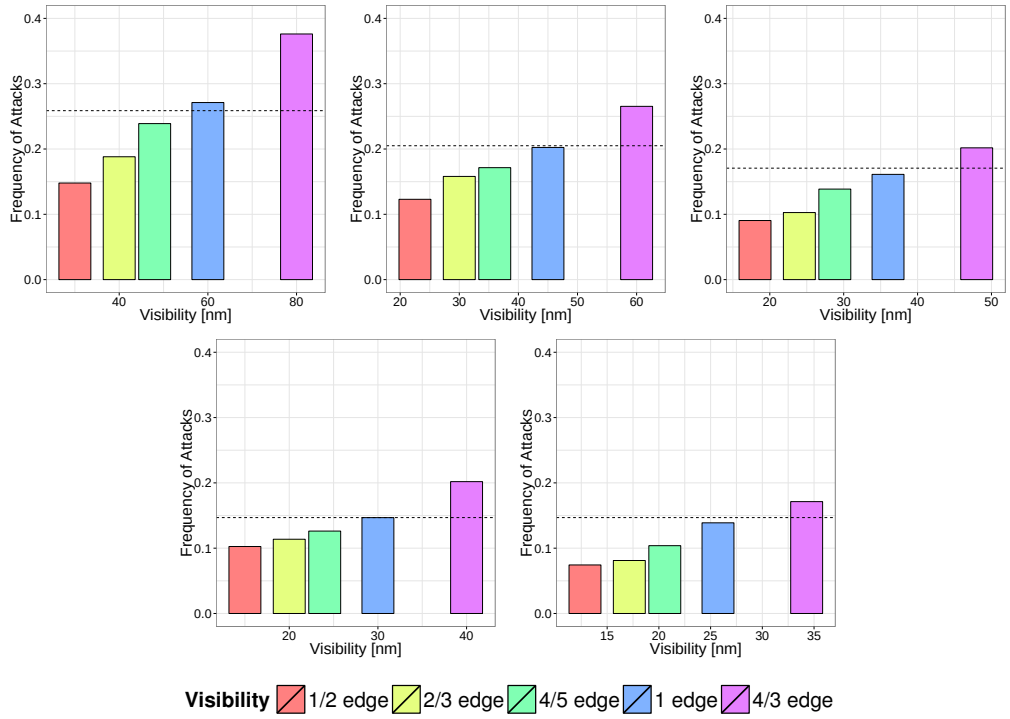


Fig. 6.5: Variation of visibility parameter in each graph ($w = \{4, 5, 6, 7, 8\}$ from upper left to lower right). The dotted lines denote expected frequency of attacks.

6.4.2 Speed Variation

In this experiment, we vary the relative speed of both vessels and we would like to demonstrate the sensitivity of the solution to the assumption of an equal speed of both agents.

A higher relative speed of the pirate vessel should lead to increased probability of catching the merchant vessel and lower relative speed of the pirate vessel should lead to the probability of static patrolling, with the simulation start bias (where the pirate starts in the base), which introduces another drop in the frequency of attacks and with an after-attack bias, which introduces another drop.

Experiment Setup

We fix the pirate to sail at speed $s_{pirate} = 15$ knots and we vary the speed of the merchant vessel $s_{merchant} = \{10, 12, 15, 18, 20\}$, together with the visibility range of the pirate. We evaluate the setup on the graph with width $w = 4$ which has a slightly higher frequency of attacks for the visibility of the vertical edge length.

Results

Figure 6.6 confirms an increased frequency of attacks, when the relative speed of the merchant vessels is lower than the speed of the pirate. And, on the contrary, the frequency of attacks is lower when the merchant vessels sail faster than the pirates. For extreme sailing speeds of

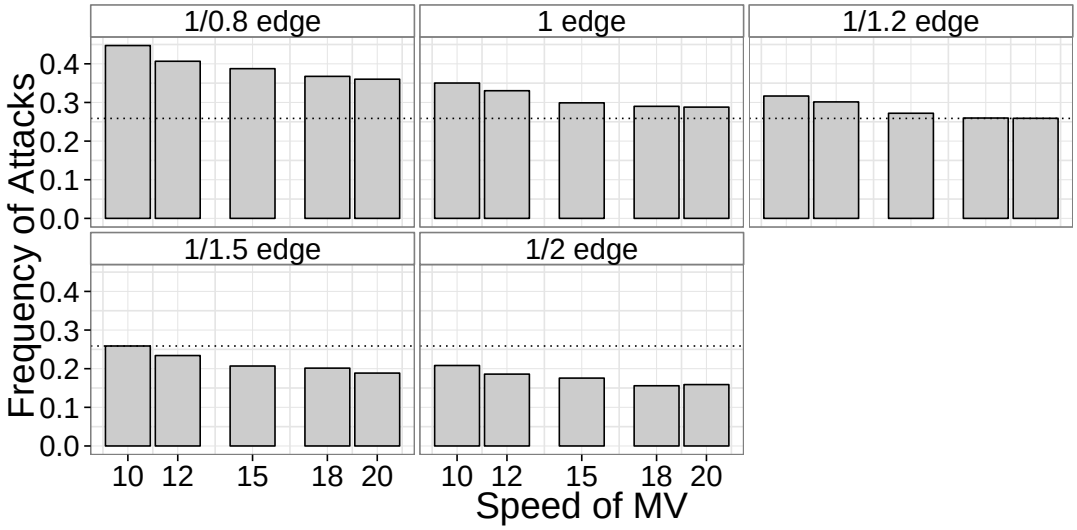


Fig. 6.6: Dependence of frequency of attacks w.r.t. change in visibility and merchant vessel speed on a graph of width $w = 4$.

merchant vessels (i.e., $s_{pirate} << s_{merchant} > 50$ knots), the simulation biases (see Section 6.2) would significantly influence the solution and the evaluation would not provide correct results and trends.

6.4.3 Multiple Merchant Ships and Pirates

The game-theoretic model assumes a single evading and a single intercepting agent. We can violate this assumption in both directions and evaluate the quality of the solution for multiple evading agents, multiple intercepting agents, or both.

Experiment Setup

We vary the number of merchant vessels present in the simulation $MV = \{1, 2, 5, 10, 20\}$ and the number of pirates $P = \{1, 2, 4\}$, all operating independently from each other (i.e., without the grouping scheme or any other communication device). We evaluate this setting on four graphs ($w = \{4, 5, 6, 7\}$) and observe the number of attacks.

Results

The results confirm expected trends: when increasing the number of pirates, the frequency of attacks rises. When increasing the number of merchant vessels, the observed variable is subject to two trends: (1) it is decreasing as the number of merchant vessels rises—the attacks are so frequent that after the attack, the pirate returns to the base without the ability to attack anymore, which allows other vessels to pass without any risk; (2) a slightly increasing attack

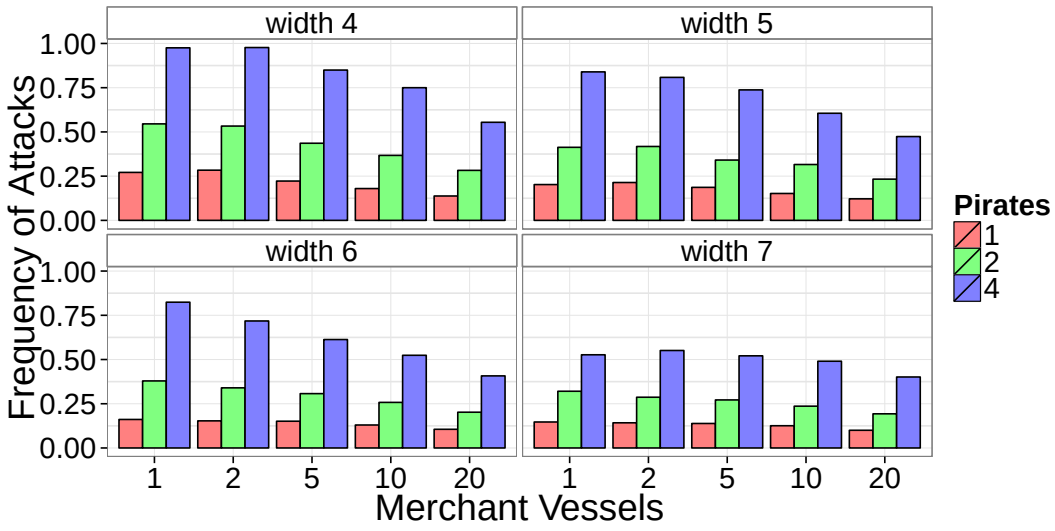


Fig. 6.7: Multiple Merchants and Pirates

frequency, as the number of merchant vessels goes from 1 to 2, visible especially when the number of pirates is high: for some merchant vessel strategies the merchant vessel passes through the base. As the pirates are synchronized (their walks have the same length and they start at the same time at the beginning of the simulation), they all attack the vessel in the base and the number of attacks rises sharply. This effect is, however, overweighted by the first (decreasing) trend.

6.4.4 Mobility Abstraction Validation

We further explore the mobility capabilities of the fixed-base Defender when modifying the length of his walk—in game-theoretic model, the length of the walk had a significant impact on the expect value of the intercepting agent. In this setting, we have to take into account the problem of the simulation time frame vs. the game-theoretic time frame: the execution ends after the interception or area transit in case of the game-theoretic model—the outcome is independent on the distance between the attack location and pirate's base. However, the simulation is not stopped after the attack and/or area transit: the pirate has to return to the base and the overall frequency of attacks thus depends on the distance between the attack location and pirate's base; i.e., for long walks, we can assume that this bias would lower the expected frequency of attacks.

Experiment Setup

To empirically evaluate the influence of the length of the walk on the probability of encounter, we compute optimal Defender's and Evader's strategies on a graph of width $w = 4$ for Defender's walk length $l = \{13, 15, 17, 19\}$ and create a simulation scenario with a merchant vessel and a

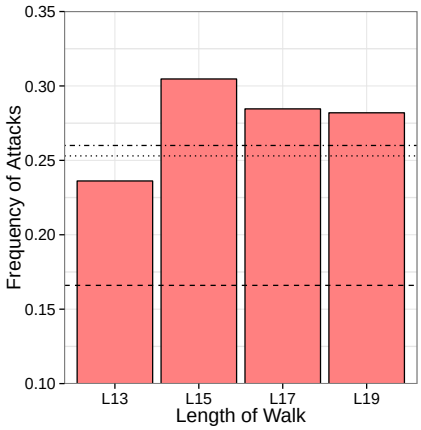


Fig. 6.8: Mobility comparison for single merchant vessels and single pirate for different lengths of pirate's walk. Horizontal lines denote predicted values.

pirate following the computed strategies where we set the visibility of both agents to the width of the edge.

Results

Figure 6.8 captures the results measured in the simulation. The results do not exactly confirm predicted probabilities, however, capture the predicted trend. The increased value for the simulated scenario with pirate's length of 13 is caused by the fact, that the merchant vessel follows a route which passes through the nodes farthest from the base in the graph. Due to the visibility radius, the pirate spends over 1/5th of the time in an area, where he can spot the merchant vessel. The same effect can be observed for the length $l = 15$, where the merchant vessel passes with higher probability through the nodes farthest from the base. The effect is almost mitigated for longer walks (i.e., $l = 17$ and $l = 19$), where the values are closer to those predicted by the model.

When increasing the number of merchant and pirates vessels, we can observe the same trend, described in the previous paragraph (Figure 6.9). Additionally, the saturation trend (i.e., when many merchant vessels are present, the pirate attacks frequently, however, he has to return to the base after the attack without a possibility to attack again, thus wasting some time) when multiple merchant vessels transit the area, is observable.

In this setting, the simulation model contradicts the behavior predicted by the game-theoretic model, which is due to the imprecise relation between different visibility assumptions in both models and due to the contingency bias present in the simulation.

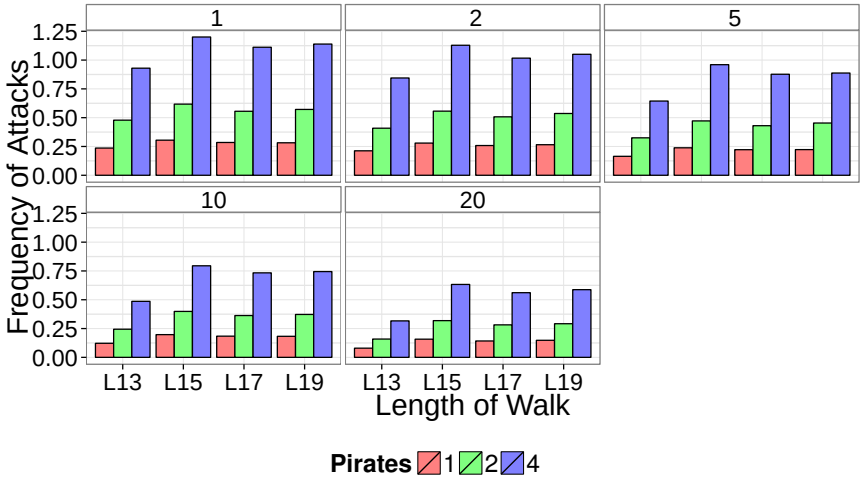


Fig. 6.9: Influence of the length of pirate's walk on the frequency of attacks while varying number of merchant vessels and number of pirates.

6.4.5 Grouping Influence

In this experiment sets, we explore the impact of grouping on the probability of an attack. The grouping should lower the number of attacks due to the following reason: the nature of the pirate's attack prohibits the pirate to attack multiple targets (i.e., merchant vessels) simultaneously. When a pirate attacks a group of merchant vessels, he picks one at random[2] and attacks it. After the attack, the pirate interrupts the strategy execution and returns to the base[3]. This behavior enhances the strength of a group, because if the pirate attacks a group, he will attack only a single vessel and other merchant vessels in the group stay unharmed during the transit.

Experiment Setup

We evaluate the grouping mechanism on graphs of width $w = \{4, 5, 6, 7, 8\}$. We sample merchant vessel speeds sampled from uniform distribution $U(10, 20)$ knots and we turn the grouping mechanism on or off (denoted as TRUE/FALSE). The parameters of the grouping mechanism are as follows: we do not assume any aggregation point and any approach zone, the maximum speed difference in groups is 2 knots, the minimum group size is set to 2, the risk coefficient is set to the maximum value to prefer grouping if possible (see Section 4.2.3.5). The number of merchant vessels is varied from 10 to 50 (where the upper bound is given by the limitation of the grouping algorithm).

[2] the pirate prefers to attack slower vessels when multiple in sight, however, all merchant vessels sail at the same speed, the pirate is thus indifferent.

[3] in real world, the pirate takes some time to recuperate and moves to a different area as the attack is reported to authorities.

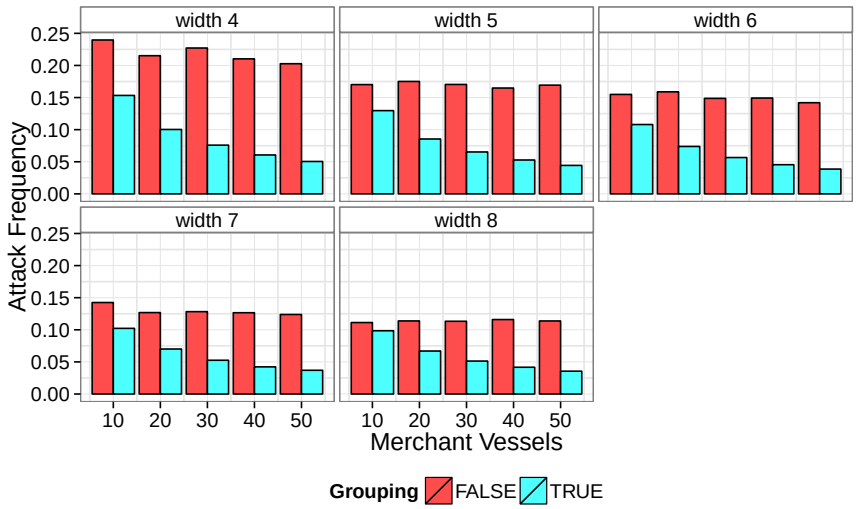


Fig. 6.10: Influence of grouping mechanism on the frequency of attack, measured on graphs with differing width and for a different number of merchant vessels.

Results

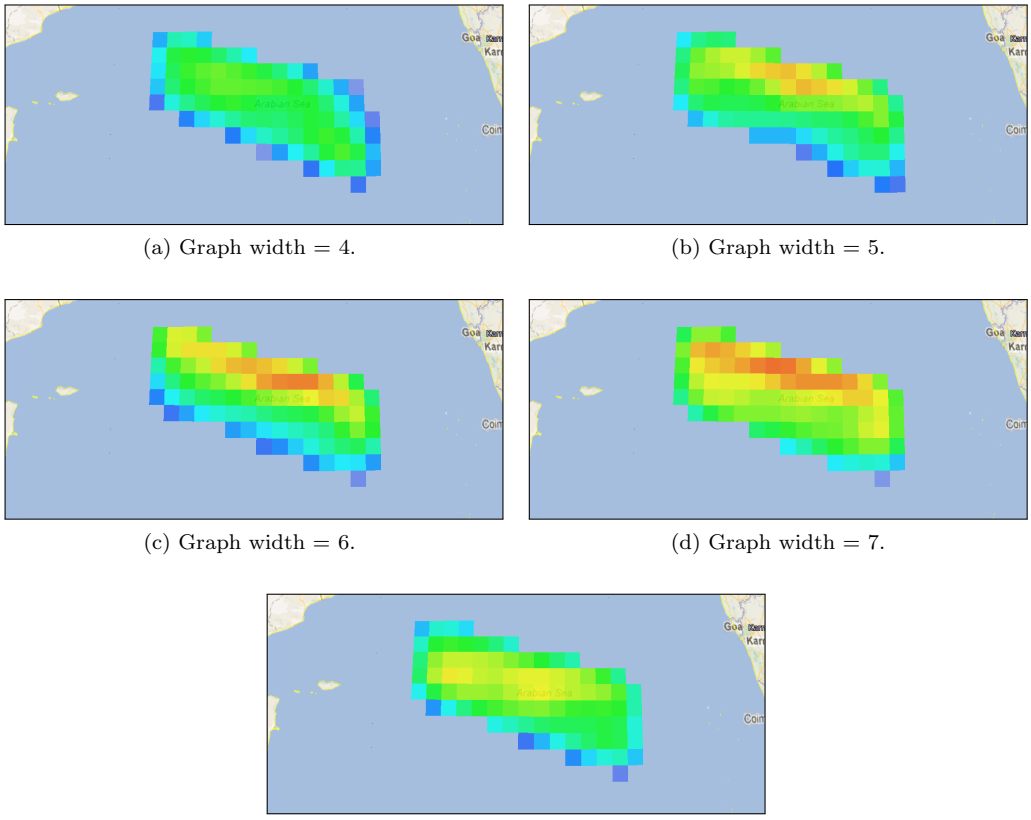
Figure 6.10 quantifies expected results. Grouping significantly reduces the frequency of attacks; additionally, the positive effect of grouping is stronger when more vessels are available: the frequency of attacks drops more than three times for 50 vessels employing the grouping compared to the ungrouped transit.

When comparing the results for 10 merchant vessels, the frequency of attacks is approximately equal for the graph of width $w = 7$ without grouping and for the graph of width $w = 4$ with grouping. This observation brings insight into the trade-off between the aggregation of vessels into groups and the spatial distribution of vessels in a region during the transit.

6.4.6 Aggregate Knowledge Model

We modify the decision-making process of the pirate to be able to utilize the information about the merchant vessel movement from the merchant traffic density map. The pirate is driven by following algorithm:

1. Destination Selection – the pirate selects a cell from the density map using epsilon-greedy strategy with $\epsilon = 0.1$.
2. Destination Transport – the pirate sails directly to the center of the selected destination cell. It is in search mode, i.e., when a merchant vessel is spotted, the pirate attacks the vessel.
3. Cell Patrolling – the pirate stays in the destination cell for a predefined amount of time, moving randomly within the cell's bounds. If no ship is spotted, the pirate selects another cell to move to. In case the same cell is selected, the pirate continues patrolling in this cell.



(a) Graph width = 4.

(b) Graph width = 5.

(c) Graph width = 6.

(d) Graph width = 7.

(e) Graph width = 8.

Fig. 6.11: Traffic density heatmaps for different graphs.

4. Timeout – if the total amount of time allowed for the merchant vessel search is spent, the pirate returns home (again, in search mode, potentially attacking merchant vessels along the way) and selects another destination.

6.4.6.1 Knowledge Representation

Pirates' knowledge about the merchant vessel movement can be represented on two levels: (1) on a finer level of the individual vessel movement, as is the case of the game-theoretic interaction or (2) on the aggregate level of traffic density in the given location. The second representation can be expressed as a *density map* mapped over the observed area. In general, a density map is a two dimensional grid with square cells of a predefined size. Each cell contains a numerical value capturing frequency of an observed event—in our case, a relative frequency of transits through the cell.

To construct the merchant traffic density map, we define a polygon over the observed area, discretize the polygon into a grid with defined cell size $cellSize = 50nm$, we run the simulation with thousands of vessels transiting the area according to a given transit scheme, such as the game-theoretic strategy for a particular graph. We record the number of transits in each cell.

After the simulation, we normalize the values in the cells to be from interval $[0, 1]$ by dividing value in each cell by the maximum value in the grid.

Figure 6.11 captures recorded merchant traffic density heatmaps for different game-theoretic graphs.

Evaluation

Experiment Setup

We have evaluated this scenario on rectangular graphs from width 4 to width 8 with default strategies. Merchant vessel speeds were uniformly sampled from an interval 10-20 knots. We have evaluated the aggregate knowledge impact against the game-theoretic setting assuming perfect knowledge about the discretization etc. We did not use any grouping of the merchant vessels and their number was varied from 1 to 20. We have measured pirate's dockings d and the number of attacks a of the pirate. We then computed pirate's probability of attack per a single trip as $p = a/d$.

Results

Figure 6.12 displays the results from the comparison of the game-theoretic and the adaptive pirate. The aggregate knowledge representation is denoted as TRUE, the game-theoretic as FALSE. The game-theoretic pirate clearly surpasses the adaptive pirate, having over a 2.5 times higher chance of attacking a merchant vessel (in average over all parameters). As stated above, we have used ϵ-greedy strategy for decision making, bringing potentially 10% sub-optimality into the results for the aggregate knowledge pirate.

6.5 Summary

We have described the process of simulation-based validation of solutions computed using models and algorithms described in Chapters 3 and 4. We consider the simulation to be a necessary validation tool serving as an inter-step between an abstract mathematical model and the real-world. As the languages and the level of expression of mathematics, the agent-based simulation and the real-world differ significantly, we provide a brief comparison of each *world* and design sets of experiments moving from the replication of the game-theoretic model as exactly as possible through a slight parameter perturbation and assumption violation to a different knowledge/strategy representation of the intercepting agent, which shows the robustness of the game-theoretic tools developed.

The validation of the grouping mechanism shows an expected superiority of agent aggregation compared to plain individual decision-making without considering other, possibly coordinating agents. Even though a validation of the grouping mechanism against optimal game-theoretic joint plans is not possible, the approach shows that for multiple agents present, it is necessary to take into account a possible collaboration of agents.

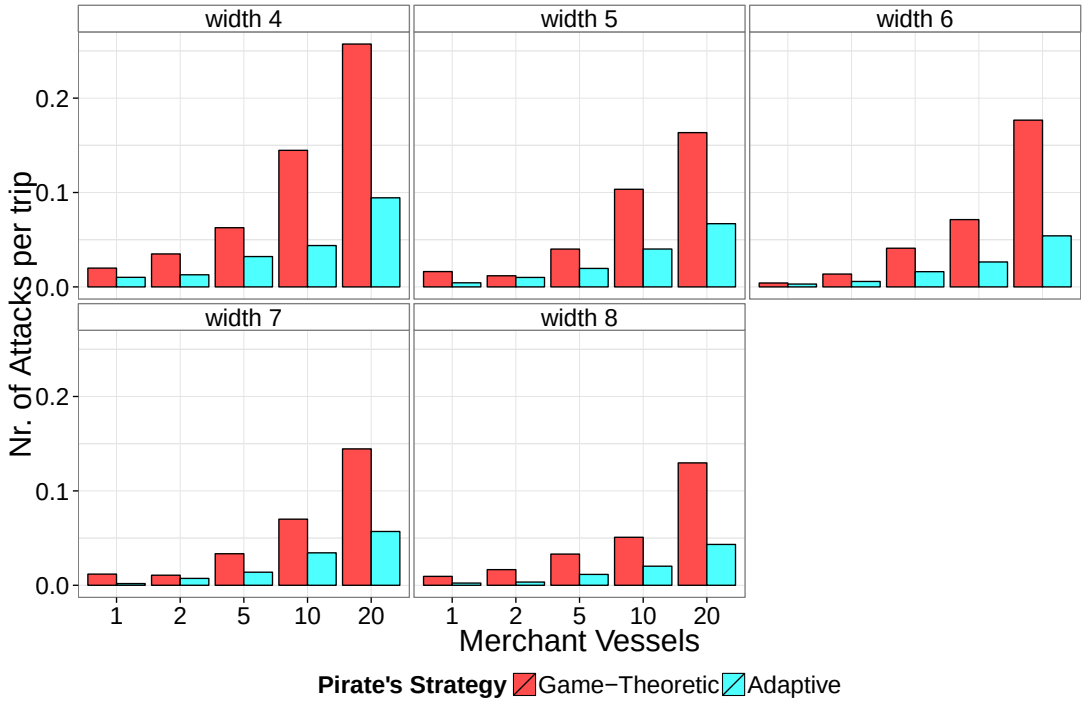


Fig. 6.12: Comparison of attack probability from pirate's perspective on a graph of width $w = 6$ while varying number of merchant vessels and number of pirates, with grouping disabled and using different representation of knowledge. Precisely, we display probability of an attack per pirate's trip given that the time of a trip is equal for the adaptive and game-theoretic pirate.

The results of the simulation-based evaluation was presented as part of the simulation framework Jakob et al. (2011c, 2010a) as well as the game-theoretic model Vaněk et al. (2010).

Chapter 7
Conclusions

Concluding remarks, a summary of the work done and goals achieved

Finally, there are no solutions for the most urgent problems but only restatements without promising perspectives.

W. Provost

The goal of the thesis was to contribute to a set of methods for modeling the problem of security of a transportation system under a persistent threat of an attack, to extend existing algorithms for the design of policies for secure movement of transportation agents, and to develop a simulation-based validation framework and use it for the assessment of quality of computed solutions.

We have proposed a formal game-theoretic model of a randomized transit through guarded areas. This model can be either used for the design of routes minimizing the probability of encounter or for the design of strategies suitable for the defending agent aiming to maximize the probability of encounter. We introduce a concept of oracle hierarchies and the decomposition of the oracle-based algorithms into an expansion and termination check phase. This step opens up a lot of promising research directions quantified below.

The evaluation of the model shows that resulting strategies are not trivial, however, a rule of thumb can be extracted for both the Evader and the Defender: (1) The Evader should fully randomize over possible paths almost uniformly in case of homogeneous environment and with not significantly constrained Defender, slightly preferring routes farther from the base; (2) the Defender walks have to intercept every possible Evader's route at least in one node. The positioning of the base in a bottleneck of the area has a significant advantage, especially when the walk length is restricted. For a very long walk (with respect to the width of the area bottleneck), the mobility of the base is not important, for a short walk, the mobility of the base is vital.The mobility of the Defender has a positive impact on its expected game value, as well as the ability to wait on place.

To cope with a possibly large number of transiting players, we assume single-attack saturation property of interceptor's behavior and propose a grouping mechanism aggregating transiting agents into groups, taking into account varying speeds and an individual risk-aversion weight for each agent. The structure of optimal groups of the grouping problem is non-trivial, most frequently grouping agents with speeds in the middle of the interval and agents who are further from the entry point of the area (especially in case of an aggregation point constraint). The algorithm scales to tens of agents and the number of agents has the biggest impact on the runtime of the algorithm—the dependency is exponential, which is expected, given the NP-hardness of the problem. We additionally demonstrate the superiority of our grouping mechanism to the

currently deployed fixed grouping scheme for a secure vessel transit through the Gulf of Aden with respect to the delay caused and the number of vessels grouped.

Finally, we design a simulation of one of the problem domains—an agent-based simulation of maritime traffic in the high seas infested with maritime pirates. The simulation is able to simulate a year of movement of thousands of vessels and tens of pirates in tens of minutes. We decompose the agent model into a population model—parameterized by a set of parameters configuring a complete population of one agent class—and an individual behavior model—parameterized by a set of parameters configuring an individual behavior of each agent, together with its decision-making process. This decomposition allows as to implement various behaviors and strategies for each agent, being it an intelligent route planner for merchant vessels or a game-theoretic spot selection for pirates. The scope of the simulation is much wider than a single purpose of the evaluation of interaction of two (or more) agents: the simulation was successfully deployed in the Naval Research Laboratory in Monterey where it is used for the prediction of pirate activity. We use the simulation for the validation of the designed solution of transit security: the robustness of the game-theoretic strategies is high with respect to the assumption violation as well as the scenario parameter modification; this shows a high potential for the utilization of game-theoretic models for real-world security.

Contributions presented in this thesis open multiple research questions: (1) the decomposition of the oracle-based algorithms into sub-game expansion check and termination check allows to integrate methods for the support sets search (Porter et al., 2004); additionally, if we relax the optimality requirement, we can focus on the design of very fast oracles proving suboptimal responses very quickly. Also, parallelization of oracle-based algorithms should be straightforward and would lead to considerable speedups—the question of load-balancing and exchange of partial solutions and strategies is, however, non-trivial and unanswered. (2) Grouping mechanisms present a wide range of problems which can be solved within the optimization framework and allow us to use all available approaches, including a wide range of metaheuristics; the exploration of this branch would allow us to define more complex criterion functions as well as imposed constraint sets. (3) Simulation models allowing non-trivial evaluation of designed solutions are required for many domains, their design is often, however, ad hoc, leading to proprietary design and validation methodologies. The final goal of the thesis is to inspire researchers to contribute to the field of simulation-based validation methodology.

7.1 Thesis Achievements

This section summarizes the contribution of the thesis to the state-of-the-art of transportation security. The achievements are the following:

1. Formal model of transit security with two mobile players termed *transit game*. The model is based on a two-player zero-sum game in a normal form and considers a mobile defender with a fixed or mobile base and an unconstrained defender, modeling various scenarios with different movement constraints of the defender. The utility model is probabilistic and considers uncertain encounters to account for limited capabilities of the intercepting player or inherent detection uncertainty in the environment. This achievement successfully fulfills the first goal of this thesis. The complexity of the utility function and size of players' strategy spaces makes the second goal of the thesis non-trivial.

2. Design of oracles for both players which would allow to utilize double-oracle algorithms for solving large games (as is this case). For each mobility model (and for both players), a novel oracle able to provide best-response for any game considered was designed. As the computational complexity of the oracles is high, suboptimal, faster oracles were designed to achieve a higher scalability of the double-oracle algorithm.

3. Proposition of oracle hierarchies as an extension of oracle-based algorithms used to solve large two-player zero-sum games in a normal form. Oracle hierarchies allow to use faster, suboptimal oracles without the loss of optimality guarantee thanks to the idea of the decomposition of the algorithm into sub-game expansion and termination check phases. The suboptimal oracles can be used in the sub-game expansion phase, thus speeding up the algorithm, whereas the optimal oracles are still needed for the termination check phase. Together with previous achievement, this state-of-the-art contribution fulfills the definition of the second goal, even though, due to the exponential size of players' strategy spaces and complexity of the oracles, the scalability remains limited.

4. Formalization of the optimal grouping problem and design of mathematical bi-objective problem able to solve the problem for different parameters. The grouping model is motivated by current needs of merchant vessels in pirate-infested areas and considers exact constraints posed on the simpler grouping mechanism currently deployed as well as more relaxed variants suitable for scenarios without restrictions posed by narrow corridors. The algorithm is scalable to tens of vessels for all constraints set as evaluated on a number of scenarios and guarantees a better grouping for existing corridor transits with respect to both the delay caused and the number of vessels grouped. This contribution resolves the third research goal of solutions for models with multiple players.

5. Design and implementation of agent-based simulation of the maritime domain with all actors involved in the conflict of maritime piracy, calibrated on real-world data. The simulation is scalable, modular, with easy-to-implement agent behavior architecture in a form of activity diagrams which allow the reuse of the simulation for different scenarios. This achievement paved the way for the fourth goal of the thesis, the simulation-based evaluation. Additionally, the simulation was successfully deployed in Naval Research Laboratory in Monterey to predict

pirate activity in the Indian Ocean based on weather reports and behavior models of pirates.

6. Integration of strategies computed using the game-theoretic models and grouping mechanism into the simulation and validation of its robustness with respect to violation of assumptions made and parameter values considered. The evaluation shows a robust behavior of game-theoretic strategies, its superiority with respect to simpler knowledge models, however, also difficulties in the estimation of the final expected quality of the solution for either of the players. This achievement contributes to the fourth and final goal of the thesis and opens many new research questions about iterative model refinements using the simulation-based validation.

7.2 Selected Related Publications

This section summarizes the author's selected publications related to the content of the thesis.

Journal Articles and Book Chapters (5)

1. **Ondřej Vaněk**, Ondřej Hrstka, and Michal Pěchouček. Improving group transit schemes to minimize negative effects of maritime piracy (submitted). *Transactions on Intelligent Transportation Systems*, 2013. (**80%**)
2. **Ondřej Vaněk**, Michal Jakob, Ondřej Hrstka, and Michal Pěchouček. Agent-based model of maritime traffic in piracy-affected waters (submitted). (**60%**) *Transportation Research Part C: Emerging Technologies*, 2013.
3. **Ondřej Vaněk**, Jakob Michal, and Michal Pěchouček. Using Data-Driven Simulation for Analysis of Maritime Piracy, chapter in *Prediction and Recognition of Piracy Efforts Using Collaborative Human-Centric Information Systems*. In (Bosse, 2013), 2013. (**90%**)
4. Michal Jakob, **Ondřej Vaněk**, and Michal Pěchouček. Using agents to improve international maritime transport security. *Intelligent Systems, IEEE*, 26(1):90–96, 2011. (**40%**)
5. **Ondřej Vaněk**. Agent-based simulation of the maritime domain. *Acta Polytechnica*, 50:94–99, 2010. (**100%**)

In Proceedings (12)

1. **Ondřej Vaněk**, Ondřej Hrstka, and Michal Pěchouček. Dynamic group transit scheme for corridor transit. In *Proceedings of the International Conference on Modeling, Simulation and Optimization (ICMSAO)*, 2013. (**90%**)
2. **Ondřej Vaněk**, Zhengyu Yin, Manish Jain, Branislav Bošanskỳ, Milind Tambe, and Michal Pěchouček. Game-theoretic resource allocation for malicious packet detection in computer networks. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, volume 1, pages 905–912. International Foundation for Autonomous Agents and Multiagent Systems, 2012. (**37%**)
3. Michal Jakob, **Ondřej Vaněk**, Ondřej Hrstka, and Michal Pěchouček. Agents vs. pirates: multi-agent simulation and optimization to fight maritime piracy. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, volume 1, pages 37–44. International Foundation for Autonomous Agents and Multiagent Systems, 2012. (**40%**)
4. **Ondřej Vaněk**, Michal Jakob, Ondřej Hrstka, and Michal Pěchouček. Using multi-agent simulation to improve the security of maritime transit. In *Proceedings of the Multi-Agent-Based Simulation Workshop (MABS)*, pages 44–58. Springer Berlin Heidelberg, 2012. (**40%**)

5. **Ondřej Vaněk**, Branislav Bošanský, Michal Jakob, Viliam Lisý, and Michal Pěchouček. Extending security games to defenders with constrained mobility. In *Proceedings of AAAI Spring Symposium, GTSSH*, 2012. (**50%**)
6. Branislav Bošanský, **Ondřej Vaněk**, and Michal Pěchouček. Strategy representation analysis for patrolling games. In *Proceedings of AAAI Spring Symposium GTSSH*, 2012. (**15%**)
7. Manish Jain, Dmytro Korzhyk, **Ondřej Vaněk**, Vincent Conitzer, Michal Pěchouček, and Milind Tambe. A double oracle algorithm for zero-sum security games on graphs. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, volume 1, pages 327–334. 2011. (**25%**)
8. Michal Jakob, **Ondřej Vaněk**, Branislav Bošanskỳ, Ondřej Hrstka, and Michal Pěchouček. Agentc: agent-based system for securing maritime transit (demonstration). In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, volume 3, pages 1309–1310. 2011. (**35%**)
9. **Ondřej Vaněk**, Michal Jakob, Viliam Lisỳ, Branislav Bošanskỳ, and Michal Pěchouček. Iterative game-theoretic route selection for hostile area transit and patrolling. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1273–1274, 2011. (**40%**)
10. **Ondřej Vaněk**. Security games with mobile patrollers (extended abstract). In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, volume 3, pages 1371–1372. 2011. (**100%**)
11. Michal Jakob, **Ondřej Vaněk**, Štěpán Urban, Petr Benda, and Michal Pěchouček. Agentc: agent-based testbed for adversarial modeling and reasoning in the maritime domain. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, volume 1, pages 1641–1642. 2010. (**30%**)
12. **Ondřej Vaněk**, B Bošanský, Michal Jakob, and Michal Pěchouček. Transiting areas patrolled by a mobile adversary. In *Proceedings of the Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*, pages 9–16. IEEE, 2010. (**40%**)

References

Adler, M., Racke, H., Sivadasan, N., Sohler, C., Vocking, B., 2002. Randomized pursuit-evasion in graphs.

Agmon, N., Kraus, S., Kaminka, G., 2008a. Multi-robot perimeter patrol in adversarial settings. In: Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on. IEEE, pp. 2339–2345.

Agmon, N., Sadov, V., Kaminka, G., Kraus, S., 2008b. The impact of adversarial knowledge on adversarial planning in perimeter patrol. In: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 1. International Foundation for Autonomous Agents and Multiagent Systems, pp. 55–62.

Allen, A. O., 1990. Probability, statistics, and queueing theory: with computer science applications. Academic Pr.

Alpern, S., 1992. Infiltration Games on Arbitrary Graphs. Journal of Mathematical Analysis and Applications 163, 286–288.

Alpern, S., Gal, S., 2003. The theory of search games and rendezvous. Springer.

Auger, J., 1991a. An infiltration game on k arcs. Naval Research Logistics 38 (4), 511–529.

Auger, J., 1991b. On discrete games of infiltration. Differential Games - Developments in Modelling and Computation, 144–150.

Baldacci, R., Maniezzo, V., Mingozzi, A., 2004. An exact method for the car pooling problem based on lagrangean column generation. Operations Research 52 (3), 422–439.

Barnhart, C., Johnson, E., Anbil, R., Hatay, L., 1994a. A column-generation technique for the long-haul crew-assignment problem. In: Optimization in industry 2. John Wiley & Sons, Inc., pp. 7–24.

Barnhart, C., Johnson, E., Nemhauser, G., Savelsbergh, M., 1994b. Branch and Price: Column Generation for Solving Huge Integer Programs. Operations Research 46, 316–329.

Barton, D., Stamber, K., 2000. An agent-based microsimulation of critical infrastructure systems. Tech. rep., Sandia National Labs., Albuquerque, NM (US); Sandia National Labs., Livermore, CA (US).

Basilico, N., Gatti, N., Amigoni, F., 2009a. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In: Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1. International Foundation for Autonomous Agents and Multiagent Systems, pp. 57–64.

Basilico, N., Gatti, N., Rossi, T., Ceppi, S., Amigoni, F., 2009b. Extending algorithms for mobile robot patrolling in the presence of adversaries to more realistic settings. In: Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology-Volume 02. IEEE Computer Society, pp. 557–564.

Basilico, N., Gatti, N., Villa, F., 2010. Asynchronous Multi-Robot Patrolling Against Intrusions in Arbitrary Topologies. In: Proceedings of the National Conference on Artificial Intelligence (AAAI). pp. 1224–1229.

Benders, J., 1962. Partitioning procedures for solving mixed-variables programming problems. Numerische Mathematik 4 (1), 238–252.

Beringer, J., Hüllermeier, E., 2006. Online clustering of parallel data streams. Data & Knowledge Engineering 58 (2), 180–204.

Boel, R., Mihaylova, L., 2006. A compositional stochastic model for real time freeway traffic simulation. Transportation Research Part B: Methodological 40 (4), 319–334.

Bosse, E. (Ed.), 2013. Prediction and Recognition of Piracy Efforts Using Collaborative Human-Centric Information Systems. IOS Press.

Bourdon, S., Gauthier, Y., Greiss, J., 2007. MATRICS: A Maritime Traffic Simulation. Tech. rep., Defence R&D Canada.

Bošanský, B., Lisý, V., Jakob, M., Pěchouček, M., 2011. Computing time-dependent policies for patrolling games with mobile targets. In: 10th International Conference on Autonomous Agents and Multiagent Systems.

Bowden, A., Hurlburt, K., Aloyo, E., Marts, C., Lee, A., 2011. The economic costs of maritime piracy. Tech. rep., Oceans Beyond Piracy, One Earth Future Foundation.

Brooks, R. R., Schwier, J., Griffin, C., 2009. Markovian Search Games in Heterogeneous Spaces. IEEE Trans. Sys. Man and Cyber. B 39 (3), 626–335.

Brown, G., Carlyle, M., Salmerón, J., Wood, K., 2006. Defending critical infrastructure. Interfaces 36 (6), 530–544.

Bruzzone, A., Massei, M., Madeo, F., Tarone, F., Gunal, M., 2011. Simulating marine asymmetric scenarios for testing different C2 maturity levels. In: Proceedings of the 16th International Command and Control Researh and Technology Symposium. pp. 12–23.

CAs, O.-D., 2007. Modeling dynamic systems with cellular automata. Handbook of Dynamic System Modeling.

Chandran, R., Beitchman, G., 29 November 2008. Battle for Mumbai Ends, Death Toll Rises to 195. Times of India.

Chardaire, P., McKeown, G., Verity-Harrison, S., Richardson, S., 2005. Solving a time-space network formulation for the convoy movement problem. Operations Research 53 (2), 219–230.

Charnes, A., Cooper, W. W., Ferguson, R. O., 1955. Optimal estimation of executive compensation by linear programming. Management science 1 (2), 138–151.

Chawdhry, P., 2009. Risk modeling and simulation of airport passenger departures process. In: Proceedings of the 2009 Winter Simulation Conference. IEEE, pp. 2820–2831.

Chen, M., Kandlur, D., Yu, P., 1993. Optimization of the grouped sweeping scheduling (gss) with heterogeneous multimedia streams. In: Proceedings of the first ACM international conference on Multimedia. ACM, pp. 235–242.

Chen, Y., Tu, L., 2007. Density-based clustering for real-time stream data. In: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pp. 133–142.

Cheng, P., 2003. A Short Survey on Pursuit-Evasion Games. Department of Computer Science, University of Illinois at Urbana-Champaign.

Chung, C., Fabbri, A., 2003. Validation of spatial prediction models for landslide hazard mapping. Natural Hazards 30 (3), 451–472.

Chvátal, V., 1975. A combinatorial theorem in plane geometry. J. Combin. Theory Ser. B 18 (39-41), 32.

Cordeau, J., Laporte, G., 2003. The dial-a-ride problem (darp): Variants, modeling issues and algorithms. 4OR: A Quarterly Journal of Operations Research 1 (2), 89–101.

Cormen, T., Leiserson, C., Rivest, R., Stein, C., 2001. Introduction to algorithms. MIT press.

CPLEX, I., 2005. High-performance software for mathematical programming and optimization. U RL http://www. ilog. com/products/cplex.

Dantzig, G., Wolfe, P., 1960. Decomposition principle for linear programs. Operations research 8 (1), 101–111.

Davidsson, P., 2001. Multi agent based simulation: beyond social simulation. Springer.

Decraene, J., Anderson, M., Low, M., 2010. Maritime counter-piracy study using agent-based simulations. In: Proceedings of the 2010 Spring Simulation Multiconference. p. 165.

Dickerson, J., Simari, G., Subrahmanian, V., Kraus, S., 2010. A Graph-Theoretic Approach to Protect Static and Moving Targets from Adversaries. In: AAMAS. pp. 299–306.

Edgeworth, F. Y., 1881. Mathematical psychics: An essay on the application of mathematics to the moral sciences. No. 10. CK Paul.

Ellis, J., Warren, R., 2008. Lower bounds on the pathwidth of some grid-like graphs. Discrete Applied Mathematics 156 (5), 545–555.

Ferguson, T., 2005. Game Theory - Lecture Notes. http://www.math.ucla.edu/~tom/math167.html.
URL http://www.math.ucla.edu/~tom/math167.html

Franklin, S., Graesser, A., 1997. Is it an agent, or just a program?: A taxonomy for autonomous agents. In: Intelligent agents III agent theories, architectures, and languages. Springer, pp. 21–35.

Fudenberg, D., Tirole, J., 1993. Game Theory. MIT Press.

Gal, S., 1980. Search Games. Academic Press, New York.

Garnaev, A., Garnaeva, G., Goutal, P., 1997. On the infiltration game. International Journal of Game Theory 26 (2), 215–221.
URL http://ideas.repec.org/a/spr/jogath/v26y1997i2p215-221.html

Ghanmi, A., Boukhtouta, A., Sebbah, S., 2011. Modeling and simulation of military tactical logistics distribution.

Goldstein, D., Shehab, T., Casse, J., Lin, H., 2010. On the formulation and solution of the convoy routing problem. Transportation Research Part E: Logistics and Transportation Review 46 (4), 520–533.

Guha, S., Mishra, N., Motwani, R., O'Callaghan, L., 2000. Clustering data streams. In: Foundations of computer science, 2000. proceedings. 41st annual symposium on. IEEE, pp. 359–366.

Haimes, Y. Y., Lasdon, L. S., Wismer, D. A., 1971. On a bicriterion formulation of the problems of integrated system identification and system optimization. IEEE Transactions on Systems, Man, and Cybernetics 1 (3), 296–297.

Halvorson, E., Conitzer, V., Parr, R., 2009. Multi-step Multi-sensor Hider-seeker Games. In: IJCAI. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 159–166.

Hansen, J., Hsu, L., Dykes, J., Dastugue, J., Allard, R., Barron, C., Abramson, M., Russell, S., Mittu, R., 2011. Information domination: Dynamically coupling METOC and INTEL for improved guidance for piracy interdiction. In: NRL Review. Naval Research Laboratory, pp. 109–115.

Hasegawa, K., Hata, K., Shioji, M., Niwa, K., Mori, S., Fukuda, H., 2004. Maritime traffic simulation in congested waterways and its applications. In: 4th Conference for New Ship and Marine Technology, Chine. pp. 195–199.

Hrstka, O., Vaněk, O., 2011. Optimizing group transit in the gulf of aden. In: Proceedings of the 15th International Student Conference on Electrical Engineering (CVUT POSTER).

Hutchins, C., 2013. Analyzing naval strategy for counter-piracy operations, using the massive multiplayer online war game leveraging the internet (mmowgli) and discrete event simulation (des).

Hwang, C. L., Masud, A. S. M., et al., 1979. Multiple objective decision making-methods and applications. Vol. 164. Springer.

Intertanko, Mar. 2009. Gulf of aden internationally recommended transit corridor & group transit explanation @ONLINE.
URL http://www.intertanko.com/upload/IRTC%20%20GT%20Explanation%20-%20March%202009%20(2).pdf

Jain, M., Kardes, E., Kiekintveld, C., Ordonez, F., Tambe, M., 2010a. Security Games with Arbitrary Schedules: A Branch and Price Approach. In: AAAI.

Jain, M., Korzhyk, D., Vaněk, O., Conitzer, V., Pěchouček, M., Tambe, M., 2011a. A double oracle algorithm for zero-sum security games on graphs. In: Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS). Vol. 1. International Foundation for Autonomous Agents and Multiagent Systems, pp. 327–334.

Jain, M., Tsai, J., Pita, J., Kiekintveld, C., Rathi, S., Tambe, M., Ordonez, F., 2010b. Software Assistants for Randomized Patrol Planning for the LAX Airport Police and the Federal Air Marshals Service. Interfaces 40, 267–290.

Jain, M., Vaněk, O., Korzhyk, D., Conitzer, V., Tambe, M., Pěchouček, M., 2011b. Double oracle algorithm for zero-sum security games on graphs. In: 10th International Conference on Autonomous Agents and Multiagent Systems.

Jakob, M., Moler, Z., Komenda, A., Yin, Z., Jiang, A. X., Johnson, M. P., Pěchouček, M., Tambe, M., 2012a. Agentpolis: towards a platform for fully agent-based modeling of multi-modal transportation. In: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 3. International Foundation for Autonomous Agents and Multiagent Systems, pp. 1501–1502.

Jakob, M., Vaněk, O., Bošanskỳ, B., Hrstka, O., Pěchouček, M., 2011a. Agentc: agent-based system for securing maritime transit (demonstration). In: Proceedings of the 10th International Conference on Autonomous Agents

and Multiagent Systems (AAMAS). Vol. 3. International Foundation for Autonomous Agents and Multiagent Systems, pp. 1309–1310.

Jakob, M., Vaněk, O., Hrstka, O., Pěchouček, M., 2012b. Agents vs. pirates: multi-agent simulation and optimization to fight maritime piracy. In: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS). Vol. 1. International Foundation for Autonomous Agents and Multiagent Systems, pp. 37–44.

Jakob, M., Vaněk, O., Urban, Š., Benda, P., Pěchouček, M., 2010a. Agentc: agent-based testbed for adversarial modeling and reasoning in the maritime domain. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS). Vol. 1. International Foundation for Autonomous Agents and Multiagent Systems, pp. 1641–1642.

Jakob, M., Vaněk, O., Bošanskỳ, B., Hrstka, O., Pěchouček, M., 2010b. Adversarial modeling and reasoning in the maritime domain year 2 report. Tech. rep., ATG, CTU, Prague.

Jakob, M., Vaněk, O., Bošanskỳ, B., Hrstka, O., Pěchouček, M., 2011b. Adversarial modeling and reasoning in the maritime domain year 3 report. Tech. rep., ATG, CTU, Prague.

Jakob, M., Vaněk, O., Pěchouček, M., 2011c. Using agents to improve international maritime transport security. Intelligent Systems, IEEE 26 (1), 90–96.

Jakob, M., Vaněk, O., Urban, S., Benda, P., Pěchouček, M., 2009. Adversarial modeling and reasoning in the maritime domain year 1 report. Tech. rep., ATG, CTU, Prague.

Joseph, F., 2005. Path-planning strategies for ambush avoidance. Ph.D. thesis, MASSACHUSETTS INSTITUTE OF TECHNOLOGY.

Joseph, F., Feron, E., 2005. Computing the Optimal Mixed Strategy for Various Ambush Games.

Khan, M., Boloni, L., 2005. Convoy driving through ad-hoc coalition formation. In: Real Time and Embedded Technology and Applications Symposium, 2005. RTAS 2005. 11th IEEE. IEEE, pp. 98–105.

Khan, M. E., 2007. Game theory models for pursuit evasion games. Tech. rep., University of British Columbia, Vancouver.

Kiekintveld, C., Islam, T., Kreinovich, V., 2013. Security games with interval uncertainty. future 27, 15.

Kiekintveld, C., Jain, M., Tsai, J., Pita, J., Ordóñez, F., Tambe, M., 2009. Computing optimal randomized resource allocations for massive security games. In: Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1. International Foundation for Autonomous Agents and Multiagent Systems, pp. 689–696.

Kiekintveld, C., Marecki, J., Tambe, M., 2011. Approximation methods for infinite bayesian stackelberg games: Modeling distributional payoff uncertainty. In: The 10th International Conference on Autonomous Agents and Multiagent Systems.

Klügl, F., 2009. Agent-based simulation engineering. Ph.D. thesis, Habilitation Thesis, University of Würzburg.

Knuth, D. E., 2006. The art of computer programming. Vol. 3. addison-Wesley.

Koch, D., 2007. PortSim - a port security simulation and visualization tool. In: Proceedings of 41st Annual IEEE International Carnahan Conference on Security Technology. IEEE, pp. 109–116.

Komenda, A., Vokriinek, J., Caap, M., Peechoucek, M., 2013. Developing multiagent algorithms for tactical missions using simulation. Intelligent Systems, IEEE 28 (1), 42–49.

Kose, E., Basar, E., Demirci, E., Goeroglu, A., Erkebay, S., 2003. Simulation of marine traffic in Istanbul strait. Simulation Modelling Practice and Theory 11 (7-8), 597–608.
URL `http://dblp.uni-trier.de/db/journals/simpra/simpra11.html#KoseBDGE03`

Kullback, S., Leibler, R., 1951. On information and sufficiency. The Annals of Mathematical Statistics 22 (1), 79–86.

Kumar, P., Narendran, T., 2010. Convoy movement problem–an optimization perspective. Innovations in Defence Support Systems–1, 79–93.

Kumar, P., Narendran, T., Sivakumar, A., 2009. Bi-criteria convoy movement problem. The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology 6 (3), 151–164.

LaPaugh, A., 1993. Recontamination does not help to search a graph. Journal of the ACM (JACM) 40 (2), 224–245.

Lauren, M., Stephen, R., 2002. Map-aware non-uniform automata (MANA)-a new zealand approach to scenario modelling. Journal of Battlefield Technology 5, 27–31.

Lund, C., Yannakakis, M., 1994. On the hardness of approximating minimization problems. Journal of the ACM (JACM) 41 (5), 960–981.

Luo, Y., Bölöni, L., 2007. Children in the forest: towards a canonical problem of spatio-temporal collaboration. In: Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems. ACM, p. 238.

McMahan, H. B., Gordon, G. J., Blum, A., 2003. Planning in the Presence of Cost Functions Controlled by an Adversary. In: ICML. pp. 536–543.

Montana, D., Bidwell, G., Vidaver, G., Herrero, J., 1999. Scheduling and route selection for military land moves using genetic algorithms. In: Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on. Vol. 2. IEEE.

Murata, T., 1989. Petri nets: Properties, analysis and applications. Proceedings of the IEEE 77 (4), 541–580.

Nisan, N., Roughgarden, T., Tardos, E., Vazirani, V. V., 2007. Algorithmic game theory. Cambridge University Press.

Onuoha, F. C., 2010. Piracy and maritime security off the Horn of Africa: Connections, causes, and concerns. African Security 3 (4), 191–215.

Pareto, V., 1964. Cours d'economie politique. Librairie Droz.

Parsons, T. D., 1976. Theory and Applications of Graphs. Springer-Verlag, Ch. Pursuit-evasion in a graph, pp. 426–441.

Pita, J., Jain, M., Ordonez, F., Portway, C., Tambe, M., Western, C., Paruchuri, P., Kraus, S., 2009. Using game theory for los angeles airport security. AI Magazine 30 (1), 43.
URL http://www.aaai.org/ojs/index.php/aimagazine/article/view/2173

Pita, J., Jain, M., Western, C., Portway, C., Tambe, M., Ordonez, F., Kraus, S., Parachuri, P., 2008. Deployed ARMOR protection: The application of a game-theoretic model for security at the Los Angeles International Airport.

Porter, R., Nudelman, E., Shoham, Y., 2004. Simple search methods for finding a nash equilibrium. In: Proceedings of the national Conference on Artificial Intelligence. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, pp. 664–669.

Rao, A. S., Georgeff, M. P., et al., 1995. Bdi agents: From theory to practice. In: Proceedings of the first international conference on multi-agent systems (ICMAS-95). San Francisco, pp. 312–319.

Ruckle, W., 1981. Ambushing random walks II: Continuous models. Operations Research 29 (1), 108–120.

Ruckle, W., Fennell, R., Holmes, P., Fennemore, C., 1976. Ambushing random walks I: Finite models. Operations Research 24 (2), 314–324.

Russell, S., Norvig, P., Davis, E., Russell, S., Russell, S., 2010. Artificial intelligence: a modern approach. Prentice hall, NJ.

Schrijver, A., 2003. Combinatorial optimization: polyhedra and efficiency. Vol. 24. Springer.

Seow, K., Lee, D., 2010. Performance of multiagent taxi dispatch on extended-runtime taxi availability: A simulation study. Intelligent Transportation Systems, IEEE Transactions on 11 (1), 231–236.

Shoham, Y., Leyton-Brown, K., 2008. Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations. Cambridge University Press.

Sislak, D., Volf, P., Jakob, M., Pechoucek, M., 2009. Distributed Platform for Large-Scale Agent-Based Simulations. LNAI 5920. Springer, Berlin, Ch. 2, pp. 16–32.

Sislak, D., Volf, P., Kopriva, S., Pechoucek, M., 2011. Agentfly: NAS-wide simulation framework integrating algorithms for automated collision avoidance. In: Proceedings of Integrated Communications, Navigation and Surveilance Conference. IEEE, pp. F7–1.

Slootmaker, L., 2011. Countering piracy with the next-generation piracy performance surface model (master thesis). Tech. rep., NPS, Monterey California.

Talbi, E.-G., 2009. Metaheuristics: from design to implementation. Vol. 74. Wiley.

Tambe, M., 2011. Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned. Cambridge University Press.

Tsai, J., Rathi, S., Kiekintveld, C., Ordonez, F., Tambe, M., 2009. IRIS a tool for strategic security allocation in transportation networks. In: AAMAS-09 (Industry Track). pp. 37–44.

Tsai, J., Yin, Z., Kwak, J., Kempe, D., Kiekintveld, C., Tambe, M., 2010a. Strategic Security Placement in Network Domains with Applications to Transit Security.

Tsai, J., Yin, Z., young Kwak, J., Kempe, D., Kiekintveld, C., Tambe, M., 2010b. Urban Security: Game-Theoretic Resource Allocation in Networked Physical Domains. In: AAAI.

Tsilis, T., 2011. Counter-piracy escort operations in the Gulf of Aden (master thesis). Tech. rep., NPS, Monterey California.

Van Dyke Parunak, H., Savit, R., Riolo, R., 1998. Agent-based modeling vs. equation-based modeling: A case study and users' guide. In: Proceedings of the 1998 Workshop on Multi-Agent Systems and Agent-Based Simulation. Springer, pp. 277–283.

Vaněk, O., 2011. Security games with mobile patrollers (extended abstract). In: Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS). Vol. 3. International Foundation for Autonomous Agents and Multiagent Systems, pp. 1371–1372.

Vaněk, O., Jakob, M., Hrstka, O., Pěchouček, M., 2012a. Using multi-agent simulation to improve the security of maritime transit. In: Proceedings of the Multi-Agent-Based Simulation Workshop (MABS). Springer Berlin Heidelberg, pp. 44–58.

Vaněk, O., Yin, Z., Jain, M., Bošanskỳ, B., Tambe, M., Pěchouček, M., 2012b. Game-theoretic resource allocation for malicious packet detection in computer networks. In: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS). Vol. 1. International Foundation for Autonomous Agents and Multiagent Systems, pp. 905–912.

Vaněk, O., 2010. Agent-based simulation of the maritime domain. Acta Polytechnica 50, 94–99.

Vaněk, O., Bošanský, B., Jakob, M., Lisý, V., Pěchouček, M., 2012. Extending security games to defenders with constrained mobility. In: Proceedings of AAAI Spring Symposium, GTSSH.

Vaněk, O., Bošanský, B., Jakob, M., Pěchouček, M., 2010. Transiting areas patrolled by a mobile adversary. In: Proceedings of the Conference on Computational Intelligence and Games (CIG), 2010 IEEE Symposium on. IEEE, pp. 9–16.

Vaněk, O., Hrstka, O., Pěchouček, M., 2013a. Dynamic group transit scheme for corridor transit. In: Proceedings of the International Conference on Modeling, Simulation and Optimization (ICMSAO).

Vaněk, O., Hrstka, O., Pěchouček, M., 2013b. Improving group transit schemes to minimize negative effects of maritime piracy (submitted). Transactions on Intelligent Transportation Systems.

Vaněk, O., Jakob, M., Hrstka, O., Pěchouček, M., 2013c. Agent-based model of maritime traffic in piracy-affected waters (submitted). Transportation Research Part C: Emerging Technologies.

Vaněk, O., Jakob, M., Lisỳ, V., Bošanskỳ, B., Pěchouček, M., 2011. Iterative game-theoretic route selection for hostile area transit and patrolling. In: Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS). pp. 1273–1274.

Vidal, R., Shakernia, O., Kim, H., Shim, D., Sastry, S., 2002. Probabilistic pursuit-evasion games: theory, implementation, and experimental evaluation. Robotics and Automation, IEEE Transactions on 18 (5), 662–669.

Vokrinek, J., Komenda, A., Pechoucek, M., May 2010. Cooperative agent navigation in partially unknown urban environments. In: PCAR '10: The Third International Symposium on Practical Cognitive Agents and Robots. Proceedings of the AAMAS-10 Workshops. IFAAMAS: Internatioal Foundation for Autonomous Agents and Multiagent Systems, Toronto, Canada, pp. 46–53.

Washburn, A., 1981. Search and detection. Military Applications Section, Operations Research Society of America.

Washburn, A., Wood, K., 1995. Two-person Zero-sum Games for Network Interdiction. Operations Research 43 (2), 243–251.

Wierzbicki, A. P., 1980. The use of reference objectives in multiobjective optimization. Springer.

Wilson, D., 2005. Use of modeling and simulation to support airport security. Aerospace and Electronic Systems Magazine, IEEE 20 (8), 3–6.

Yang, Z., Mueller, R., 2008. Unbiased histogram matching quality measure for optimal radiometric normalization. In: Proc. American Society for Photogrammetry and Remote Sensing Annual Conference.

Yin, Z., Korzhyk, D., Kiekintveld, C., Conitzer, V., Tambe, M., 2010. Stackelberg vs. Nash in security games: Interchangeability, equivalence, and uniqueness. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1. International Foundation for Autonomous Agents and Multiagent Systems, pp. 1139–1146.

Yu, P., Chen, M., Kandlur, D., 1993. Grouped sweeping scheduling for dasd-based multimedia storage management. Multimedia Systems 1 (3), 99–109.

Appendix A
Mathematical Programs for Oracles

Defender Oracle: Static Defender — Exact Utility

$$
\begin{aligned}
&\max \sum_{s_a \in S_E} x_a \cdot z_a \\
&z_a = 1 - \prod_{k=0}^{K} (1 - c_l \cdot B_l^a) && \forall a \in S_E \\
&\sum_l c_l \leq K \\
&c_l = \{0, 1\} \qquad z_a = [0, 1]
\end{aligned}
$$

Fig. A.1: Defender Oracle: Static Defender — Exact Utility — Uncertain Encounters

$$
\begin{aligned}
&\max \sum_{s_a \in S_E} x_a \cdot z_a \\
&z_a \leq \sum_l c_l \cdot B_l^a && \forall a = 1 \ldots |S_E| \\
&\sum_l c_l \leq K \\
&c_l = \{0; 1\} \qquad z_a = [0, 1]
\end{aligned}
$$

Fig. A.2: Defender Oracle: Static Defender — Exact Utility — Certain Encounters

Defender Oracle: Fixed-Base Defender — Exact Utility

$$\max \quad \sum_{s_a \in S_E} x_a \cdot \frac{1}{\Omega} \sum_{j=0}^{\Omega-1} 1 - \prod_{l \in s_a} \left(1 - {}^{j}_{a}z_l \cdot \rho_l\right)$$

$$
\begin{aligned}
{}^{j}_{a}z_l \geq {}^{j}c^{i}_{l} + {}_{a}p^{i}_{l} - 1 \qquad {}^{j}_{a}z_l \geq 0 \qquad & \forall a \in E, \quad \forall l \in L, \quad \forall j \\
\sum_{e \in out(b)} {}^{j}c^{1}_{e} = 1 \qquad {}^{j}c^{\Omega}_{b} = 1 \qquad & \forall j \\
\sum_{e \in out(n)} {}^{j}c^{i+2}_{e} = \sum_{e \in in(n)} {}^{j}c^{i}_{e} \qquad & i = 1, \ldots, \Omega - 2, \forall n \in N, \forall j \\
{}^{j}c^{i+1}_{n} = \sum_{e \in in(n)} {}^{j}c^{i}_{e} \qquad & i = 1, \ldots, \Omega - 2, \forall n \in N, \forall j \\
\sum_{l \in L} {}^{j}c^{i}_{l} = 1 \qquad & i = 1, \ldots, \Omega, \forall j \\
{}^{j}c^{i}_{l} = {}^{j+1}c^{i+2}_{l} \qquad & i = 1, \ldots, \Omega, \forall l \in L, \forall j \\
{}^{j}_{a}z_l = \{0, 1\} \qquad & \forall a \in E, \forall l \in L, \forall j \\
{}^{j}c^{i}_{l} = \{0, 1\} \qquad & i = 1, \ldots, \Omega, \forall j
\end{aligned}
$$

Fig. A.3: Defender Oracle: Fixed-Base Defender — Exact Utility — **Uncertain** Encounters

$$\max \quad \sum_{s_a \in S_E} x_a \cdot \frac{1}{\Omega} \sum_{j=0}^{\Omega-1} {}^{j}_{a}\omega$$

$$
\begin{aligned}
M \cdot {}^{j}_{a}\omega - \sum_{l \in s_a} {}^{j}_{a}z_l \geq 0 \qquad & \forall j, \forall a \\
{}^{j}_{a}z_l \geq {}^{j}c^{i}_{l} + {}_{a}p^{i}_{l} - 1 \qquad {}^{j}_{a}z_l \geq 0 \qquad & \forall a \in E, \quad \forall l \in L, \quad \forall j \\
\sum_{e \in out(b)} {}^{j}c^{1}_{e} = 1 \qquad {}^{j}c^{\Omega}_{b} = 1 \qquad & \forall j \\
\sum_{e \in out(n)} {}^{j}c^{i+2}_{e} = \sum_{e \in in(n)} {}^{j}c^{i}_{e} \qquad & i = 1, \ldots, \Omega - 2, \forall n \in N, \forall j \\
{}^{j}c^{i+1}_{n} = \sum_{e \in in(n)} {}^{j}c^{i}_{e} \qquad & i = 1, \ldots, \Omega - 2, \forall n \in N, \forall j \\
\sum_{l \in L} {}^{j}c^{i}_{l} = 1 \qquad & i = 1, \ldots, \Omega, \forall j \\
{}^{j}c^{i}_{l} = {}^{j+1}c^{i+2}_{l} \qquad & i = 1, \ldots, \Omega, \forall l \in L, \forall j \\
{}^{j}_{a}\omega = \{0; 1\} & \\
{}^{j}_{a}z_l = \{0, 1\} \qquad & \forall a \in E, \forall l \in L, \forall j \\
{}^{j}c^{i}_{l} = \{0, 1\} \qquad & i = 1, \ldots, \Omega, \forall j
\end{aligned}
$$

Fig. A.4: Defender Oracle: Fixed-Base Defender — Exact Utility — **Certain** Encounters

Defender Oracle: Unconstrained Defender — Exact Utility

$$\max \quad \sum_{s_a \in S_E} x_a \cdot \frac{1}{\Omega} \sum_{j=0}^{\Omega-1} 1 - \prod_{l \in s_a} \left(1 - {}^{j}_{a}z_l \cdot \rho_l\right)$$

$$
\begin{aligned}
{}^{j}_{a}z_l \geq {}_{j}c^{i}_{l} + {}_{a}p^{i}_{l} - 1 \qquad {}^{j}_{a}z_l \geq 0 && \forall a \in E, l \in L, j \\
{}^{j}c^{i+1}_{n} = \sum_{e \in in(n)} {}^{j}c^{i}_{e} && \forall i = 1, \ldots, \Omega - 2, n \in N, j \\
\sum_{l \in L} {}^{j}c^{i}_{l} = 1 && \forall i = 1, \ldots, \Omega, j \\
{}^{j}_{a}z_l = \{0, 1\} && \forall a \in E, l \in L, j \\
{}^{j}c^{i}_{l} = \{0, 1\} && \forall i = 1, \ldots, \Omega, j
\end{aligned}
$$

Fig. A.5: Defender Oracle: Unconstrained Defender — Exact Utility — **Uncertain** Encounters

$$\max \quad \sum_{s_a \in S_E} x_a \cdot \frac{1}{\Omega} \sum_{j=0}^{\Omega-1} {}^{j}_{a}\omega$$

$$
\begin{aligned}
M \cdot {}^{j}_{a}\omega - \sum_{l \in s_a} {}^{j}_{a}z_l \geq 0 && \forall j, \forall a \\
{}^{j}_{a}z_l \geq {}^{j}c^{i}_{l} + {}_{a}p^{i}_{l} - 1 \qquad {}^{j}_{a}z_l \geq 0 && \forall a \in E, \quad \forall l \in L, \quad \forall j \\
{}^{j}c^{i+1}_{n} = \sum_{e \in in(n)} {}^{j}c^{i}_{e} && \forall i = 1, \ldots, \Omega - 2, n \in N, \forall j \\
\sum_{l \in L} {}^{j}c^{i}_{l} = 1 && i = 1, \ldots, \Omega, \forall j \\
{}^{j}c^{i}_{l} = {}^{j+1}c^{i+2}_{l} && i = 1, \ldots, \Omega, \forall l \in L, \forall j \\
{}^{j}_{a}\omega = \{0; 1\} && \\
{}^{j}_{a}z_l = \{0, 1\} && \forall a \in E, \forall l \in L, \forall j \\
{}^{j}c^{i}_{l} = \{0, 1\} && i = 1, \ldots, \Omega, \forall j
\end{aligned}
$$

Fig. A.6: Defender Oracle: Unconstrained Defender — Exact Utility — **Certain** Encounters

Defender Oracle: Approximate Utility

$$\max \quad \sum_{s_a \in S_E} x_a \cdot \sum_{l \in s_a} c_l \cdot \rho_l$$
$$\sum_{l \in L} c_l \leq K$$
$$c_l = \{0, 1\}$$

Fig. A.7: Defender Oracle: Static Defender — Approximate Utility

$$\max \quad \sum_{s_a \in S_E} x_a \cdot \frac{1}{\Omega} \sum_{j=0}^{|\Omega|-1} \sum_{l \in s_a} \sum_{i=0}^{|s_a|-1} {}^{j}c_l^i \cdot {}_{a}p_l^i \cdot \rho_l$$
$${}^{j}c_l^i = \{0, 1\}$$
$$\sum_{e \in out(b)} {}^{j}c_e^1 = 1 \qquad \forall j$$
$${}^{j}c_b^{\Omega} = 1 \qquad \forall j$$
$$\sum_{e \in out(n)} {}^{j}c_e^{i+2} = \sum_{e \in in(n)} {}^{j}c_e^i \qquad i = 1, \ldots, \Omega - 2, \forall n \in N, \forall j$$
$${}^{j}c_n^{i+1} = \sum_{e \in in(n)} {}^{j}c_e^i \qquad i = 1, \ldots, \Omega - 2, \forall n \in N, \forall j$$
$$\sum_{l \in L} {}^{j}c_l^i = 1 \qquad i = 1, \ldots, \Omega, \forall j$$
$${}^{j}c_l^i = {}^{j+1}c_l^{i+2} \qquad {}^{j}c_l^i = \{0, 1\} \qquad i = 1, \ldots, \Omega, \forall l \in L, \forall j$$
$${}^{j}_{a}z_l = \{0, 1\} \qquad \forall a \in E, \forall l \in L, \forall j$$

Fig. A.8: Defender Oracle: Fixed-Base Defender — Approximate Utility

$$\max \quad \sum_{s_a \in S_E} x_a \cdot \frac{1}{\Omega} \sum_{j=0}^{|\Omega|-1} \sum_{l \in s_a} \sum_{i=0}^{|s_a|-1} {}^{j}c_l^i \cdot {}_{a}p_l^i \cdot \rho_l$$
$${}^{j}c_n^{i+1} = \sum_{e \in in(n)} {}^{j}c_e^i \qquad \forall i = 1, \ldots, \Omega - 2, n \in N, j$$
$$\sum_{l \in L} {}^{j}c_l^i = 1 \qquad {}^{j}c_l^i = \{0, 1\} \qquad \forall i = 1, \ldots, \Omega, j$$
$${}^{j}_{a}z_l = \{0, 1\} \qquad \forall a \in E, l \in L, j$$

Fig. A.9: Defender Oracle: Unconstrained Defender — Approximate Utility

Evader Oracle: Static Defender — Exact Utility

$$
\begin{aligned}
&\min \sum_{a \in S_P} x_a \cdot z_a \\
z_a &= \prod_{l \in P} (1 - c_l \cdot B_l^a \cdot \rho_l) && \forall s_a \in S_P \\
\sum_{o \in \mathcal{O}} \sum_{l \in out(o)} c_l &= 1 \\
\sum_{d \in \mathcal{D}} \sum_{l \in in(d)} c_l &= 1 \\
\sum_{e \in in(v)} c_e &= \sum_{e \in out(n)} c_e && \forall n \in N \\
c_n &\geq \sum_{e \in in(n)} c_e && \forall n \in N
\end{aligned}
$$

Fig. A.10: Evader Oracle: Static Defender — Exact Utility — **Uncertain** Encounters

$$
\begin{aligned}
&\min \sum_{a \in S_P} x_a \cdot z_a \\
z_a &\geq c_l + B_l^a - 1 \\
z_a &\geq 0 \\
z_a &= \{0, 1\} \\
\sum_{o \in \mathcal{O}} \sum_{l \in out(o)} c_l &= 1 \\
\sum_{d \in \mathcal{D}} \sum_{l \in in(d)} c_l &= 1 \\
\sum_{e \in in(v)} c_e &= \sum_{e \in out(n)} c_e && \forall n \in N \\
c_n &\geq \sum_{e \in in(n)} c_e && \forall n \in N
\end{aligned}
$$

Fig. A.11: Evader Oracle: Static Defender — Exact Utility — **Certain** Encounters

Evader Oracle: Fixed-Base, Unconstrained Defender — Exact Utility

$$\min \quad \sum_{s_a \in S_P} x_a \cdot \frac{1}{|s_a|} \sum_{j=0}^{|s_a|-1} 1 - \prod_{l \in s_a} \left(1 - {}^{j}_{a}z_l \cdot \rho_l\right)$$

$$ {}^{j}_{a}z_l \geq c^i_l + {}^{j}_{a}p^i_l - 1 \qquad {}^{j}_{a}z_l \geq 0 \qquad \forall a \in E, \forall l \in L, \forall j$$

$$\sum_{o \in \mathcal{O}} \sum_{e \in out(o)} c^1_e = 1$$

$$\sum_{d \in \mathcal{D}} \sum_{e \in in(d)} \sum_{i=2}^{K} c^i_e = 1$$

$$\sum_{e \in out(n)} c^{i+2}_e = \sum_{e \in in(n)} c^i_e \qquad i = 1, \ldots, K-2, \forall n \in N$$

$$c^{i+1}_n = \sum_{e \in in(n)} c^i_e \qquad \forall n \in N, i = 1, \ldots, K-2$$

$${}^{j}_{a}z_l = \{0, 1\} \qquad \forall a \in E, \forall l \in L, \forall j$$

$$c^i_l = \{0, 1\} \qquad i = 1, \ldots, \Omega, \forall l \in L$$

Fig. A.12: Evader Oracle: Fixed-Base, Unconstrained Defender — Exact Utility — **Uncertain Encounters**

$$\min \quad \sum_{s_a \in S_P} x_a \cdot \frac{1}{|s_a|} \sum_{j=0}^{|s_a|-1} {}^{j}_{a}\omega$$

$$M \cdot {}^{j}_{a}\omega - \sum_{l \in s_a} {}^{j}_{a}z_l \geq 0 \qquad \forall j, \forall a$$

$$ {}^{j}_{a}z_l \geq c^i_l + {}^{j}_{a}p^i_l - 1 \qquad {}^{j}_{a}z_l \geq 0 \qquad \forall a \in E, \forall l \in L, \forall j$$

$$\sum_{o \in \mathcal{O}} \sum_{e \in out(o)} c^1_e = 1$$

$$\sum_{d \in \mathcal{D}} \sum_{e \in in(d)} \sum_{i=2}^{K} c^i_e = 1$$

$$\sum_{e \in out(n)} c^{i+2}_e = \sum_{e \in in(n)} c^i_e \qquad i = 1, \ldots, K-2, \forall n \in N$$

$$c^{i+1}_n = \sum_{e \in in(n)} c^i_e \qquad \forall n \in N, i = 1, \ldots, K-2$$

$${}^{j}_{a}\omega = \{0; 1\}$$

$${}^{j}_{a}z_l = \{0, 1\} \qquad \forall a \in E, \forall l \in L, \forall j$$

$$c^i_l = \{0, 1\} \qquad i = 1, \ldots, \Omega, \forall l \in L$$

Fig. A.13: Evader Oracle: Fixed-base, Unconstrained Defender — Exact Utility — **Certain Encounters**

Evader Oracle: Approximate Utility

$$\min \sum_{s_a \in S_P} x_a \sum_{l \in s_a} c_l \cdot \rho_l$$
$$c_l = \{0, 1\}$$

Fig. A.14: Evader Oracle: Static Defender — Approximate Utility

$$\min \sum_{s_a \in S_P} x_a \cdot \frac{1}{|s_a|} \sum_{j=0}^{|s_a|-1} \sum_{l \in L} \sum_{i=0}^{K-1} c_l^i \cdot {}_a^j p_l^i \cdot \rho_l$$
$$\sum_{o \in \mathcal{O}} \sum_{e \in out(o)} c_e^1 = 1$$
$$\sum_{d \in \mathcal{D}} \sum_{e \in in(d)} \sum_{i=2}^{K} c_e^i = 1$$
$$\sum_{e \in out(n)} c_e^{i+2} = \sum_{e \in in(n)} c_e^i \qquad i = 1, \ldots, K-2, \forall n \in N$$
$$c_n^{i+1} = \sum_{e \in in(n)} c_e^i \qquad \forall n \in N, i = 1, \ldots, K-2$$
$${}_a^j z_l = \{0, 1\} \qquad \forall a \in E, \forall l \in L, \forall j$$
$$c_l^i = \{0, 1\} \qquad i = 1, \ldots, \Omega, \forall l \in L$$

Fig. A.15: Evader Oracle: Fixed-Base, Unconstrained Defender — Approximate Utility

Appendix B
Computed Strategies for Merchant Vessel and Pirate in the Simulation-based Evaluation

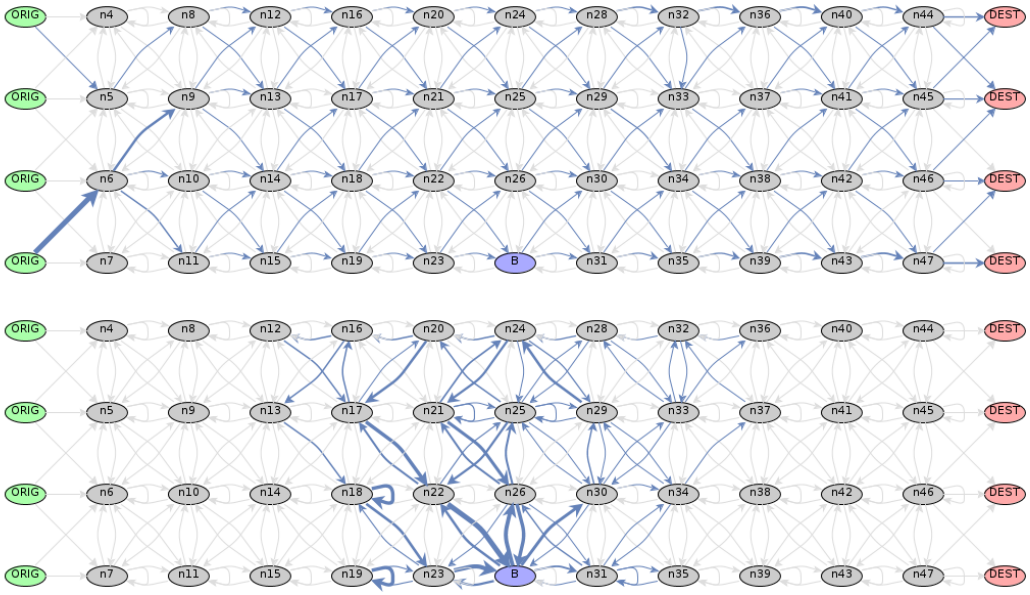


Fig. B.1: Graph of width $w = 4$ with computed Merchant Vessel and Pirate strategies used in the experiments.

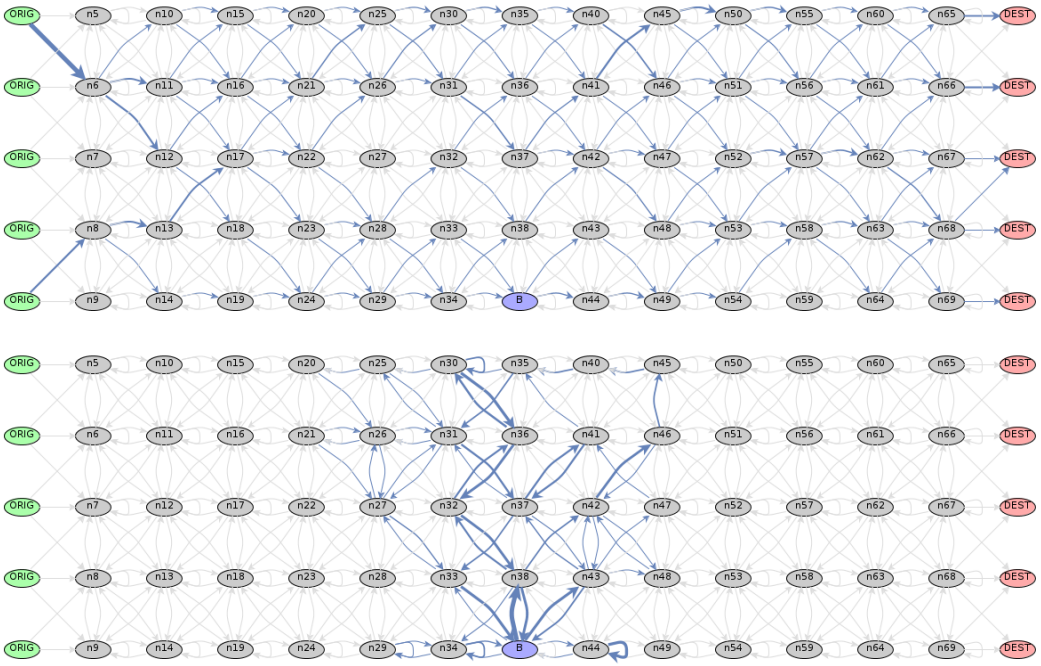


Fig. B.2: Graph of width $w = 5$ with computed Merchant Vessel and Pirate strategies used in the experiments.

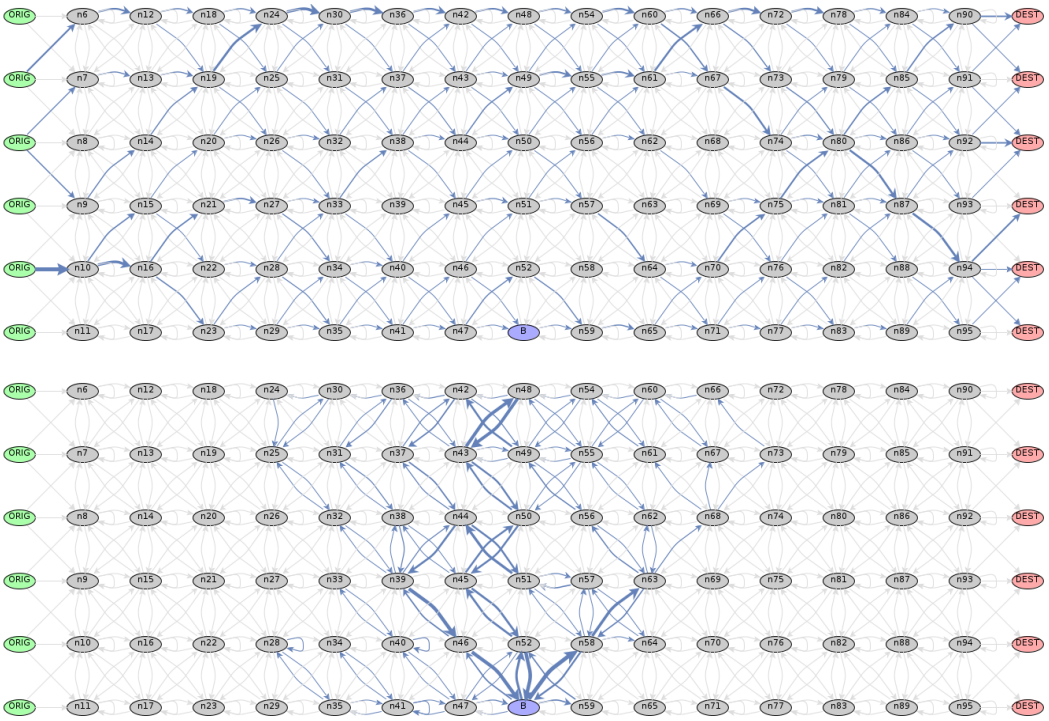


Fig. B.3: Graph of width $w = 6$ with computed Merchant Vessel and Pirate strategies used in the experiments.

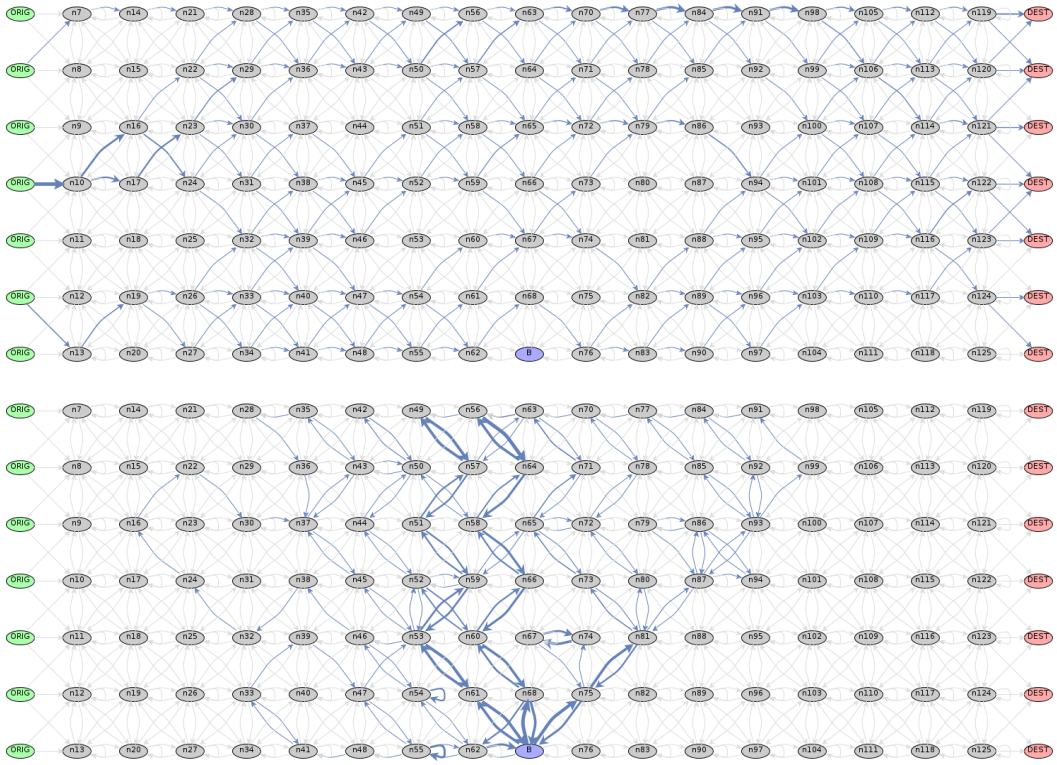


Fig. B.4: Graph of width $w = 7$ with computed Merchant Vessel and Pirate strategies used in the experiments.

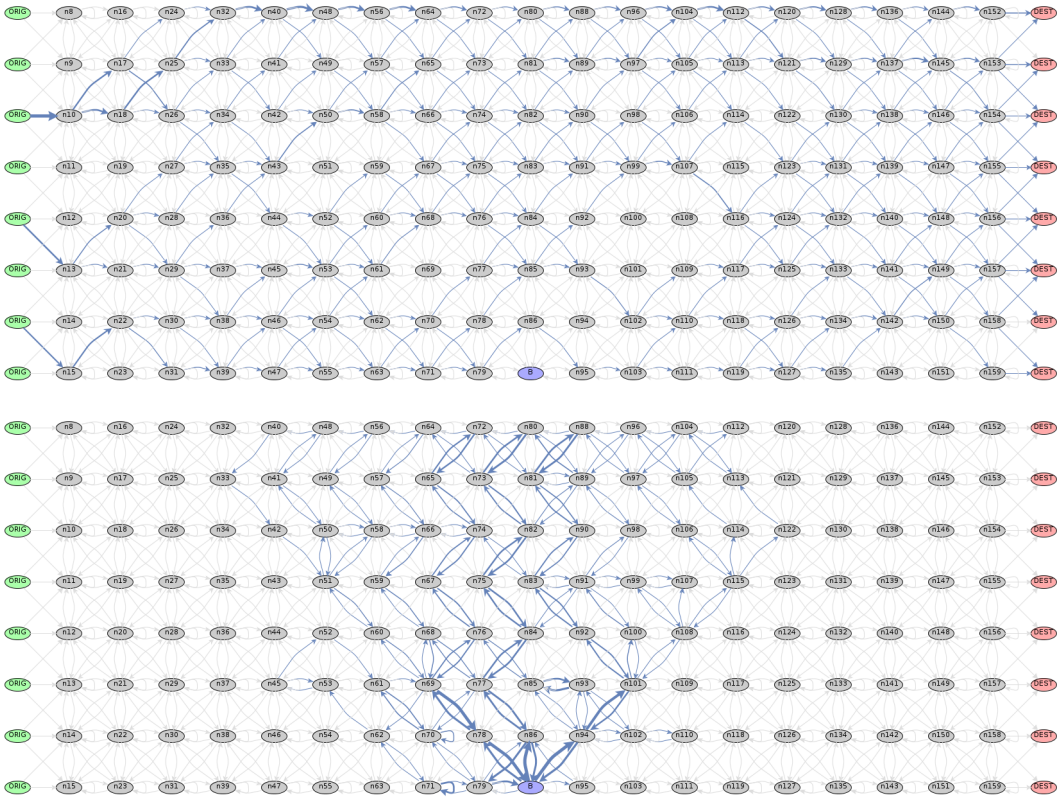


Fig. B.5: Graph of width $w = 8$ with computed Merchant Vessel and Pirate strategies used in the experiments.