Erasmus University Rotterdam

Master Thesis
Operations Research and Quantitative Logistics

Allocation of maritime assets in surveillance missions

Author:
B.B. van Genderen
Studentnumber 323963

Supervisors:
Erasmus University Rotterdam
Dr. D. Huisman
TNO
Ir. J.F.J. Vermeulen

november 2013

Abstract

This thesis presents a way to optimize the allocation of surveillance units in an anti piracy mission. Because of the many similarities with search theory we have used the yet available research in this field to solve this problem. We assume that the total area to be surveyed consists of a finite set of non-overlapping subareas to which we can assign our surveillance units. We assume that the time during which we have to allocate our units consists of a finite set of periods with equal length. In these periods each of our surveillance units can be assigned to exactly one area. We introduce a measure indicates the risk (of pirate attacks) in a certain area. With respect to this value we try to optimize the allocation of our units, taking transit times between areas into account. In this thesis we use CPLEX to solve a mixed-integer linear formulation. We also consider two metaheuristics, namely cross-entropy and simulated annealing. The methods were tested on a variety of problems. CPLEX was able to solve most of these problems, except the larger ones. Simulated annealing finds near optimal solutions and is able to find a solution for all cases. Cross-entropy finds near optimal solutions for the small cases, but when increasing the number of periods it is unable to still find good solutions.

Contents

1 Introduction **3**

2 Problem Description **4**

3 Literature Survey **8**

4 Mathematical Formulation **11**
4.1 Network Formulation - NMSP 11
4.2 Network Formulation - MSP 14
4.3 Linear Formulation 14
4.3.1 Identical Searcher NMSP 15
4.3.2 Identical Searcher MSP 16
4.3.3 Different Searcher NMSP 18
4.3.4 Different Searcher MSP 19

5 Solving NMSP and MSP **21**
5.1 Network Formulation 21
5.1.1 Cross-Entropy Method 21
5.1.2 Simulated annealing 23
5.2 Linear Formulation 25

6 Results **26**
6.1 Setting the parameters 27
6.2 Identical Searcher 28
6.2.1 NMSP 28
6.2.2 MSP 29
6.3 Different Searchers 30
6.3.1 NMSP 30
6.3.2 MSP 30

7 Conclusion and Recommendations **32**

8 Literature **34**

9 Appendix **35**
9.1 Appendix A - List of Definitions 35
9.2 Appendix B - Variations 36
9.2.1 Cells 36
9.2.2 Risk reduction 36

1 Introduction

Maritime surveillance is surveillance of the sea or other waters. This type of surveillance is like any other type, often done to prevent illegal activities. On the water common examples of such activities are smuggling, illegal fishing and piracy. This thesis focuses on the latter one.

When talking about pirates most people think about tough men with eye patches and peg legs that lived centuries ago, however during the last decades piracy has again become a serious problem. Modern piracy mainly takes place near the coast of Somalia, in the Indian Ocean, the Gulf of Oman and in several other places. The targets of pirates are often cargo ships. These ships continuously pass through the area infected with pirates. The pirates use small high speed boats to approach the ships. Their goal is to hijack these cargo ships and demand a ransom. In a report by the world bank it was estimated that piracy costs the global economy up to 18 billion US dollars each year. Apart from the negative economic effect, piracy also endangers the lives of the crew on the cargo ships.

To encounter pirates and prevent attacks, several countries have located maritime units in the piracy zones. These units survey the area and deal with any pirates encountered. The effectiveness of these maritime units partly depends on the location in which they are active. Therefore this location should be chosen wisely. To maximize the effect of our maritime units we should optimize the allocation. This optimization will be the focus of this thesis. We will develop a model which is able to allocate maritime units such that the effect of their units is maximized.

Our goal however, is not to maximize the amount of pirates that we catch. We only want to catch pirates when they are actually able to attack ships, or preferably pirates that are near victims to prevent attacks. We start with a definition of the problem in sections 2, then we continue with a survey of the relevant literature in Section 3. After discussing the literature we give a mathematical formulation of the problem in Section 4. The methods that we will use to solve the problem are explained in Section 5. The results and the conclusion are given in section 6 and 7.

2 Problem Description

In this thesis we investigate the allocation of maritime units in maritime surveillance missions. In these missions the navy is often operating in a large area. We will refer to this area as the Area of Operations (AOO). An example of an AOO is given in Figure 1. There are different causes for problems in the AOO. The main cause of these problems is the presence of pirates. These pirates frequently attack so called *white vessels* that pass through the area. Other causes of trouble are for example illegal fishing and smuggling. By doing maritime surveillance they navy wants to prevent the malicious activities that take place inside their AOO. In this thesis we will focus on piracy, however other factors such as illegal fishing and smuggling could be incorporated in the model as well.

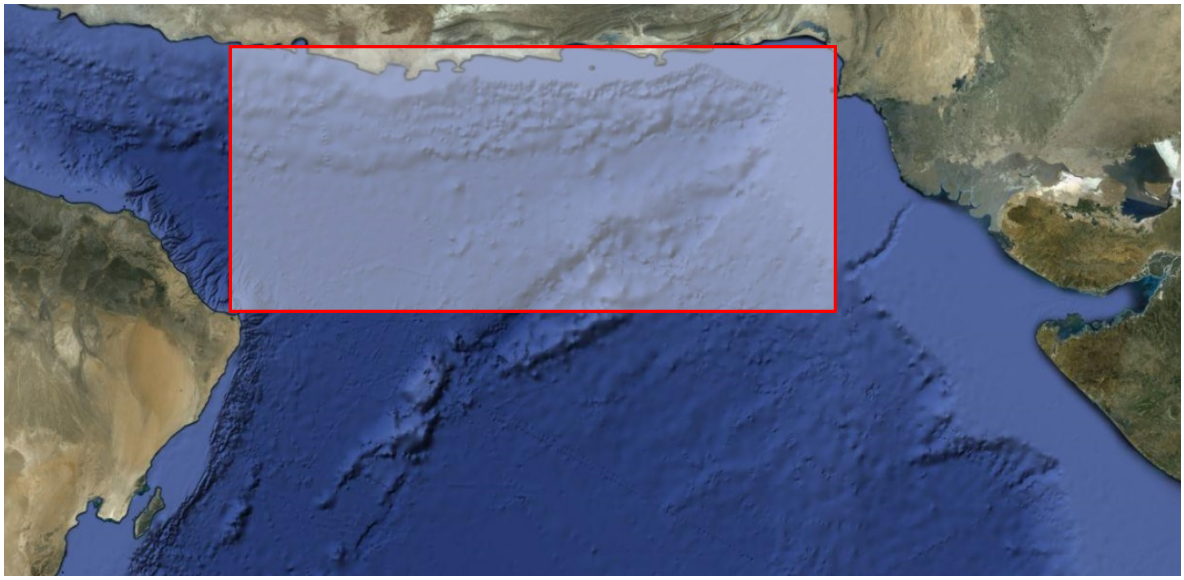

Figure 1: The Gulf of Oman with an example of the AOO

To protect the AOO, several maritime units or *assets* are available. Examples of such assets are Frigates, UAVs and Helicopters. These assets do not work individually but are grouped in so called *task groups*. The compositions of the task groups are fixed, however different task groups may have different capabilities. Some task groups could contain more assets and are therefore more effective in protecting the sea, while some task groups may be faster and are therefore able to travel to other areas in a shorter time. The totality of all task groups forms the *task force*.

We divide the total AOO into smaller areas, to which we will refer as cells, as can be seen in Figure 2. Furthermore we divide the total period for which we want to make a planning for our task groups into periods of equal length. In each period, each task group will be assigned to exactly one cell. This is also called the Area of Responsibility (AOR) of a task group. The task groups may not operate in the same cell during the same period. Task groups are allowed to stay in a cell during consecutive periods. If a task group is assigned to a different cell in the next period a transit time must be taken into account. This transit time is equal to the distance between the centers of the two areas divided by the average speed of the task group. It is impossible to consecutively visit two areas with a transit time that exceeds the length of

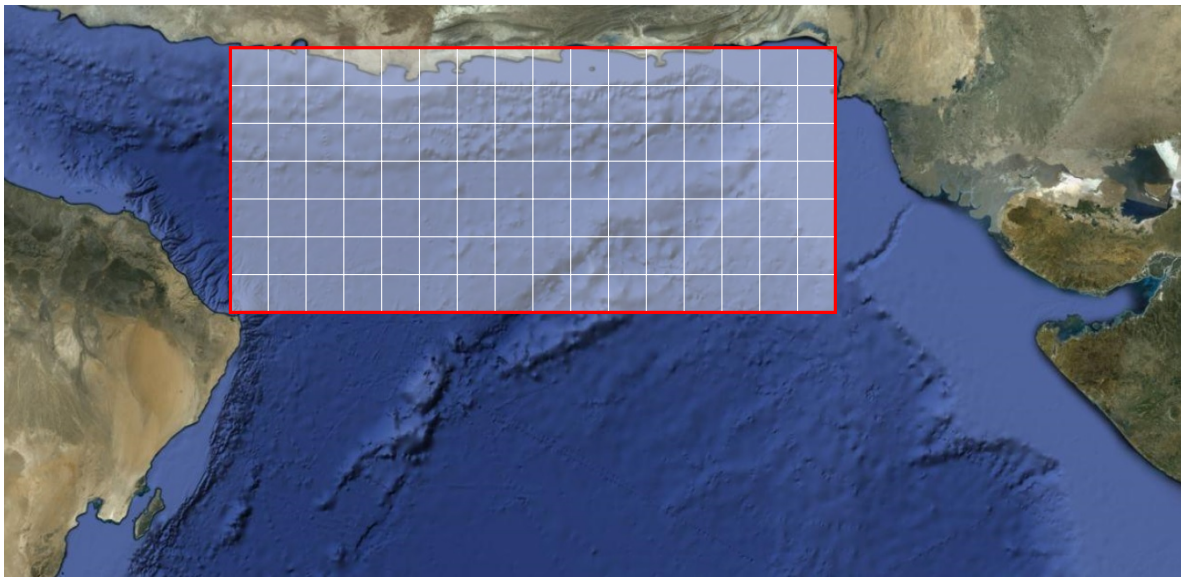

Figure 2: A cell structure over the Area of Operations

one period. The transit possibilities for a task group are visually explained in Figure 3. A certain task group is positioned in the cell with the red dot. Its average speed per period is equal to two times the length of a cell. The green circle, which has a radius of 2 cell lengths, represents the area that can be reached in the following period. All possible locations for the next period are given by the black dots.

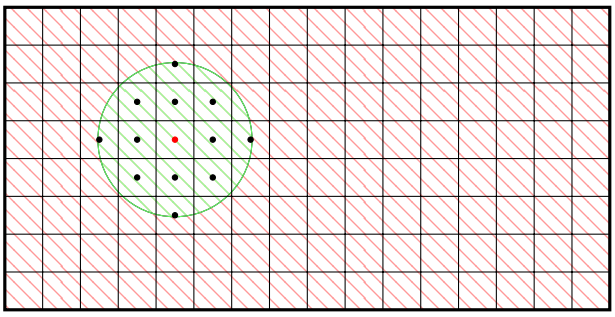

Figure 3: Example of task groups transit possibilities

To make an optimal assignment of task groups to cells we need to define a measure that tells us what benefit we can gain from assigning task groups to cells. Therefore we will first clarify the situation inside the areas. A cell may contain several pirates. These pirates move regularly to different places or back to the coast. We will however not model the behavior of these pirates. Instead we assume that there is a so-called *threat map* of the whole AOO, indicating the average amount of pirate activity in each cell within the AOO. This map indicates where the pirates are often located. As stated before, pirates are not of any interest as long as there is no white vessel in their neighbourhood. Therefore we will also include information about the white vessels in our data. The movements of white vessels through

our AOO are assumed to be known. The two sources of information are combined into the so-called *risk map*. The risk map holds a value for each cell in the AOO, for each period. This value is high if a cell has both high pirate activity and a lot of white vessels passing through this cell. The risk value in a cell will be referred to as the *Risk Density* (RD) of a cell. The value of the threat map is assumed to be the same during the whole period, however the movement of white vessels changes each period. Therefore the RD in a cell can also change every period.

The RD can be seen as the risk in a cell that we can prevent - or the profit that we can gain - by visiting this cell. Note that we cannot always prevent the full risk, thus not gain the full value of the RD when assigning a task group to a cell. This is explained by the fact that we cannot perfectly protect some cell, even if we assign a task group to it. We will introduce a measure of effectiveness (MOE) that indicates how good we prevent the risk in some cell, or which percentage of the RD we will gain when visiting some cell. This measure depends on the capabilities of a task group such as speed. It will also depend on the time that the task group is spending in this cell during the period. The time that a Task Group can spend in the cell is equal to the length of the period minus the transit time from the previous location to this cell.

We will now differentiate between two versions of the problem. In the first version we assume that the pirate activity does not change after a task group has visited a cell. Visiting a cell only reduces the risk in the current period. We will refer to this problem as the Non-reactive Multiple Searcher Problem (NMSP). In the second variation we assume that a task group also neutralizes the pirates in some area. Therefore visiting a cell no longer only reduces the risk in the current period, but also in the following period. In this thesis we will assume that if a task group visits a cell in period t, the risk in that cell in period $t + 1$ will be completely reduced. In period $t + 2$ the risk in this cell will be normal again. We will refer to this problem as the Multiple Searcher Problem (MSP). Example 2.1 clarifies the idea of the MSP.

Example 2.1 *Consider an AOO which is divided into three cells, a time horizon of three periods and a single task group. Figure 4 represents this situation. The left column shows the initial RD values in the different cells for all three periods. This value does not take into account which cells we are going to visit. We will assume that the MOE of our task group is equal to 1 multiplied by the time that a task group spends in a cell. E.g. if a task group spends half a period in a cell the risk in this cell is reduced with* 50% *during this period. Furthermore we will assume that our task group takes half a period to travel from a cell to the neighbour cell. Our task group starts in the left cell.*

Say we choose to stay in the left cell at $t = 1$*. Since we do not have to travel to reach this cell we can spend the total period in this cell, resulting in a MOE of 1. This means we reduce the RD at* $t = 1$ *in the left cell by 100%. We also reduce the RD in the left cell at* $t = 2$ *with* 100%*. In terms of the objective function we gain* $10 + 20 = 30$*. Say we move to the central cell at* $t = 2$*. Now we first need to travel to this cell, which will cost us half a period. Our MOE is equal to 0.5 so we reduce* 50% *of the risk in this cell in the second period. We also reduce the RD in the following period with* 100%*. The objective value is increased with* $0.5 \times 40 + 20 = 40$*. This process continues until we have chosen a location for all periods.*

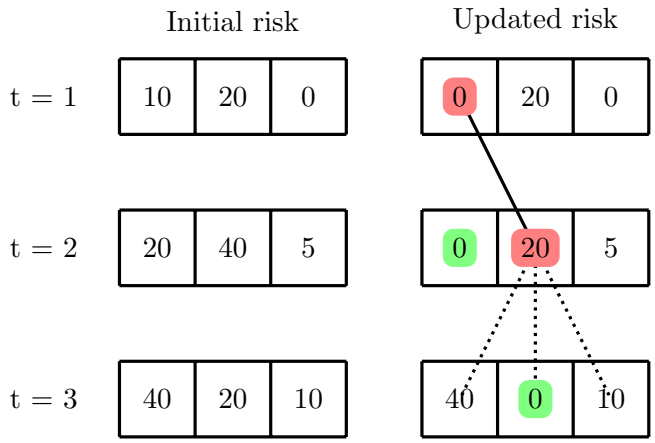

Figure 4: RD reduction example

We have made various assumptions in the problem description. In appendix B we present some variations on the assumptions that we made. These variations might be useful when facing a real surveillance mission or when facing other types of surveillance missions.

3 Literature Survey

The problem that we are facing in this thesis is closely related to search theory. Therefore a brief overview of the important literature in this field is given in this section. Search theory has been studied for several decades and finds its origin in the second world war. In a typical problem of search theory, one or more searchers and one or more targets are considered. The searcher searches the area, in which the target is assumed to be, for a certain amount of time. This area is simplified to a grid with a finite number of cells, similar as in Figure 2. Furthermore the time is divided into discrete time periods. In each period the searcher searches in one of the cells in the area. A distinction can be made between search for static and dynamic targets. In the static case, the target remains in the same position, and in the dynamic case the target moves through the AOO according to a probability distribution. In most related papers the target is assumed to move according to some probability distribution. There are various paths which the target can take with given probabilities. A path is a sequence of cells that are visited by the target. The target is assumed to be in exactly one cell in each period. Since the possible paths and their probabilities are known we can calculate for each period the probability that the target is in a certain cell, or its location probability. The searcher wants to maximize the probability of detecting the target. Therefore we try to find a path for the searcher with the highest cumulative detection probability. Note that both the searcher and the target stay in exactly one cell per time period.

If one of the searchers visits a certain cell and he does not observe the target it becomes less likely that the target has taken any path in which he would visit that cell during that period. This conditionality is also incorporated in most of the research in search theory.

The sketched scenario for a typical search theory problem has many similarities with our problem. We also divide our area into a finite number of cells, and our total period into smaller periods of equal length. We assign each of our task groups - searchers - to exactly one cell per period. Furthermore we try to maximize some objective value which depends on the cells we choose to visit. This objective value is no probability function, describing the location probability of our target(s), but a risk value. Similar to the searcher problem the risk values are known from the beginning. As explained in the problem description and in example 2.1, the risk value in a cell also depends on the earlier visits of cells in the MSP, which is similar to the conditional location probability. Therefore the form of the problem is similar to the searcher problem.

Eagle (1984) faced the searcher problem which is described above for a single searcher and a single target. Both the searcher and the target are restricted to move only to adjacent cells in consecutive periods. Eagle ignored the conditional location probability. He presented a dynamic programming approach which is able to solve a 9 cell, 10 period problem to optimality. Eagle and Yee (1990) faced the same problem again but included the conditional location probability. They extended a branch-and-bound method which was initially presented by Stewart (1979). The bounds are obtained by solving a relaxed nonlinear formulation of the problem. Again optimal solutions were realized for small problem instances and the computation time was reduced.

Martins (1993) also investigates this problem. He solved the problem, using a branch-and-bound method just like Eagle and Yee, however instead of relaxing the constraints in the problem, he relaxed the objective function. Instead of maximizing the probability of detection he maximized the expected number of detections. The expected number of detections is an upper bound on the probability of detecting a target. The computation speed of the branch-

and-bound procedure was improved.

More complex search theory focuses on finding a single target with multiple searchers. Dell et al. (1996) investigated this problem for up to three searchers. They made the same assumptions with respect to the movement of the target and the searchers as was done by Eagle and Yee (1990). Dell et al. formulated the problem as a directed graph, where the nodes represent the locations of all the searchers. By defining the nodes as the location of all searchers, the problem can easily be applied to both single- and multiple searcher problems. Several heuristics were compared in solving this problem. Dell used Local search, a heuristic presented by Martins (1993) as well as an improved version of this heuristic, a Genetic Algorithm, a Genetic Algorithm which uses the results from the other heuristics and a moving horizon heuristic. Last but not least the branch-and-bound method by Eagle and Yee was extended for multiple searchers and used as a benchmark. For larger instances of the problem only the heuristic by Martins and the Genetic algorithm were able to find solutions within a reasonable time. This research showed that the branch and bound method which produces optimal solutions for smaller instances is not effective for solving large problem instances.

Sato (2008) continued the research for the multiple searcher problem, single target problem. He used a new branch and bound method and two different heuristics to solve the problem. The branch and bound method was not sufficient to solve larger instances (more than two searchers) of the problem, but it obtained optimal solutions for smaller instances. The first heuristic, based on the heuristic by Martins (1993), was able to find high quality solutions in a short time, however it was unable to solve large problems. The second heuristic, a cross-entropy method (CE), finds near optimal solutions in a relatively short time. The cross-entropy method, introduced by Rubinstein (1997,1999), is a metaheuristic that randomly generates paths for the searchers. The paths are made by selecting a move from one cell to another with a certain probability until the path is complete. The probability of selecting a move depends on the quality of earlier solutions using this move. The method keeps record of the M_{elite} best paths found so far, called the elite sample. If many paths in the elite sample use a certain move, the chance is high, if few or none of these solutions contains the move, it will be low.

Sato and Royset (2010) present a new way to view the multiple searcher problem. In this research the problem was extended and transit times between cells were included. The problem was formulated as a mixed-integer nonlinear problem for the multiple searcher, multiple target problem. They also presented a linear formulation for the multiple searcher, single target problem. This formulation however requires all searchers to be identical. Several cutting plane techniques as well as some other solvers, among which CPLEX, were applied. The developed cutting plane techniques were able to solve large problems to (near) optimality. The CPLEX solver performed best in situations with a small amount of searchers.

In our research we will consider two different scenarios. The first scenario, NMSP, is similar to a Multiple Searcher Single Target problem where the conditional location probabilities are ignored. The second scenario, MSP, is similar to a common Multiple Searcher Single Target problem where the conditional location probabilities are included. Different from earlier research we allow searchers to travel between non adjacent cells. Furthermore we include a transit time between cells, which has only been done by Sato and Royset (2010).

In our research we will apply several techniques to solve the problem. We will use the cross-entropy method as is done by Sato (2008). Furthermore we will also use simulated annealing (SA) by Kirkpatrick et al. (1983), which has never been applied to a (multiple) searcher

problem. Last but not least we will introduce a new linear formulation of the problem, which is an extension of the linear formulation in Sato and Royset (2010). In contrary to their linear formulation our formulation does not require the searchers to be identical.

4 Mathematical Formulation

In this section we introduce two different mathematical formulations for the problem that we have described in section 2. The first one is a network formulation with nodes and weighted directed arcs. The second formulation is a linear formulation of the problem. We start with the network formulation for the NMSP, which is based on the Multiple Searcher Problem by Sato (2008). All mathematical notations that are used in this formulation are presented below.

4.1 Network Formulation - NMSP

Sets

$\mathcal{C} = \{1, \ldots, C\}$	set with all cells in the AOO
$\mathcal{J} = \{1, \ldots, J\}$	set with all task groups
$\mathcal{T} = \{1, \ldots, T\}$	set with all time periods

Graph

$\mathcal{G} = (\mathcal{N}, \mathcal{A})$	a weighted directed acyclic graph
$\mathcal{N} \subseteq \mathcal{C}^J \times \mathcal{T}$	set of all nodes, where $\mathcal{C}^J$ is a J-dimensional set of cells
$\mathcal{A} \subseteq \mathcal{N} \times \mathcal{N}$	set of all arcs
$\mathcal{F}(n) \subseteq \mathcal{N}$	set of all nodes that can be reached from node $n \in \mathcal{N}$

Parameters

$d_{j,c,c'}$	time in periods between cells $c \in \mathcal{C}$ and $c' \in \mathcal{C}$ for task group $j \in \mathcal{J}$
$r_{c,t}$	risk density in cell $c \in \mathcal{C}$ at time $t \in \mathcal{T}$
n_j	the cell belonging to task group $j \in \mathcal{J}$ in node $n \in \mathcal{N}$
$t(n)$	the time period belonging to node $n \in \mathcal{N}$
$w(n, n')$	the weight on the arc between node $n \in \mathcal{N}$ and $n' \in \mathcal{N}$
α_j	the effectiveness of task group $j \in \mathcal{J}$

Variables

$\mathcal{P}_t$	a path through the graph $\mathcal{G}$ from period 0 to period $t \in \mathcal{T}$

Function

$g_j(c, c')$	reduction for task group $j \in \mathcal{J}$ in cell $c \in \mathcal{C}$ from cell $c' \in \mathcal{C}$
$z(\mathcal{P}_t)$	the objective function for a path $\mathcal{P}_t$

The set of all task groups is represented by $\mathcal{J}$. These task groups are available for surveying the total AOO. The AOO consists of a finite number of cells represented by the set $\mathcal{C}$. The total planning period consists of a finite number of periods, represented by the set $\mathcal{T}$. We introduce the parameter $d_{j,c,c'}$, the time in periods that task group $j \in \mathcal{J}$ requires to go from the center of cell $c \in \mathcal{C}$ to the center of cell $c' \in \mathcal{C}$. The risk density in cell c in period t is represented by $r_{c,t}$. We introduce a function $g_j(c, c')$ as can be seen in equation (1), which indicates the percentage of the risk we prevent if task group $j \in \mathcal{J}$ visits cell c' from cell c. This function may take any form, however we have decided to simply define it as the effective time that we spend in the cell times the effectiveness, α_j, of task group j. The time we spend

in cell c' if we visit it after we visited cell c is equal to 1 minus the travel time between cell c and cell c' in periods.

$$g_j(c, c') = (1 - d_{j,c,c'})\alpha_j \tag{1}$$

We model the problem with a weighted directed acyclic graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$. The nodes $n \in \mathcal{N}$ represent the location of the task groups at some point in time. Therefore $\mathcal{N} \subset (\mathcal{C}^J \times \mathcal{T})$. The arcs $(n, n') \in \mathcal{A}$ represent the movement from the cells belonging to n at time t to the cells belonging to n' in period $t + 1$. In other words the arcs represent the movement of the task groups. Note that an arc is only defined if the transition from one location to another is possible within one period for all task groups. The nodes that can be reached from a certain node n are given by $\mathcal{F}(n)$. In the graph we also include a beginning node and a terminal node, representing the location of the task group before and after the planning period. We introduce the notation n_j representing the cell to which the j^{th} task group is assigned to if node n is chosen. Furthermore we introduce $t(n)$ representing the time corresponding to node n.

If we would consider an AOO with 3 cells and 2 task groups there would be several different ways to allocate our task groups. The different possibilities, six in total, are shown in Figure 5. We have also created the network that belongs to this situation if we would consider 5 consecutive periods. This network is shown in Figure 6. In this network all nodes within a certain period can reach all nodes in a consecutive period. This will not always be the case. If we would only allow the task groups to travel a distance of 1 cell length in a single period, a task group could not travel from cell 1 to cell 3 at once. As result the node that corresponds to situation 1 in Figure 5 would not be connected to the node that corresponds to situation 5 or 6.

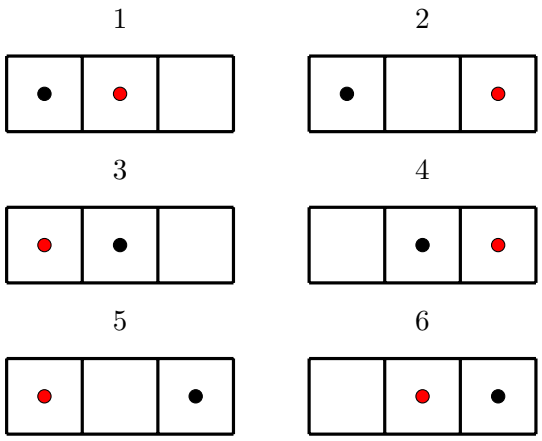

Figure 5: Example of all the different situations for a three-cell, two searcher situation

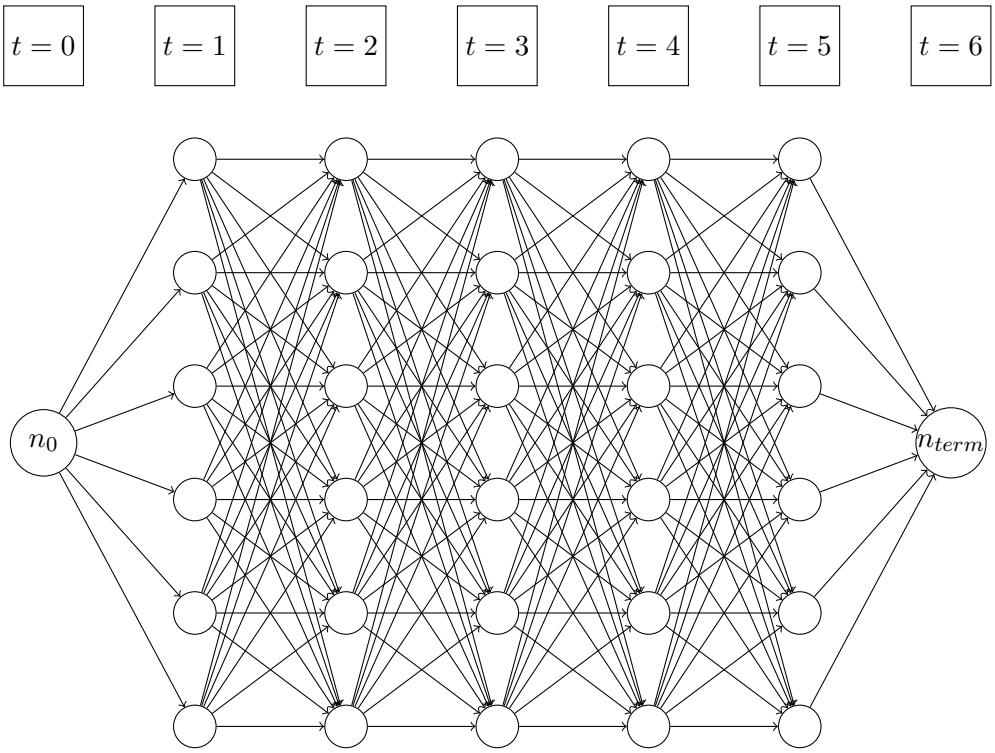

Figure 6: a Directed Network

The weights on the arcs represent the profit, or risk reduction, that is obtained by going from some allocation of the task groups to the allocation in the next period. Let the weight on an arc (n, n') be denoted by $w(n, n')$. The weights are calculated by multiplying the risk density in the cells that will be visited with the measure of effectiveness of the task group that is assigned to the cell, see equation (2).

$$w(n, n') = \sum_{j \in \mathcal{J}} r_{n'_j, t(n')} g_j(n_j, n'_j) \tag{2}$$

A planning for the total period can be represented by a path along the graph from beginning to end. The total risk reduction of such a path is given by the sum of the weights on the arcs along the path. We denote a path from period 1 to period t by $\mathcal{P}_t$. The node which is visited at time t', $t' = 0, \ldots, t$, in path $\mathcal{P}_t$ is given by $\mathcal{P}_t(t')$. The profit of such a path is given by $z(\mathcal{P}_t)$, see equation (3). We want to find the path with the highest value for $z(\mathcal{P}_T)$. Therefore the problem is defined as shown in equation (4).

$$z(\mathcal{P}_t) = \sum_{i=1}^{t} w(\mathcal{P}_t(i-1), \mathcal{P}_t(i)) \tag{3}$$

$$NMSP = \max z(\mathcal{P}_T) \tag{4}$$

4.2 Network Formulation - MSP

For the MSP we will need to make a few adjustments to the NMSP formulation. In the NMSP a task group reduces the risk in the current period if it visits a cell. In the MSP a task group reduces the risk in the current period but also in the following period when it visits a cell. The risk in the current period is reduced in the same way as in the NMSP. The risk in the following period is reduced completely. The risk in a cell now depends on the locations of the task groups in the previous period. Namely, if a cell was visited in the previous period, the risk is equal to zero. Else it is equal to $r_{c,t}$. To incorporate this in the model we introduce $\hat{r}_{c,t}|\mathcal{P}_{t-1}$, the risk density in cell c at time t, given a subpath $\mathcal{P}_{t-1}$. We will simply denote this by $\hat{r}_{c,t}$, however note that this value is conditional to $\mathcal{P}_{t-1}$, see equation (5). We also introduce an indicator function $I(c, \mathcal{P}_t(t-1))$ which is equal to 1 if one of the task groups visited cell c in period $t-1$ and 0 else in the chosen subpath. The new function for the weights is given in equation (6). The first part is the same as in equation (2). The second part is the risk that we reduce in the following period. In equation (7) the new objective function is given which is the sum over all the conditional weights on the arcs along the chosen path. We want to find the path that maximizes the value of $\hat{z}$, see (8).

$$\hat{r}_{c,t} = r_{c,t}(1 - I(c, \mathcal{P}_t(t-1))) \tag{5}$$

$$\hat{w}(n, n') = \sum_{j \in \mathcal{J}} \hat{r}_{n'_j, t(n')} g_j(n_j, n'_j) + r_{n'_j, t(n')+1} \tag{6}$$

$$\hat{z}(\mathcal{P}_t) = \sum_{i=1}^{t} \hat{w}(\mathcal{P}_t(i-1), \mathcal{P}_t(i)) \tag{7}$$

$$MSP = \max \hat{z}(\mathcal{P}_T) \tag{8}$$

4.3 Linear Formulation

Besides the formulation of the problem as a network, we also consider a linear formulation of the problem. This formulation is based on Sato and Royset (2010), however it had to be changed for our situation. In Sato and Royset (2010) a linear formulation is presented for cases where the searchers are identical. Some of the problems that we will be facing do not consider identical searchers. Therefore we have extended the formulation in Sato and Royset (2010) for different task groups. We will first introduce the original model that does not consider different capabilities for the task groups and thereafter extend the model.

4.3.1 Identical Searcher NMSP

Sets

$\mathcal{C} = \{1, \ldots, C\}$	set with all cells in the AOO
$\mathcal{T} = \{1, \ldots, T\}$	set with all time periods
$\mathcal{F}(c) \subseteq \mathcal{C}$	the cells that can be reached from cell c
$\mathcal{R}(c) \subseteq \mathcal{C}$	the cells that can reach cell c

Parameters

α	effectiveness of a task group
$d_{c,c'}$	fraction of a period required to travel from cell $c \in \mathcal{C}$ to $c' \in \mathcal{C}$
$r_{c,t}$	risk density in cell $c \in \mathcal{C}$ at time $t \in \mathcal{T}$
x_c^0	equals 1 if a task group must begin in cell $c \in \mathcal{C}$ in period 0, otherwise 0
x_c^{T+1}	equals 1 if a task group must end in cell $c \in \mathcal{C}$ in period T+1, otherwise 0

Variables

$X_{c,c',t}$	equals 1 if a task group goes from cell $c \in \mathcal{C}$ to cell $c' \in \mathcal{C}$ in which he will stay during period t and 0 otherwise.
$Q_{c,c',t}$	equals $r_{c,t}\alpha(1 - d_{c,c'})$ if $X_{c,c',t} = 1$ and 0 otherwise.

Functions

$z(Q)$	$= \sum_{t \in \mathcal{T}} \sum_{c \in \mathcal{C}} \sum_{c' \in \mathcal{F}(c)} Q_{c,c',t}$

$$\max_{X,Q} \quad z(Q) \tag{9}$$

$$s.t \qquad Q_{c,c',t} \leq r_{c,t}\alpha(1 - d_{c,c'})X_{c,c',t} \qquad \forall c, c' \in \mathcal{C}, t \in \mathcal{T} \tag{10}$$

$$\sum_{c' \in \mathcal{F}(c)} X_{c,c',1} = x_c^0 \qquad \forall c \in \mathcal{C} \tag{11}$$

$$\sum_{c' \in \mathcal{R}(c)} X_{c',c,T} = x_c^T \qquad \forall c \in \mathcal{C} \tag{12}$$

$$\sum_{c' \in \mathcal{R}(c)} X_{c',c,t} = \sum_{c'' \in \mathcal{F}(c)} X_{c,c'',t+1} \qquad \forall c \in \mathcal{C}, t = 1, \ldots, T-1 \tag{13}$$

$$\sum_{c' \in \mathcal{F}(c)} X_{c,c',t} = 1 \qquad \forall c \in \mathcal{C}, t \in \mathcal{T} \tag{14}$$

$$\sum_{c' \in \mathcal{R}(c)} X_{c',c,t} = 1 \qquad \forall c \in \mathcal{C}, t \in \mathcal{T} \tag{15}$$

$$Q_{c,c',t} \geq 0 \qquad \forall c, c' \in \mathcal{C}, t \in \mathcal{T} \tag{16}$$

$$X_{c,c',t} \in \{0, 1\} \qquad \forall c, c' \in \mathcal{C}, t \in \mathcal{T} \tag{17}$$

The objective function, see equation (9) is given by the sum of all the risk reductions, $Q_{c,c',t}$, over the whole period. We need to maximize this value in order to obtain the highest risk reduction. The constraints shown in equation (10) force $Q_{c,c',t}$ to be 0 if we do not visit cell c' from cell c in period t. Else $Q_{c,c',t}$ must be smaller than the risk that we reduce in period t.

The constraints given in equation (11) and (12) are used when we require the task groups to start from a specific cell or end in a specific cell. If we do not require this, we need to add some artificial cell with distance 0 to all other cells. The constraints given in equation (13) require task groups that enters cell c in period t to depart from this cell in period $t+1$. In other words, if a task group goes in a cell it must continue from this cell and task groups can only leave a cell if they actually were in that cell. This constraint, in combination with the constraints given in equations (11) and (12) also guarantee that we do not use more than J task groups. To prevent two task groups from being in the same cell at the same time we need the constraints in equations (14) and (15).

Finally we want all values of Q to be non-negative and the value of X to be binary. This is the mixed integer formulation of the NMSP with identical searchers. We will continue with the formulation of the MSP for identical searchers.

4.3.2 Identical Searcher MSP

Sets

$\mathcal{C} = \{1, \ldots, C\}$	set with all cells in the AOO
$\mathcal{T} = \{1, \ldots, T\}$	set with all time periods
$\mathcal{F}(c) \subseteq \mathcal{C}$	the cells that can be reached from cell c
$\mathcal{R}(c) \subseteq \mathcal{C}$	the cells that can reach cell c

Parameters

α	effectiveness of a task group
$d_{c,c'}$	fraction of a period required to travel from cell $c \in \mathcal{C}$ to $c' \in \mathcal{C}$
$r_{c,t}$	risk density in cell $c \in \mathcal{C}$ at time $t \in \mathcal{T}$
x_c^0	equals 1 if a task group must begin in cell $c \in \mathcal{C}$ in period 0, otherwise 0
x_c^{T+1}	equals 1 if a task group must end in cell $c \in \mathcal{C}$ in period T+1, otherwise 0

Variables

$X_{c,c',t}$	equals 1 if a task group goes from cell $c \in \mathcal{C}$ to cell $c' \in \mathcal{C}$ in which he will stay during period t and 0 otherwise.
$Q_{c,c',t}$	equals $R_{c,t}\alpha(1 - d_{c,c'}) + r_{c,t+1}$ if $X_{c,c',t} = 1$ and 0 otherwise.
$R_{c,t}$	the updated risk density.

Functions

$$z(Q) = \sum_{t \in \mathcal{T}} \sum_{c \in \mathcal{C}} \sum_{c' \in \mathcal{F}(c)} Q_{c,c',t}$$

$$f(X_{.,c,t-1}) = 1 - \sum_{c' \in \mathcal{C}} X_{c',c,t-1}$$

$$\max_{X,Q,R} \quad z(Q) \tag{18}$$

$$s.t \qquad Q_{c,c',t} \leq (R_{c,t}\alpha(1-d_{c,c'}) + r_{c,t+1})X_{c,c',t} \qquad \forall c,c' \in \mathcal{C}, t \in \mathcal{T} \tag{19}$$

$$R_{c,1} = r_{c,1} \qquad \forall c \in \mathcal{C} \tag{20}$$

$$R_{c,t} = r_{c,t} f(X_{.,c,t-1}) \qquad \forall c \in \mathcal{C}, t = 2,\ldots,T \tag{21}$$

$$\sum_{c' \in \mathcal{F}(c)} X_{c,c',1} = x_c^0 \qquad \forall c \in \mathcal{C} \tag{22}$$

$$\sum_{c' \in \mathcal{R}(c)} X_{c',c,T} = x_c^T \qquad \forall c \in \mathcal{C} \tag{23}$$

$$\sum_{c' \in \mathcal{R}(c)} X_{c',c,t} = \sum_{c'' \in \mathcal{F}(c)} X_{c,c'',t+1} \qquad \forall c \in \mathcal{C}, t = 1,\ldots,T-1 \tag{24}$$

$$\sum_{c' \in \mathcal{F}(c)} X_{c,c',t} = 1 \qquad \forall c \in \mathcal{C}, t \in \mathcal{T} \tag{25}$$

$$\sum_{c' \in \mathcal{R}(c)} X_{c',c,t} = 1 \qquad \forall c \in \mathcal{C}, t \in \mathcal{T} \tag{26}$$

$$Q_{c,c',t}, R_{c,t} \geq 0 \qquad \forall c,c' \in \mathcal{C}, t \in \mathcal{T} \tag{27}$$

$$X_{c,c',t} \in \{0,1\} \qquad \forall c,c' \in \mathcal{C}, t \in \mathcal{T} \tag{28}$$

The objective function, see equation (18) is the same as in the NMSP. The constraints shown in equation (19) force $Q_{c,c',t}$ to be 0 if we do not visit cell c' from cell c in period t. Else $Q_{c,c',t}$ must be smaller than the risk that we reduce in period t plus the risk that we reduce in the following period. These constraints are however non-linear because we multiply two variables. Therefore we replace them with the constraints in the equations (29) and (30). The constraints given in equation (29) force us to set the value of $Q_{c,c',t}$ to 0 if $X_{c,c',t}$ is 0. Otherwise it must be smaller than the initial risk density in period t multiplied by the effective time spend in cell c' plus the risk density in period $t+1$. This value is an upper bound on the real risk reduction because the initial risk density is always larger than or equal to the updated risk value. To make sure we choose a correct value for $Q_{c,c',t}$ we need the constraints in equation (30). In here we require $Q_{c,c',t}$ to be smaller than the updated risk value, multiplied by the MOE of a task group plus the initial risk in period $t+1$.

$$Q_{c,c',t} \leq (r_{c,t}\alpha(1-d_{c,c'}) + r_{c,t+1})X_{c,c',t} \qquad \forall c,c' \in \mathcal{C}, t \in \mathcal{T} \tag{29}$$

$$Q_{c,c',t} \leq R_{c,t}\alpha(1-d_{c,c'}) + r_{c,t+1} \qquad \forall c,c' \in \mathcal{C}, t \in \mathcal{T} \tag{30}$$

The updated risk value in period 1 is equal to $r_{c,1}$ since there has been no search yet, see equation (20). After period 1 the risk value should be updated, see equation (21). The updated risk value, $R_{c,t}$, depends on the values of $X_{c',c,s}$ for all $c' \in \mathcal{R}_{(c)}$ and for $s \leq t$. The function $f(X_{.,c,t-1})$ is equal to 1 if any task group visits cell c in period $t-1$ and 0 otherwise. As results $R_{c,t}$ becomes 0 if cell c was visited in period $t-1$ and $r_{c,t}$ otherwise.

The remaining constraints in equations (22)- (28) are exactly the same as in the NMSP, except for constraint (27) in which we also require R to be non-negative.

We will now look at the formulations for the different searcher problems, starting with the NMSP formulation.

4.3.3 Different Searcher NMSP

Sets

$\mathcal{C} = \{1, \ldots, C\}$	set with all cells in the AOO
$\mathcal{T} = \{1, \ldots, T\}$	set with all time periods
$\mathcal{J} = \{1, \ldots, J\}$	set with all task groups
$\mathcal{F}(c) \subseteq \mathcal{C}$	the cells that can be reached from cell c
$\mathcal{R}(c) \subseteq \mathcal{C}$	the cells that can reach cell c

Parameters

$d_{j,c,c'}$	time in periods required to travel between cells $c \in \mathcal{C}$ and $c' \in \mathcal{C}$ for task group $j \in \mathcal{J}$
$r_{c,t}$	risk density in cell $c \in \mathcal{C}$ at time $t \in \mathcal{T}$ and $t = T + 1$
α_j	effectiveness of task group $j \in \mathcal{J}$
$x^0_{c,j}$	equals 1 if task group $j \in \mathcal{J}$ must begin in cell $c \in \mathcal{C}$ in period 0, otherwise 0
$x^{T+1}_{c,j}$	equals 1 if task group $j \in \mathcal{J}$ must end in cell $c \in \mathcal{C}$ in period T+1, otherwise 0

Variables

$X_{j,c,c',t}$	equals 1 if task group $j \in \mathcal{J}$ goes from cell $c \in \mathcal{C}$ to cell $c' \in \mathcal{C}$ in which he will stay during period t and 0 otherwise.
$Q_{j,c,c',t}$	equals $r_{c,t}\alpha_j(1 - d_{j,c,c'})$ if $X_{j,c,c',t} = 1$ and 0 otherwise.

Functions

$$z(Q) = \sum_{j\in\mathcal{J}} \sum_{t\in\mathcal{T}} \sum_{c\in\mathcal{C}} \sum_{c'\in\mathcal{F}(c)} Q_{j,c,c',t}$$

$$f(X_{j,.,c,t-1}) = 1 - \sum_{c'\in\mathcal{C}} X_{j,c',c,t-1}$$

$$\max_{X,Q} \quad z(Q) \tag{31}$$

$$s.t \quad Q_{j,c,c',t} \leq (r_{c,t}\alpha_j(1 - d_{j,c,c'}))X_{j,c,c',t} \quad \forall c, c' \in \mathcal{C}, t \in \mathcal{T}, j \in \mathcal{J} \tag{32}$$

$$\sum_{c'\in\mathcal{F}(c)} X_{j,c,c',1} = x^0_{c,j} \quad \forall c \in \mathcal{C}, j \in \mathcal{J} \tag{33}$$

$$\sum_{c'\in\mathcal{R}(c)} X_{j,c',c,T} = x^T_{c,j} \quad \forall c \in \mathcal{C}, j \in \mathcal{J} \tag{34}$$

$$\sum_{c'\in\mathcal{R}(c)} X_{j,c',c,t} = \sum_{c''\in\mathcal{F}(c)} X_{j,c,c'',t+1} \quad \forall c \in \mathcal{C}, t = 1, \ldots, T-1, j \in \mathcal{J} \tag{35}$$

$$\sum_{c'\in\mathcal{F}(c), j\in\mathcal{J}} X_{j,c,c',t} = 1 \quad \forall c \in \mathcal{C}, t \in \mathcal{T} \tag{36}$$

$$\sum_{c'\in\mathcal{R}(c), j\in\mathcal{J}} X_{j,c',c,t} = 1 \quad \forall c \in \mathcal{C}, t \in \mathcal{T} \tag{37}$$

$$Q_{j,c,c',t} \geq 0 \quad \forall c, c' \in \mathcal{C}, t \in \mathcal{T}, j \in \mathcal{J} \tag{38}$$

$$X_{j,c,c',t} \in \{0, 1\} \quad \forall c, c' \in \mathcal{C}, t \in \mathcal{T}, j \in \mathcal{J} \tag{39}$$

The formulation of the different searcher NMSP is similar to the identical searcher NMSP, however we introduce an extra index j to distinct between the separate task groups. The objective function, see equation (31), is given by the sum of all the risk reductions, $Q_{j,c,c',t}$, over the whole period. The constraints in equations (32) guarantee that the value of $Q_{j,c,c',t}$ is chosen correctly. If $X_{j,c,c',t}$ equals one, the value of $Q_{j,c,c',t}$ is equal to $r_{c,t}\alpha_j(1-d_{j,c,c'})$ and else it will be equal to zero.

The constraints (33) - (37) are similar to (11) - (15), but adjusted for the separate task groups. Finally the restrictions on Q and Xremain the same in this formulation.

4.3.4 Different Searcher MSP

Sets

$\mathcal{C} = \{1, \ldots, C\}$	set with all cells in the AOO
$\mathcal{T} = \{1, \ldots, T\}$	set with all time periods
$\mathcal{J} = \{1, \ldots, J\}$	set with all task groups
$\mathcal{F}(c) \subseteq \mathcal{C}$	the cells that can be reached from cell c
$\mathcal{R}(c) \subseteq \mathcal{C}$	the cells that can reach cell c

Parameters

$d_{j,c,c'}$	time in periods required to travel between cells $c \in \mathcal{C}$ and $c' \in \mathcal{C}$ for task group $j \in \mathcal{J}$
$r_{c,t}$	risk density in cell $c \in \mathcal{C}$ at time $t \in \mathcal{T}$ and $t = T + 1$
α_j	effectiveness of task group $j \in \mathcal{J}$
$x^0_{c,j}$	equals 1 if task group $j \in \mathcal{J}$ must begin in cell $c \in \mathcal{C}$ in period 0, otherwise 0
$x^{T+1}_{c,j}$	equals 1 if task group $j \in \mathcal{J}$ must end in cell $c \in \mathcal{C}$ in period T+1, otherwise 0

Variables

$X_{j,c,c',t}$	equals 1 if task group $j \in \mathcal{J}$ goes from cell $c \in \mathcal{C}$ to cell $c' \in \mathcal{C}$ in which he will stay during period t and 0 otherwise.
$Q_{j,c,c',t}$	equals $R_{c,t}\alpha_j(1-d_{j,c,c'}) + r_{c,t+1}$ if $X_{j,c,c',t} = 1$ and 0 otherwise.
$R_{c,t}$	the updated risk density in cell $c \in \mathcal{C}$ at time $t \in \mathcal{T}$.

Functions

$$z(Q) = \sum_{j\in\mathcal{J}} \sum_{t\in\mathcal{T}} \sum_{c\in\mathcal{C}} \sum_{c'\in\mathcal{F}(c)} Q_{j,c,c',t}$$

$$f(X_{j,.,c,t-1}) = 1 - \sum_{c'\in\mathcal{C}} X_{j,c',c,t-1}$$

$$\max_{X,Q,R} \quad z(Q) \tag{40}$$

$s.t$

$$Q_{j,c,c',t} \leq (r_{c,t}\alpha_j(1-d_{j,c,c'}) + r_{c,t+1})X_{j,c,c',t} \quad \forall c, c' \in \mathcal{C}, t \in \mathcal{T}, j \in \mathcal{J} \tag{41}$$

$$Q_{j,c,c',t} \leq R_{c,t}\alpha_j(1-d_{j,c,c'}) + r_{c,t+1} \quad \forall c, c' \in \mathcal{C}, t \in \mathcal{T}, j \in \mathcal{J} \tag{42}$$

$$R_{c,1} = r_{c,1} \quad \forall c \in \mathcal{C} \tag{43}$$

$$R_{c,t} = r_{c,t} f(X_{j,.,c,t-1}) \quad \forall c \in \mathcal{C}, t = 2, \ldots, T, j \in \mathcal{J} \tag{44}$$

$$\sum_{c' \in \mathcal{F}(c)} X_{j,c,c',1} = x^0_{c,j} \quad \forall c \in \mathcal{C}, j \in \mathcal{J} \tag{45}$$

$$\sum_{c' \in \mathcal{R}(c)} X_{j,c',c,T} = x^T_{c,j} \quad \forall c \in \mathcal{C}, j \in \mathcal{J} \tag{46}$$

$$\sum_{c' \in \mathcal{R}(c)} X_{j,c',c,t} = \sum_{c'' \in \mathcal{F}(c)} X_{j,c,c'',t+1} \quad \forall c \in \mathcal{C}, t = 1, \ldots, T-1, j \in \mathcal{J} \tag{47}$$

$$\sum_{c' \in \mathcal{F}(c), j \in \mathcal{J}} X_{j,c,c',t} = 1 \quad \forall c \in \mathcal{C}, t \in \mathcal{T} \tag{48}$$

$$\sum_{c' \in \mathcal{R}(c), j \in \mathcal{J}} X_{j,c',c,t} = 1 \quad \forall c \in \mathcal{C}, t \in \mathcal{T} \tag{49}$$

$$Q_{j,c,c',t}, R_{c,t} \geq 0 \quad \forall c, c' \in \mathcal{C}, t \in \mathcal{T}, j \in \mathcal{J} \tag{50}$$

$$X_{j,c,c',t} \in \{0,1\} \quad \forall c, c' \in \mathcal{C}, t \in \mathcal{T}, j \in \mathcal{J} \tag{51}$$

The objective function, see equation (40), is given by the sum of all the risk reductions, $Q_{j,c,c',t}$, over the whole period. Again we want to maximize this value. The constraints in equations (41) and (42) guarantee that the value of $Q_{j,c,c',t}$ is chosen correctly. The first restriction forces us to set the value of $Q_{j,c,c',t}$ to 0 if $X_{j,c,c',t}$ is 0. Otherwise it must be smaller than $r_{c,t}\alpha_j(1-d_{j,c,c'}) + r_{c,t+1}$. This value is an upper bound on the actual risk that is reduced by visiting cell c' from cell c in period t with task group j. To make sure we choose a correct value for $Q_{j,c,c',t}$ we need the constraints in equation (42). In here we require $Q_{j,c,c',t}$ to be smaller than $r_{c,t}\alpha_j(1-d_{j,c,c'}) + r_{c,t+1}$.

The constraints (43) - (49) are similar to (20) - (26), but adjusted for the separate task groups. Finally the restrictions on Q, X and R remain the same in this formulation.

5 Solving NMSP and MSP

We will now introduce the methods that are used to solve the problems in this thesis. This section is divided in two parts, since we are considering two different formulations. The first part is about the network formulation. In this part we explain the two methods that we use to solve this formulation. These methods, simulated annealing and cross-entropy are both metaheuristics. The second part explains how we solve the mixed-integer linear formulations of the four problems. In this case we will use CPLEX to solve the problem.

5.1 Network Formulation

To solve the network formulation we will use two different metaheuristics. Since we do already consider a linear formulation which we will solve exactly, we will not consider any exact techniques to solve the network formulation. We expect that the larger problems cannot be solved to optimality and therefore we think it will be useful to develop metaheuristics that can still handle larger problems. These metaheuristics should be able to find (near) optimal solutions within a reasonable time. We will now explain the two metaheuristics that are used in this thesis.

5.1.1 Cross-Entropy Method

When solving the network formulation of the (N)MSP we want to construct a feasible and optimal path through the network. This optimal path should follow arcs from the beginning of the network to the end of the network in such a way that the total value of these (weighted) arcs is maximized. A simple way to construct paths is by starting in the first node and randomly selecting arcs until we reach the terminal node. The cross-entropy method is based on this idea. However, instead of selecting arcs perfectly random, the arcs are selected according to a probability distribution. For each node $n \in \mathcal{N}$ we specify a probability distribution $\sigma_{n,n'}$ over the outgoing arcs (n, n') for all $n' \in \mathcal{F}(n)$. This distribution specifies the likelihood of going from node n to node n'.

The probability distribution is updated in each iteration in order to achieve a distribution that specifies the optimal path. To achieve such a distribution we first generate a large amount of paths according to our initial distribution. Then we evaluate these paths and select a fraction of the paths with the highest objective value. These so called Elite paths are used to update our distribution. The arcs that are chosen in the elite paths will become more likely to be chosen in further iterations. After updating the probability distribution a new set of solutions is generated and the process is repeated. The pseudocode of the cross-entropy method is given in heuristic 1.

As input parameters we require the sample size M, which specifies the number of paths we want to generate in each iteration. The elite sample size M^{elite} is required, which specifies the number of samples we consider as superior. Obviously M must be larger than M^{elite} since the set of elite samples is a subset of all samples.

In step 0 we set the best solution so far equal to zero. And we start with iteration 1. In step 1 we initiate the distribution for generating the paths. This could be done in several ways. The better the initial distribution, the easier we may find an optimal solution. In previous research this was done by creating an upper bound on the path. We have chosen to set the probability of selecting arc n' when we are in position n equal to the value of arc

Input Sample size M, elite size M^{elite}, stopping parameter s and smoothing parameter β.
Step 0 Set $z^* = 0$ and $i = 1$
Step 1 For all $n \in \mathcal{N}$ and $n' \in \mathcal{F}(n)$, define an initial probability distribution $\sigma_{n,n'}^{(1)}$
Step 2 Generate M search plans based on probability distribution $\sigma_{n,n'}^{(i)}$.
Step 3 Evaluate all search plans. Select M^{elite} best plans. Set $\bar{z}^{(i)}$ to $\underset{m \in M^{elite}}{max}\, z(m)$. If $\bar{z}^{(i)} \geq z^*$, then $z^* = \bar{z}^{(i)}$.
Step 4 If $\bar{z}^{(i)} = \bar{z}^{(i-1)} = \ldots = \bar{z}^{(i-s)}$, then stop. Otherwise go to Step 5.
Step 5 Calculate $\sigma_{n,n'}^{(i+1)}$ with equation (52). Increase i by 1 and go to step 2.

$$\sigma_{n,n'}^{(i+1)} = \beta \frac{l_{n,n'}^{(i)}}{\sum_{n' \in \mathcal{F}(n)} l_{n,n'}^{(i)}} + (1-\beta)\sigma_{n,n'}^{(i)} \tag{52}$$

With $l_{n,n'}^{(i)}$ the number of times arc (n, n') was selected in the elite samples.

Heuristic 1: cross-entropy

$w(n, n')$ divided by the sum of the weights of all outgoing arcs belonging to n. The initial value of the distribution can be calculated with equation (53).

$$\sigma_{(n,n')}^{(0)} = \frac{w(n, n')}{\sum_{n' \in \mathcal{F}(n)} w(n, n')} \tag{53}$$

In step 2 we generate M search plans using the probability distribution we defined. This is done by starting in the initial node and selecting the arcs according to the probability distribution until we reach the terminal node. In step 3 we calculate the objective values of the M paths. From these paths the M^{elite} best paths are selected. Furthermore we check if the best search plan of this iteration, $\bar{z}^{(i)}$, is better than the best solution so far, z^*. If this is true we update z^*. In step 4 we check whether there has been any change in the best solution during the last s iterations. If this is the case we continue, otherwise we assume that we have found the optimum. In step 5 the probability function of the arcs is updated for the next iteration. The new probability $\sigma_{n,n'}^{(i+1)}$ depends on the value of $l_{n,n'}$, the amount of paths in the elite sample in which arc (n, n') was used. The new value for $\sigma_{n,n'}^{(i+1)}$, the probability of going to node n' when we are in node n is equal to a discounted value of the old probability plus the fraction of elite paths in which we have chosen to go to node n' when we were in node n. After updating the probability we return to step 2 and the process is repeated.

When facing problems with a large amount of cells we will have a very large amount of nodes, which are connected with a lot of other nodes. This results in an incredibly large amount of arcs. For each of these arcs we need to keep track of the value for $\sigma_{n,n'}$. For an increasing number of cells the amount of arcs quickly escalates. To prevent this we will introduce a variation on the cross-entropy method, also used by Sato (2008). Instead of generating paths through the weighted directed network, we will generate paths for each task group separately. Each task group gets its own distribution, $\sigma_{j,n,n'}$, to generate paths with. The nodes in this case do not represent the locations of all task groups, but just of a single task group. After generating paths for each task group separately, the paths are combined

and evaluated together in the same way as is done before. The rest of the heuristic will remain the same. This heuristic is given in heuristic 2.

We have also added an extra step to the original cross-entropy to improve the heuristic. In this extra step, step 6 in heuristic 2, we add some randomness in the probability distribution. The heuristics goes to this step if the last s iterations have found the same best solution, which indicates that the probability distribution has converged. This randomness creates the opportunity to move out of a local optimum. The randomness is achieved by multiplying the value of σ with γ and adding $(1-\gamma)$. The pseudocode in Heuristic 2 represents the adjusted cross-entropy method. We also need a new stopping criterion. We have now choose to stop the heuristic if there has been no improvement in the best solution during the last i_{max} iterations

Input Sample size M, elite size M^{elite}, stopping parameter s and smoothing parameter β.
Step 0 Set the best solution so far $z^* = 0$ and $i = 1$
Step 1 For all $j \in \mathcal{J}$, $n \in \mathcal{N}$ and $n' \in \mathcal{F}(n)$, define an initial probability distribution $\sigma^{(1)}_{j,n,n'}$
Step 2 Generate M search plans for each task group, based on probability distribution $\sigma^{(i)}_{j,n,n'}$.
Step 3 Evaluate the joint search plans of all task groups combined. Select M^{elite} best plans. Set $\bar{z}^{(i)}$ to $\underset{m \in M^{elite}}{max}\, z(m)$. If $\bar{z}^{(i)} \geq z^*$, then $z^* = \bar{z}^{(i)}$. If there has been no improvement in the solution for i_{max} iterations, stop.
Step 4 If $\bar{z}^{(i)} = \bar{z}^{(i-1)} = \ldots = \bar{z}^{(i-s)} =$, then go to step 6. Otherwise go to Step 5.
Step 5 Calculate $\sigma^{(i+1)}_{j,n,n'}$ with equation (54). Increase i by 1 and go to step 2.

$$\sigma^{(i+1)}_{j,n,n'} = \beta \frac{l^{(i)}_{j,n,n'}}{\sum_{n' \in \mathcal{F}(n)} l^{(i)}_{j,n,n'}} + (1-\beta)\sigma^{(i)}_{j,n,n'} \tag{54}$$

With $l^{(i)}_{j,n,n'}$ the number of times arc (n, n') was selected in the elite samples for task group $j \in \mathcal{J}$.
Step 6 Include some randomness in the distribution by applying updating σ as follows.

$$\sigma^{(i)}_{j,n,n'} = (1-\gamma) + \gamma\sigma^{(i)}_{j,n,n'} \tag{55}$$

Heuristic 2: cross-entropy-II

5.1.2 Simulated annealing

Simulated annealing is one of the most common metaheuristics. This metaheuristic is basically a neighbourhood search in which we allow worse solutions to be chosen. The probability to choose solutions of lower quality depends on the cooling parameter, or the temperature, which is slowly reduced. At some point the probability of selecting worse solutions will become so small that we only accept solutions with a higher value. If we cannot find any improvement from this point we terminate the program. The idea behind simulated annealing is that we search many local optima in the hope to find a global optimum. Because we also accept worse solutions we do not easily get stuck in local optima (assuming the parameter setting is done correctly). The exact steps of the heuristic are shown in the following pseudo-code.

Step 0 Set temperature to t_0 and start with a path $\mathcal{P}$
Step 1 Randomly select a neighbour of $\mathcal{P}$; $\mathcal{P}_{new}$.
Step 2 If $z(\mathcal{P}_{new}) > z(\mathcal{P})$, accept the neighbour. Otherwise accept the neighbour with probability $p(Temperature, z(\mathcal{P}_{new}))$.
Step 3 Update the temperature: $Temperature = Temperature \times \beta$, where β is the cooling parameter. If the temperature is higher than t_{end}, return to step 1.
Step 4 Return the best solution found.

Heuristic 3: Simulated annealing

In step 0 the heuristic is initialized. The beginning temperature has to be chosen such that we will accept all solutions in the neighbourhood of our current solution. In the next section we will explain our choice for the parameters such as beginning temperature. In step 0 we also need to construct a feasible path. This can be done with a greedy heuristic. In step 1 we will have to select a neighbour from the chosen path. To select neighbours we first need to define the neighbourhood of a solution. This is often the critical part in simulated annealing, as well as in other metaheuristics. We have considered several different neighbourhoods and tested them all. From these neighbourhoods we have selected the one that performs the best. The following neighbourhood choices were considered for our simulated annealing heuristic.

- change the position of one task group in one period
- change the position of several task groups in one period
- change the position of several task groups in several periods

The first neighbourhood is the most basic neighbourhood that one can think of. This neighbourhood only looks at the change of location of one task group in one period. The benefit of this neighbourhood above others is that it can do single changes that are beneficial. However it might sometimes be good to do a switch of allocation between two task groups. Therefore we introduce neighbourhood two, in which we change the location of several task groups in a single period. The third neighbourhood is the broadest and considers the change of several task groups in several periods.

In step 2 we determine whether we want to accept the neighbour or not. If the neighbour has a better objective function value than the previous path we automatically accept it, however if this is not the case we accept it with a probability depending on the difference in objective value and the current temperature. The probability can be calculated with equation (56). This probability is high if the difference between the two paths is small. Note that the value of $z(\mathcal{P}_{new}) - z(\mathcal{P}_{old})$ is always negative. We know that e^{-x} approaches 1 if x approaches zero. The probability of accepting a neighbour with a solution that is much lower than our current solution is small since e^{-x} approaches 0 if x becomes large. The probability of accepting depends on the value of the temperature. If the temperature is very large we might accept all possible changes. If the temperature becomes low we do accept only neighbours whose objective value is very close to the current solution.

$$p(Temperature, z(\mathcal{P}_{new})) = e^{\frac{z(\mathcal{P}_{new}) - z(\mathcal{P}_{old})}{Temperature}} \tag{56}$$

In step 3 we update the temperature. If the temperature becomes lower than the specified lowest temperature we will stop searching for better solutions. If this is not the case we will return to step 1.

5.2 Linear Formulation

The mixed-integer linear formulation of the problem is based on Sato and Royset (2010). In this paper a linear formulation for the MSP was introduced. This formulation requires that the searchers are identical. They also presented a nonlinear formulation for the MSP with different searchers. We have changed the linear formulation in such a way that it can be used for different types of searchers, without making the problem nonlinear. Sato and Royset used several methods to solve the problem. The best performing methods were 3 different cutting plane algorithms and the CPLEX solver. The cutting plane methods performed best in situations with an increasing amount of searchers, however in instances with not too many searchers CPLEX turned out to be the best method. In our case we are not facing a problem with increasing searchers and therefore we think it will be better to use the CPLEX solver. We have implemented all four formulations into AIMMS, a mathematical modeling software package, and used the implemented CPLEX 12.4 solver to solve the problems.

6 Results

We have applied our methods to several cases to test the quality of the suggested methods. Because there are no real data available we have generated our own data. As data we required a risk density for each cell in each period. In the problem description we have explained that the risk density is actually a combination of the threat map (density of pirates) and the presence of white vessels. In order to generate data we separated the two parts, which are the ingredients for creating the risk densities. First we created a density map for the pirates, which is nothing more then a single number for each cell in the area. As stated in the problem description, this value is fixed over the whole period. We think it is unlikely that every cell has the same pirate density and therefore we have chosen different densities for different cells. Some cells are dealing with a high pirate density, while others have a very low density. The presence of white vessels changes each period. We generated the presence of white vessels in each cell with an exponential distribution. The parameter for this distribution is different per cell, as it is unlikely that white vessels pass each cell with the same likelihood. We have chosen to use the exponential distribution, because we do not want negative values and we want a possibility for high values. An example of one of the cases is shown in figure 7. In this figure we can see an area of 50 cells during 10 periods. The risk density is shown on a scale from black (low risk) to white (high risk). This figure shows us that the cells are generated with different parameters. E.g. the upper cells has a much lower chance of high risk than the central cells.

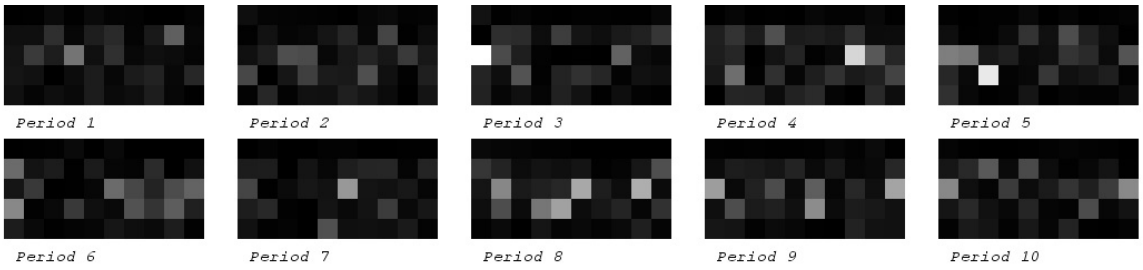

Figure 7: An example of a generated case (50 cells, 10 periods)

The cases that we will consider are listed in table 1. We will consider three different sizes for the AOO; 50, 100 and 200 cells. The sizes are chosen in correspondence with realistic surveillance missions. A total AOO of 100 to 200 cells could correspond with a realistic maritime surveillance mission. For each of the types of AOO we consider 10, 20 and 30 time periods. In all cases we will use 3 task groups. We will consider both the case where we have identical searchers as well as the case where we use searchers with different qualities. In the identical searcher cases, the task groups are all operating with $\alpha_j = 1$. The distance that can be traversed by a task group in one period is set equal to 3. In the different searcher cases, we set $\alpha_1 = 1$, $\alpha_2 = 0.75$ and $\alpha_3 = 0.5$. The distance that can be traversed by the task groups are 3, 4 and 5 respectively.

We will consider all cases in both the NMSP and the MSP problem, meaning there are actually 36 cases. For each case we generated 10 instances to make the results more accurate. This amount does of course not guarantee that the results are perfectly accurate, however it gives us more certainty. To the NMSP as well as to the MSP we applied cross-entropy,

Case	nr. of Cells	nr. of Periods	nr. Task Groups	α Task Groups	speed Task Groups
1	50	10	3	$\{1, 1, 1\}$	$\{3, 3, 3\}$
2	50	20	3	$\{1, 1, 1\}$	$\{3, 3, 3\}$
3	50	30	3	$\{1, 1, 1\}$	$\{3, 3, 3\}$
4	100	10	3	$\{1, 1, 1\}$	$\{3, 3, 3\}$
5	100	20	3	$\{1, 1, 1\}$	$\{3, 3, 3\}$
6	100	30	3	$\{1, 1, 1\}$	$\{3, 3, 3\}$
7	200	10	3	$\{1, 1, 1\}$	$\{3, 3, 3\}$
8	200	20	3	$\{1, 1, 1\}$	$\{3, 3, 3\}$
9	200	30	3	$\{1, 1, 1\}$	$\{3, 3, 3\}$
10	50	10	3	$\{1, 0.75, 0.5\}$	$\{3, 4, 5\}$
11	50	20	3	$\{1, 0.75, 0.5\}$	$\{3, 4, 5\}$
12	50	30	3	$\{1, 0.75, 0.5\}$	$\{3, 4, 5\}$
13	100	10	3	$\{1, 0.75, 0.5\}$	$\{3, 4, 5\}$
14	100	20	3	$\{1, 0.75, 0.5\}$	$\{3, 4, 5\}$
15	100	30	3	$\{1, 0.75, 0.5\}$	$\{3, 4, 5\}$
16	200	10	3	$\{1, 0.75, 0.5\}$	$\{3, 4, 5\}$
17	200	20	3	$\{1, 0.75, 0.5\}$	$\{3, 4, 5\}$
18	200	30	3	$\{1, 0.75, 0.5\}$	$\{3, 4, 5\}$

Table 1: Case Scenarios

simulated annealing and CPLEX 12.4. Both the heuristics are implemented with java in the Eclipse compiler. We have implemented the mixed-integer linear problem in AIMMS. Everything was performed on a laptop with an intel i5-2450M processor (2,50 GHZ) with 4,00 GB RAM.

6.1 Setting the parameters

For both the metaheuristics we had to determine the right parameters in order to optimize their performance. Choosing the parameters is often a tricky part when using a metaheuristic. Simulated annealing also requires us to define a neighbourhood. In section 5 we explained three different neighbourhoods. We have applied them all to several instances of the problem with different parameter settings. It turned out that the most basic neighbourhood is able to obtain the best solutions. Both other neighbourhoods often got stuck before even solutions that were 10% from the optimal solution. Therefore we have chosen to use neighbourhood 1. The parameter that need to be defined in the simulated annealing heuristic are the beginning temperature, a cooling parameter and a stopping parameter.

The beginning temperature determines the likelihood of accepting worse solutions in the beginning. When we start the heuristic we want to accept all solutions, because we do not want to exclude the possibility of reaching the global optimum. Therefore we set the beginning temperature equal to twice the highest RD over all periods. The stopping parameter needs to be chosen in such a way that we will only stop if the probability of accepting a worse solution approaches zero, and we have searched the whole neighbourhood of the current solution. We chose to terminate the program when the temperature reaches 0.001 times the beginning temperature as we found that the heuristic often stops improving before even reaching 0.01

times its beginning temperature. The cooling parameter must be chosen in such a way that we search enough time in each area, however not too long. This parameter is often best defined by trial and error. We found that a value of $1 - 10^{-7}$ is a sufficient choice for this parameter. Choosing a smaller value, for example $1-10^{-6}$ results in evidently worse solutions. Choosing a value that is closer to 1 did not result in much better solutions, but mainly in a longer computation time. Therefore we have chosen to set the value of this parameter equal to $1 - 10^{-7}$.

For the cross-entropy method its difficult to determine a good set of parameters in a theoretical way. The required parameters are M, M_{elite}, β and γ which are the sample size, elite sample size, smoothing parameter and reset parameter respectively. The sample size determines how many solutions we will generate in each iteration. The elite sample size determines how many solutions we will use to update the probability distribution that is used for generating solutions. The smoothing parameter determines how much the distribution is changed after each iteration. Finally the reset parameter determines how much randomness we add if the heuristic gets stuck in a local optimum.

We tested many combinations of different values for M, M_{elite} and β. We found that choosing a high value for M, such as 10^5 or even 10^6 did not make the heuristic more efficient than a sample size of 10^4. Naturally each iteration takes much longer if M is chosen very large and this would be worth the effort if the program would find better solutions, however we did not find much difference between the improvement of the solutions when setting M equal to 10^4 and 10^6. Since each iteration in the latter case takes roughly 100 times longer we thought it was better to use a sample size of 10^4. We also tested smaller sample sizes, but that resulted in a heuristic that is initially quick, however after a while it gets stuck and from there it improves very slow. If M_{elite} is high the heuristic becomes somewhat like a random search. We found that a very small value for M_{elite} works best. The value of β determines how much the distribution changes in each iteration. If this value is chosen to large the distribution converges to quick and we get stuck in local optima. If β is chosen to small, it takes very long before anything happens. The value of γ is not so difficult to choose. It should be chosen in such a way that we allow enough randomness to move out of a local optimum, however not to big so we loose our current progression. Finally we have chosen to set $M = 10^4$, $M_{elite} = 3$, $\beta = 0.4$ and $\gamma = 0.90$. We stop the heuristic if there has been no improvement for 100 iterations.

6.2 Identical Searcher

6.2.1 NMSP

Table 2 shows us the results for the NMSP problem for identical searchers. The table contains the results for all three methods for case 1-9. The value in this table is the average deviation from optimality over 10 different instances (different random numbers) of the same case. CPLEX solves the problem to optimality for all cases. Simulated Annealing is not able to solve the problem to optimality for all cases, however its solutions are often near optimal. The solutions found by simulated annealing deviate on average little more than 1% from the optimal solution. Simulated annealing found the actual optimal solution in 10% of the cases. The worst solution found by simulated annealing deviated only 4.4% from optimality. Cross-entropy is able to find near optimal solutions for the smaller cases. In the cases with more cells, and especially in the cases with more periods the solutions are far from optimality.

Table 3 shows the average computation times for the cases. CPLEX is by far the fastest of the three methods, however simulated annealing and cross-entropy do not have very large computation times either.

Case	CE	SA	CPLEX
1	0.67	0.86	0.0
2	1.66	0.94	0.0
3	4.22	0.87	0.0
4	2.31	1.13	0.0
5	10.52	1.13	0.0
6	29.25	1.33	0.0
7	4.09	0.44	0.0
8	18.00	1.56	0.0
9	37.39	1.45	0.0

Table 2: Results as percentage from optimality

Case	CE	SA	CPLEX
1	99s	146s	$\leq$ 10s
2	187s	292s	$\leq$ 10s
3	307s	423s	$\leq$ 10s
4	181s	155s	$\leq$ 10s
5	318s	303s	$\leq$ 10s
6	541s	451s	28s
7	291s	165s	$\leq$ 10s
8	508s	314s	$\leq$ 10s
9	945s	461s	42s

Table 3: Computation Times

6.2.2 MSP

We now continue to look at the MSP for identical searchers. The results as average percentage from optimality are given in table 4. CPLEX is again able to solve case 1-9 to optimality. The results for simulated annealing are similar to those for the NMSP. The solutions deviate on average 1% from optimality. In almost 9% of the cases an optimal solution was found, and the largest deviation from optimality was 3.7%. If we look at the computation times in table 5 we see that the computation times for CPLEX have become much larger than for the NMSP, while the computation times for simulated annealing remain more or less the same. Even though simulated annealing does not find the optimal solution, it is much faster than CPLEX. Cross-entropy is unable to provide good solutions to the problem.

Case	CE	SA	CPLEX
1	1.88	0.24	0.0
2	3.25	0.48	0.0
3	20.14	0.46	0.0
4	2.61	0.12	0.0
5	11.04	0.74	0.0
6	26.85	0.68	0.0
7	1.63	0.43	0.0
8	16.20	1.12	0.0
9	34.88	1.17	0.0

Table 4: Results as percentage from optimality

Case	CE	SA	CPLEX
1	112s	164s	63s
2	220s	322s	268s
3	340s	500s	981s
4	212s	168s	123s
5	391s	329s	491s
6	649s	496s	2026s
7	370s	178s	340s
8	720s	342s	1683s
9	1085s	512s	3821s

Table 5: Computation Times

6.3 Different Searchers

We continue with the problem where searchers have different capabilities. For this problem we will also look at the NMSP and the MSP variant.

6.3.1 NMSP

The results in table 6 show us that CPLEX also performs good in the different searcher problem. CPLEX obtains optimal solutions in a short time. Though we can observe a growing computation time. For very large cases this might become a problem. Simulated annealing is again able to find good solutions, and does not seem to have any trouble with the different searcher problem. If we compare the solutions with the NMSP for identical searchers we do not observe a large difference in quality, nor in computation time for SA. Cross-entropy also performs similar to the identical searcher NMSP. Its solutions are near optimal for the smaller cases, but they become very bad for the larger cases.

Case	CE	SA	CPLEX
10	2.89	0.27	0.0
11	10.93	0.46	0.0
12	26.94	0.79	0.0
13	3.06	0.49	0.0
14	21.21	0.54	0.0
15	34.44	1.08	0.0
16	5.17	0.81	0.0
17	25.49	0.65	0.0
18	40.12	1.10	0.0

Table 6: Results as percentage from optimality

Case	CE	SA	CPLEX
10	101s	152s	5s
11	182s	284s	17s
12	310s	423s	31s
13	208s	154s	19s
14	341s	291s	141s
15	575s	425s	304s
16	310s	161s	77s
17	549s	299s	466s
18	1081s	436s	4212s

Table 7: Computation Times

6.3.2 MSP

The MSP for different searchers is different from the other problems. CPLEX is unable to solve most of the cases to optimality or even unable to find a solution due to memory issues. Therefore we have solved the linear relaxation of the problem instead to obtain an upper bound. We use this upper bound to determine an upper bound on the distance from optimality for the metaheuristics. We used CPLEX to solve the linear relaxation.

For case 10 CPLEX was still able to find a solution for most instances. It was even able to solve a single instance to optimality. The upper bound on the instances for case 10 is obtained by the lowest upper bound that was found by CPLEX when solving the original problem. In all other cases it was provided by the best solution of the linear relaxation of the problem. For the instance of case 10 that was solved to optimality the solution by simulated annealing was only 0.49% from the optimal solutions. The solutions for case 10 that were obtained by simulated annealing were on average 4.91% away from the upper bound that was found by CPLEX. CPLEX was unable to solve the linear relaxations for case 17 and 18. Therefore we cannot make any statements about the quality of the solutions produced by simulated annealing and cross-entropy for these cases.

If we look at the results in table 8 we see that the results for simulated annealing are much worse than the solutions to the other problems. This is likely due to the fact that

we used an upper bound instead of the real optimal solution. We do not expect that the solutions for simulated annealing are much worse than those for the other cases, because there is no significant difference between the quality of the NMSP and the MSP solutions in the identical searcher case. Neither is there a significant difference between the quality of the NMSP solutions for the identical and the different searcher cases. However there is no proof to justify these statements and therefore we have to settle with the upper bounds. The computation times for simulated annealing are still not very high. The results for cross-entropy are again far from optimality.

Case	CE	SA
10	10.12	4.91
11	24.75	14.84
12	36.53	14.37
13	17.76	15.45
14	26.96	10.73
15	42.94	13.24
16	13.91	8.57
17	-	-
18	-	-

Table 8: Results as percentage from UB

Case	CE	SA
10	121s	157s
11	239s	317s
12	369s	478s
13	207s	163s
14	417s	323s
15	649s	489s
16	385s	174s
17	772s	347s
18	1171s	502s

Table 9: Computation times

7 Conclusion and Recommendations

In this thesis we have developed several methods to optimize the allocation of task groups in maritime surveillance missions. We faced four different versions of the problem, namely the Non reactive Multiple Searcher problem with identical searchers, the Multiple Searcher problem with identical searchers, the Non reactive Multiple searcher problem with different searchers and finally the Multiple Searcher problem with different searchers.

We have extensively tested the developed solution methods for all four problems. For each problem we considered 9 different cases which vary in amount of cells and number of periods. Each case was generated 10 times to obtain stable results. The optimization methods that we used are a cross-entropy method, simulated annealing and CPLEX.

In the NMSP for identical searchers as well as for different searchers we found that CPLEX performs better than the metaheuristics. However in the MSP for identical searchers the CPLEX solver becomes very slow for the larger instances, while simulated annealing obtains near optimal solutions within a short time. In the MSP for different searchers we see the power of simulated annealing. Where the CPLEX solver cannot solve the problem anymore for large cases, simulated annealing still finds solutions in a reasonable time. The quality of these solutions is difficult to determine exactly because we do not have the exact solutions to the problem. Therefore we solved a linear relaxation of the problem to obtain an upper bound on the optimal solution. The results that were produced by simulated annealing, are within 15% of the upper bound on the optimal solution. The results for cross-entropy are less promising. In the smaller cases cross-entropy finds near optimal solutions, however when considering more than 10 periods cross-entropy is unable to find good solutions.

All with all we can conclude that the CPLEX solver as well as simulated annealing are very successful in solving the defined problem. CPLEX is able to solve a wide range of problems but is limited with respect to the size of the problem. Simulated annealing has shown to provide stable and near optimal solutions in a short time. The choice between these two methods is a choice between optimality and computation time.

Even though we are able to find optimal solutions to our defined problem, we should not forget that an optimal solution is only as good as the assumptions of the model and the data. If this method would be applied in a real surveillance missions for which it is also designed, it would require risk values that are representing the real situation as good as possible. The threat maps which indicate the likely locations of the pirates should be very accurate. The prediction of the movement of white vessels through the AOO should also be done as good as possible, which is difficult if we are looking far ahead. Furthermore the functions that are used in our model, such as the reduction of risk and the MOE, should be specified in a correct way. If this is done carelessly, the optimal solution is unlikely to be optimal for the real situation.

This thesis has faced the allocation of searchers in anti piracy surveillance missions. However, if one would desire to do so, the problem could easily be extended to other types of surveillance missions. The same method could be applied to maritime surveillance against smuggling or illegal fishing. It would only require a different measure instead of the risk density that we have used in this thesis. Furthermore different assumptions should be made about the MOE and the risk reduction function. Apart from maritime surveillance it could also be applied to surveillance on land. Though land has more restrictions in movement than water, one could easily overcome this by defining the distances between cells in a good way.

We will also do some recommendations with respect to further research. First of all it may

be interesting to improve the simulated annealing heuristic that we have used in this thesis. The current heuristic uses a very simple neighbourhood, which we think could be improved by combining the different neighbourhoods that we have mentioned in our research. This could be done by selecting a neighbour from the neighbourhood that we used with a probability p_1 and from another (or several other) neighbourhood(s) with a probability $1 - p_1$. We think this change could improve the current heuristic.

With respect to the cross-entropy heuristic we think there is more room for improvement. The solutions that were found for the larger cases were far from optimal. We expect that other parameter settings can lead to solutions that are closer to optimal. It is however likely that a longer computation time will be required to achieve solutions that are near optimal. It may also be difficult to obtain a version of the cross-entropy heuristic that can compete with the solutions of simulated annealing. Therefore we think that any effort with respect to this subject is better spend in improving simulated annealing than cross-entropy.

8 Literature

R.F. Dell, J.N. Eagle, G.H.A. Martins, and A.G. Santos. Using multiple searchers in constrained-path, moving-target search problems. *Naval Research Logistics*, 43(4):463–480, 1996.

J.N. Eagle. The optimal search for a moving target when the search path is constrained. *Operations research*, 32(5):1107–1115, 1984.

J.N. Eagle and J.R. Yee. An optimal branch-and-bound procedure for the constrained path, moving target search problem. *Operations research*, 38(1):110–114, 1990.

S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simmulated annealing. *science*, 220(4598):671–680, 1983.

G.H. Martins. A new branch-and-bound procedure for computing optimal search paths. Technical report, DTIC Document, 1993.

H. Sato. Path optimization for single and multiple searchers: models and algorithms. Technical report, DTIC Document, 2008.

H. Sato and J.O. Royset. Path optimization for the resource-constrained searcher. *Naval Research Logistics (NRL)*, 57(5):422–440, 2010.

T.J. Stewart. Search for a moving target when searcher motion is restricted. *Computers & operations research*, 6(3):129–140, 1979.

9 Appendix

9.1 Appendix A - List of Definitions

AOO	Area Of Operations
AOR	Area of Responsibility
Asset	A maritime unit such as Frigates, Helicopters and UAV's
NMSP	Non Reactive Multiple Searcher Problem
MOE	Measure of Effectiveness
MSP	Multiple Searcher Problem
RD	Risk Density, the risk in a certain area at some point in time
Task Force	All task groups together
Task Group	A combination of assets that stays together as a team
White Vessel	A vessel that is a possible victim for pirates

9.2 Appendix B - Variations

This thesis presents several methods to solve the identical and different searcher versions of both the NMSP and the MSP. We have made multiple assumptions, which mainly apply to the anti piracy missions that we consider. These assumptions are very likely to be wrong when considering other problems. In this section we will take a closer look at the assumptions and look if they can be changed without changing the difficulty of the problem.

9.2.1 Cells

We will first look at the assumptions regarding the cells. In our problem we have assumed that all cells are equally shaped and sized. If we would let go this assumption, thus allowing cells to have different sizes, would the problem change? The problem would in fact stay exactly the same because we can still define distances between cells and the risk density in cells. However one should note that if we have different sizes of cells, the effectiveness of a task group in cells might also change. It is off course more difficult to protect a cell that is larger. Therefore in a variation of the problem where different sizes of cells occur, it should be taken into account that a task group that operates in a larger cell should have a smaller measure of effectiveness. In figure 8 a simple instance with different sized cells is given. Two task groups are operating in this area, shown by the colored dots. The blue task group should have a larger measure of effectiveness than it would have in one of the other cells. We could incorporate this in the model by adding an extra parameter which is different for each cell, task group combination.

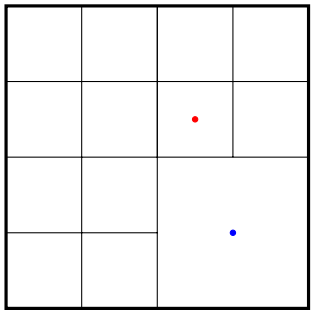

Figure 8: Variation in cell size

Another assumption with regards to the cells is that none of the cells overlap and that all cells together form the AOO. In a realistic mission it might be useful to ignore some parts of the AOO if nothing of interest could happen. For example, areas in the sea in which pirates are never sighted, or areas that are avoided by white vessels. Ignoring several cells does not directly affect the problem, but only makes it easier to solve. On the other hand, ignoring cells reduces the possibility to transit between other cells. Therefore it may not always be clever to leave out cells.

9.2.2 Risk reduction

In the MSP a risk reduction takes place after visiting a cell. We assumed that visiting a cell in period t resulted in a complete reduction of the risk in period $t+1$. This reduction was easy

to implement in the linear model, however if we would have chosen a different risk reduction this might have not been so easy. We will now look at three different reduction functions and look if we could use them without loosing the linearity of our model.

First of all we could have chosen a fixed reduction of $\epsilon > 1$ in the following period, meaning the risk would not be completely reduced in the following period but just partly. This would result in the same model as we have now, however we would have to add the factor ϵ to several of the constraints in the model. In the identical searcher model the function $f(X_{.,c,t-1})$ should be replaced with (57). Furthermore the factor $r_{c,t+1}$ in constraints (29) and (30) needs to be multiplied with ϵ.

$$f(X_{.,c,t-1}) = 1 - \epsilon \sum_{c' \in \mathcal{C}} X_{c',c,t-1} \tag{57}$$

We could also choose a variable reduction depending on the amount of time that we spend in the cell. E.g. if a task group spends half a period in a cell, the risk will be reduced with 50% during the following period. In this case the reduction becomes dependent of the factor $(1 - d_{c,c'})$. To achieve this we need to change the same constraints as before, however instead of using ϵ we need to use $(1 - d_{c,c'})$. This change as well as the previous change do not affect the difficulty of the problem, since no extra variables nor constraints were added to the model. We only multiply with constants.

We will now look what happens if we choose a risk reduction that affects several periods. For example, if a task group visits a cell in period t, the risk will be reduced with 50% in period $t + 1$ and with 25% in period $t + 2$. Constraint (29) and (30) would become (58) and (59). The function $f(X_{.,c,t-1})$ should be replaced with (60). The problem remains linear after applying these changes and should not become much more difficult to solve.

$$Q_{c,c',t} \leq (r_{c,t}(1 - d_{c,c'}) + 0.5r_{c,t+1} + 0.25r_{c,t+2})X_{c,c',t} \qquad \forall c, c' \in \mathcal{C}, t \in \mathcal{T} \tag{58}$$

$$Q_{c,c',t} \leq R_{c,t}(1 - d_{c,c'}) + 0.5r_{c,t+1} + 0.25r_{c,t+2} \qquad \forall c, c' \in \mathcal{C}, t \in \mathcal{T} \tag{59}$$

$$f(X_{.,c,t-1}) = 1 - 0.5 \sum_{c' \in \mathcal{C}} X_{c',c,t-1} - 0.25 \sum_{c' \in \mathcal{C}} X_{c',c,t-2} \tag{60}$$

We can conclude that several changes in the risk reduction function are possible without extremely increasing the difficulty of the problem. We have only looked at the linear model since any change can be applied to the metaheuristics as they do not require a linear objective function. The changes in constraints that we mentioned were only for the identical searcher problem, but they are exactly the same for the different searcher problem.