OPTIMIZATION OF SENSOR USAGE IN NET-CENTRIC OPERATIONS

by

Thi Thanh Mai Ho

A thesis submitted in partial fulfilment of the requirements for the degree of Master of Science in Computing Science

UNIVERSITY OF GRONINGEN

Supervisors:

Prof. J.B.T.M. Roerdink
Internal Supervisor; Department of Mathematics and Computing Science, University of Groningen

Prof. M. Aiello
Internal Supervisor; Department of Mathematics and Computing Science, University of Groningen

Dr. ir. M.A.W. Houtsma
External Supervisor; Research Authority Innovation, Above Water Systems, Lead System Integration - Engineering, Thales Nederland

Chris van Gemeren
External Supervisor; Above Water Systems, Lead System Integration - Engineering, Thales Nederland

Final version thesis: June 19, 2009

THALES

university of groningen

Abstract

In the military, joint and combined operations are quite common. Such operations may involve a great number of platforms working together, which, in general, means an even larger number of available resources (e.g. sensors and weapons). Because of the large scale and the available resources residing on different platforms, it is difficult for the platform operators to keep a good overview of the situation and to determine the best way to make use of their available resources to fulfill the mission.

On behalf of the Royal Netherlands Navy, Thales Nederland is working on a Sensor Tasking Management (STM) system that is to manage all available sensors of the own force during an operation. Currently, a computer application is being developed that simulates this system. This thesis addresses the problem of finding the optimal sensor usage within the simulator system, where the notion of being optimal is actually not yet defined. Many different factors play a role here and there are basically infinitely many ways in which the available sensors can be deployed.

Multicriteria decision making, in particular the multi-attribute utility theory, is used to determine the sensor usage in the STM-simulator. To keep it simple, only two dimensions are taken into account in the implementation. The idea stays the same, though. With this, even in relatively simple situations it may take too long to find the optimal sensor deployment. It is possible, however, to obtain results that come quite close to being optimal within a significantly smaller amount of time.

Table of Contents

1 Introduction

Thales Nederland is working on a Sensor Tasking Management (STM) system for the Royal Navy of the Netherlands. The system is designed for controlling all available sensors during an operation. These sensors may be on different platforms (e.g. ships and missile sites) and, in general, a sensor's availability may change throughout the operation. Before actually engaging any of the sensors, the STM system needs to determine the desired sensor usage (i.e. which sensors to use and which modes and parameters, etcetera).

This chapter gives some background information on Thales, the Royal Netherlands Navy, the STM-system and optimal sensor usage.

The remainder of the thesis discusses optimization of the sensor usage within the given context, using multicriteria decision making, and an implementation of this within an existing simulation environment.

1.1 Thales

1.1.1 Thales Group

Thales is a world leader in mission-critical information systems for the aerospace, defence and security markets and generates more than 12 billion euro of revenues a year. The worldwide Thales Group employs 68.000 employees and is present in 50 countries all over the world. To strengthen its leadership position, Thales works with many different partners, such as the important international groups Alcatel-Lucent, DCNS, Raytheon and Diehl Aerospace.

Being a high-tech organisation, Thales cannot do without research and development. Research & Development at Thales, with its 22.000 high-level researchers, accounts for 20% of all revenues. There are about 30 ongoing cooperation agreements with universities and public research establishments in Europe, the United States and Asia. Furthermore, Thales owns a patent portfolio of 15.000 references and, annually, more than 250 patent applications are filed.

1.1.2 Thales Nederland

Thales Nederland is a part of the Thales Group and was founded in 1922 as 'N.V. Hazemeyers Fabriek van Signaalapparaten'. After the Second World War, the Dutch government identified the importance of a strong defence-industry and bought the plant. The company was renamed 'N.V. Hollandse Signaalapparaten' or in popular speech 'Signaal'. In 1956, the majority of the shares were sold to Philips and after this transaction the company grew enormously. At the end of the 1980's, Signaal employs over 5000 employees in multiple establishments all over the Netherlands. The cutback on defence budgets in the early 1990's resulted in major reorganizations at Signaal. Philips decided that the 'defence & control systems' were no longer a core

business and sold Signaal to the French Thomson-CSF. In 2001, Thomson-CSF was renamed to Thales and the plant in Hengelo became the head office of Thales Nederland B.V.

Thales Nederland currently employs about 2000 employees, of which more than 85% works in Hengelo. In 2007, Thales Nederland generated about 450 million euro worth of sales, 75% of which is export. Thales Nederland is the largest defence company in the Netherlands and is the main centre of excellence for Thales naval activities.

1.2 Royal Netherlands Navy

The Royal Netherlands Navy is one of the branches of the military of the Netherlands. The other branches consist of the Royal Netherlands Army, the Royal Netherlands Air Force and the Royal Military Police. These military forces are used to operate on behalf of the Dutch Ministry of Defense, that has the following main tasks:

- To defend the territory of the Kingdom of the Netherlands and its allies.
- To pursue international legal order and stability.
- To support civil authorities in law enforcement, emergency management and humanitarian aid, both nationally and internationally.

The navy contributes to these tasks by looking after safety at and from the sea, all over the world. The Royal Netherlands Navy employs over 11.000 people, led by the Royal Netherlands Navy Commander, and consists of three components:

- The Netherlands Maritime Force (NLMARFOR); the dispersible, operational staff, with as main location the navy headquarters in Den Helder.
- The vessels; different above water units (some of which carrying Thales equipment on board) and some submarines, also mainly located in Den Helder.
- The Netherlands Marine Corps; an elite expeditionary rapid reaction amphibious infantry force, trained to operate anywhere in the world under any (geological and climatological) condition.

For a mission, the Royal Netherlands Navy can work together with other (not necessarily Dutch) services of the armed forces. Some examples of cooperation between the navy and units from the army or the air force are naval gunfire support for ground-based units and air defense in coastal waters or above the sea.

1.3 Sensor Tasking Management (STM)

In the military, joint and combined operations (operations involving more than one armed service of the same nation and operations conducted by forces of two or more allied nations, respectively) are quite common. Such operations may involve a great number of platforms working together, which, in general, means an even larger number of available resources (e.g. sensors and weapons). Because of the large scale and the available resources residing on different platforms, it is difficult for the platform operators to keep a good overview of the situation and to determine the best way to make use of their available resources to fulfill the mission. Besides this, there

are some other reasons that call for a single system that controls all resources of the cooperating platforms:

- There are different types of resources, with a diversity of technical possibilities and different interfaces, that do not meet up to any standard.
- The platform resources are rather complex systems (possibly having different modes and/or parameters, for example), requiring their operator to possess a certain amount of fairly specific technical knowledge.
- Nowadays, platforms are manned less heavily and the average platform operator has less technical knowledge than before.
- The large number of factors (e.g. constraints, preferences, environmental factors) that are of influence on the best way to engage the available resources.

Thales Nederland is developing a Sensor Tasking Management (STM) system, that manages all available sensors during an operation. (In the future, it may be extended to include the management of other platform resources, but for now, only sensors are considered.) Since the complicated, technical information is processed and resides inside the system, this new system should be more efficient and easier to operate on. Taking the STM-system into use, will ease the sensor management considerably for system operators. The operator needs to give some system input; mainly, in the form of operational tasks for fulfilling the mission. The system determines internally which of the available sensors to use to execute the given tasks and how to employ them and then controls them as such. In doing this, the operational situation at hand, the possibilities and limitations of the sensors and all kinds of other constraints are taken into account.

Currently, there is just a simulation environment (based on Java, Java Expert System Shell and XML), which allows for simulating different time steps during an operation. Different platforms, sensors and operational tasks are encapsulated, as well as relevant factors, such as constraints.

1.4 Optimal Sensor Usage

Although a central STM-system may be easier and more efficient, it is not of very much use if it delivers poor results. But what is a good result? There are actually two questions here. First of all, we need to have a look at what the results of the system are. Secondly, there is the question of when a result can be considered to be a good result.

The sensor usage of the own forces can be seen as the result of the STM-system. Many different ways exist, in which the available sensors can be engaged. For one thing, there is the matter of which sensors to use. But then, for each sensor there is left to decide on the mode and functions to employ and the parameter settings to use, which include the regions over which these sensor functions are engaged.

Obviously, there may be a very large number of ways to use the available sensors. But how to tell between two cases which is to be preferred, if any? For that, we have to consider the main goal here, which is to fulfill the operational tasks. Hence, the level of usefulness of the sensor usage depends on the degree in which these tasks

are fulfilled. Things like region coverage and delivered performance could be considered now. Furthermore, the costs that come with the sensor usage might play a role. Besides financial costs, we could think of the time, power or effort it takes to deploy the available sensors in a certain way. However, without a specific method to rate different ways of sensor deployment, we cannot tell which way is better.

2 Problem Definition

As mentioned before, Thales is currently simulating a Sensor Tasking Management (STM) system that manages all available sensors during a military operation. Taking into account all relevant information and constraints, the system makes use of these sensors to execute operational tasks. This chapter defines the scope of our problem by giving a more detailed and rather technical description of the core of the situation and system, after which, the actual problem is stated.

A somewhat short description of the STM-simulator is given in Appendix A. The situation is as follows. The own force may consist of different platforms, each with their own (geographical) positions and sensors. Each sensor has its own set of technical sensor functions, which will be referred to as sensor capabilities in the remainder of this thesis. A sensor capability has a certain range, in which it can be used for a specified type of task and in a specified environment type, and its performance may differ per target type. See Figure 2-1 for an example. For each of these sensor capabilities to be able to contribute to the execution of given operational tasks, the specified task types need to match and the region corresponding with the operational task must, at least partly, be covered by the capability. Furthermore, all operational tasks come with a priority value and a specified target type. The desired performance values for different target types are included in the scenario.

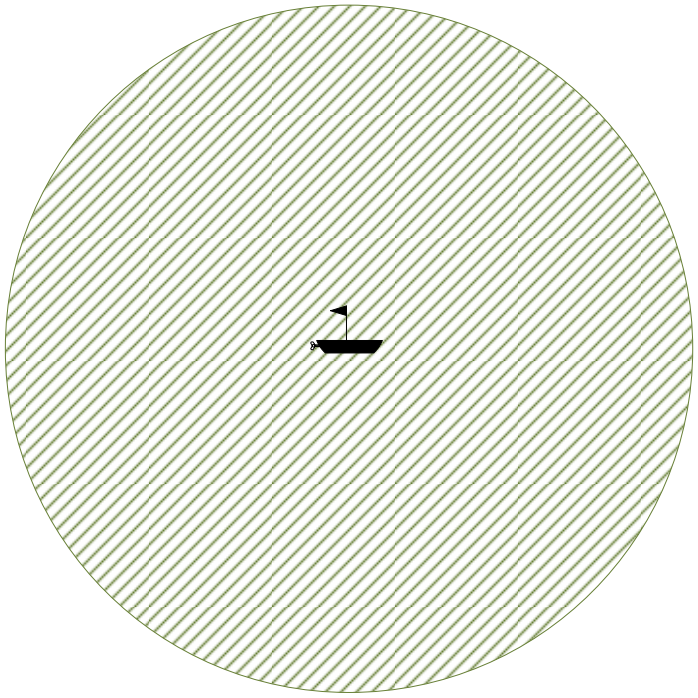


Figure 2-1: A schematic view, in 2D, of a platform and the (maximal) region that can be covered by a sensor capability that is associated with a sensor on the platform.

When the sensor deployment is being calculated, during a simulation step, all available sensors (and thus their capabilities) are considered, taking into account all possibilities and limitations within the scenario (see Appendix A for the information included in the scenario). In general, the sensor suite can be employed in many

different ways, meeting all constraints. The goal, however, is to make optimal use of the available sensors.

Let us now have a look at the process of determining the sensor usage. Starting out with all available sensor capabilities, the capabilities and parts of capabilities that cannot contribute (physically) to any of the given operational tasks are left out. This leaves the 'appropriate capabilities'. An example is given in Figure 2-2. In the same way, capabilities and parts of capabilities that do not satisfy the constraints are filtered out, leaving the 'allowable capabilities'. An overview of the different levels of sensor capabilities is given in Figure 2-3. Note that we are not just dealing with sensor capabilities here, but also with the regions over which they can be used.

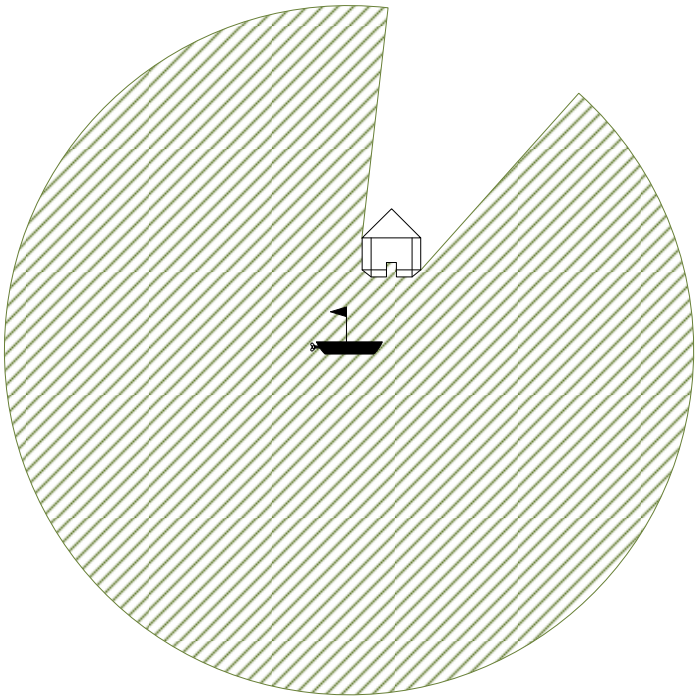


Figure 2-2: A schematic view of the region that can be covered by the sensor capability shown in Figure 2-1 with something in the way.

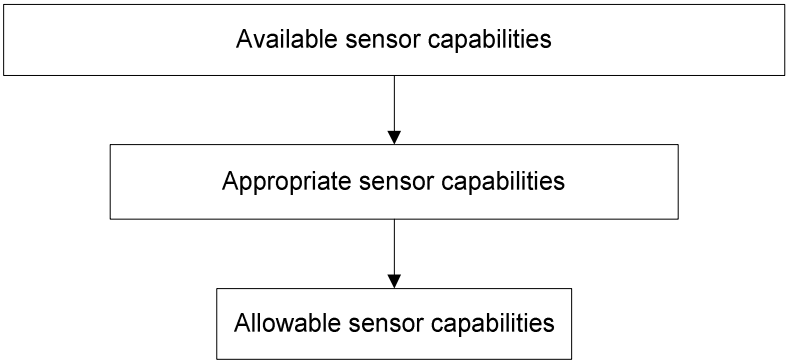


Figure 2-3: An overview of the different levels of sensor capabilities.

Just because each of the allowable sensor capabilities can be used separately over the region that is associated with it, does not mean that we should use it in that way or that it would be favourable. Anyway, in some situations it may even be totally

unnecessary to maximally use the allowable sensor capabilities, as shown in Figure 2-4. For one thing, the sensor capability only needs to be used in the direction of any operational tasks that it can contribute to (Figure 2-5). Moreover, it may not be necessary to use the sensor capability over its maximum range (Figure 2-6). On the other hand, the range over which a sensor capability can be used may not be as flexible such that it can be set to any value.

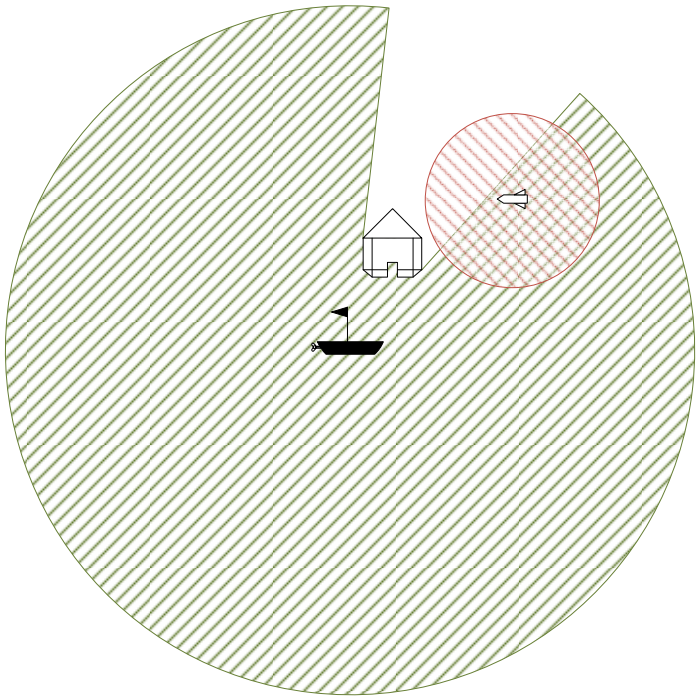


Figure 2-4: A schematic view of an operational task (given by the red shaded disk) in the situation of Figure 2-2.

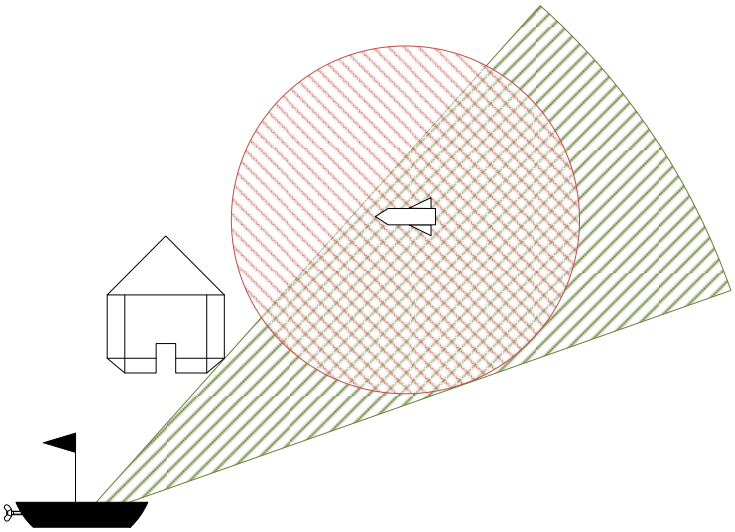


Figure 2-5: The maximum angle over which the sensor capability from Figure 2-4 can contribute to the given operational task.

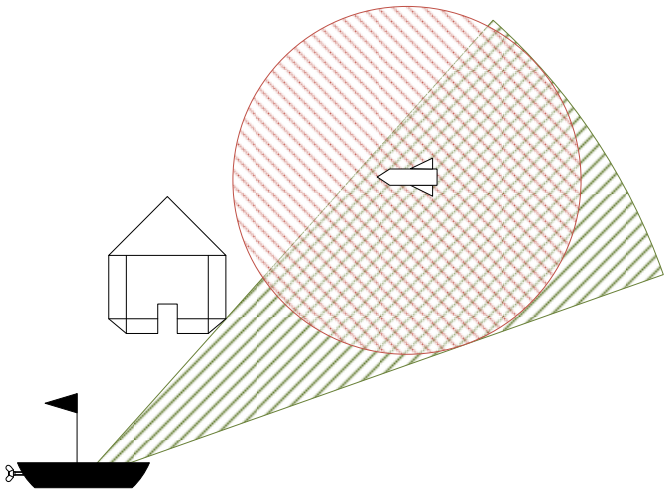


Figure 2-6: The largest region over which the sensor capability from Figure 2-4 needs to be used to maximally contribute to the given operational task.

Next, starting from the allowable sensor capabilities a 'capability list' is constructed for each operational task, see Figure 2-7. Such a capability list contains exactly the sensor capabilities that can be used for the associated operational task, together with parameters that define the largest region over which the sensor capability can and needs to be used to maximally contribute to the operational task.

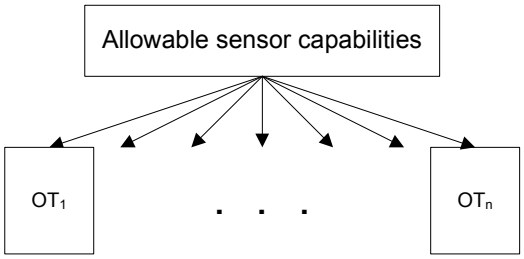


Figure 2-7: A schematic view of the relation between the allowable sensor capabilities and the capability lists for each operational task.

Problem definition

From the constructed capability lists, taking into account all constraints and other scenario data that still apply, a set of 'selected capabilities', giving shape to the chosen sensor deployment, needs to be determined. An algorithm needs to be found that selects an optimal set of sensor capabilities and parameters.

Although computing time is a critical issue during a real operation, there are no performance requirements for the simulator. It should, however, be easy to adjust and extend, on account of possible future developments.

3 Problem Analysis

In the STM-simulator, some preparation steps have been taken in the process of determining the (optimal) sensor usage, as described in the problem definition (see Chapter 2). An algorithm still needs to be found that determines the actual sensor usage, i.e. the sensor capabilities and parameters to use. This chapter lists and describes the most important factors in the sensor tasking management context and gives an idea of the complexity of the stated problem. Then the problem, including the starting point and the form of the desired output, is being viewed in terms of the newly introduced important factors. Finally, a discussion follows on which of these factors (still) need to be considered by the algorithm to be found.

3.1 Factors of Influence

Many different factors play a role in the process of determining the sensor deployment. The most important, are factors related to operational tasks, sensors and sensor capabilities and various constraints, see Table 3-1 to Table 3-4.

Table 3-1: Listing and description of some important properties of operational tasks.

Operational tasks	
Relevant factor	**Description**
Type	The type of the operational task.
Priority	The priority of the operational task.
Region	The region the operational task is concerned with, which is given by a (geographical) point and a radius.
Target type	The type of target the operational task is concerned with.

Table 3-2: Listing and description of some important characteristics of sensors.

Sensors	
Relevant factor	**Description**
Position	The (geographical) position of the sensor.
Orientation	The orientation of the sensor; depends on the orientation of the platform to which the sensor belongs.
Type	The type of the sensor.
Switching time	The time it takes for the sensor to switch from one mode to another.
Switching costs	The costs when the sensor is switched from one mode to another.

Sensor capabilities do belong to a sensor, but are not mentioned in Table 3-2. They are, in fact, the most important features of a sensor and are assigned their own table of characteristics, see Table 3-3.

Table 3-3: Listing and description of some important characteristics of sensor capabilities.

Sensor capabilities			
Relevant factor			**Description**
Operational task type			The type of operational task that the sensor capability can contribute to.
Parameters	Environment		The type of environment in which the sensor capability can be of use.
	Range		The distance over which the sensor capability can be used.
	Bearing		An interval included in [0°, 360°] that defines the directions in the horizontal plane in which the sensor capability can be used, where 0° represents the direction of the north pole and going clockwise (viewed from above) the angle is increased. The capability can also be employed over one or more non-overlapping subintervals.
	Elevation		An interval included in [0°, 90°] that defines the angles in the vertical plane over which the sensor capability can be used, where 0° and 90° represent the horizontal and upward directions, respectively. The capability can also be employed over one or more non-overlapping subintervals.
Quality of service	Target type		A type of target for which the sensor capability can be used.
	Performance	Detection	An indication of the distance at which the sensor capability can detect targets of the specified target type.
		Measurement	An indication of the quality of a single measurement, when the sensor capability is used on targets of the specified target type.
		Resolution	An indication of ability to distinguish targets of the specified target type when using the sensor capability.
		Update	An indication of the measurement rate when the sensor capability is used on targets of the specified target type.
Sensor load			The part of the total power of the sensor that is being consumed when the sensor capability is being used; depends on how the capability is employed.

Note that the range that is mentioned in Table 3-3 as one of the parameters of a sensor capability, has another meaning than the word "range" used earlier in this thesis. Furthermore, a sensor capability can have several 'quality of service'-elements, each of which is describing the performance delivered when the capability is being used on a target of the specified target type. Each sensor capability can have at most one 'quality of service'-element per target type. It is also possible for the quality of service to be defined for "any" target type, which occurs when the quality of service is not defined for all target types separately. When this is the case, this special 'quality of service'-element describes the performance on all target types that are not yet anticipated.

Table 3-4: Listing and description of different important constraints.

Constraints		
Relevant factor		**Description**
Platform-level	Blind sectors	Sectors where the sight of a sensor is obstructed, due to its placement on the platform.
	Intra-sensor constraints	Limitations of a sensor. (For example, exclusive mode selection.)
	Inter-sensor constraints	Limitations of sensors when used simultaneously. (To prevent interference, for example.)
Mission-level	Preference rules	Rules that define preferences, like the use of a certain sensor for a specified operational task type or target type or under certain climatic conditions.
	Emission control (EMCON)	Rules concerning radiation and to what extent it is allowed.
	Dynamic rules	Rules concerning to what extent it is allowed to radiate towards certain regions, other platforms and enemy forces.
In-action level	Objects in the vicinity	Objects in the vicinity that define regions where radiation is not desirable.
	Meteorological information	Meteorological information that can be of influence on the performance of sensor capabilities.
	Geographical information	Geographical elements that may be in the way; obstructing the view and/or (partly) hindering sensors from executing certain tasks.

3.2 Complexity of the Problem

The important factors that were just described relate to the total process of determining the sensor usage, not just the part that is left to deal with. The following shows what it is that makes the whole problem complicated, demanding for a particular method and for the preparation steps to be performed.

To fulfill the given operational tasks, the easiest solution would be just to use all available sensor capabilities maximally. However, setting a sensor to a specific mode

to allow for the use of certain capabilities, automatically means excluding the sensor capabilities of other modes. Moreover, in general, using (different) sensor capabilities to their maximum may not be desirable, on account of interference of the signals, or even allowed, due to emission control (EMCON) or dynamic rules. A more realistic and permissible solution is to deploy the most 'useful' sensor capability, over the region over which it can and may be used and where it contributes to the execution of the given operational tasks, and repeat this with the remaining capabilities until there are no 'useful' capabilities left. (The term 'useful' that is used here relates to the extent to which a sensor capability can and may contribute to the set of operational tasks.)

The next example, however, shows that this algorithm does not guarantee optimal sensor usage. Suppose there are two operational tasks to be fulfilled and two sensor capabilities that can be used for these tasks, as in Figure 3-1. The available sensor capabilities can be used at the same time, but not over the same region. When the choice is between both capabilities as shown in Figure 3-1, capability A is preferred over capability B, which would (when using the suggested algorithm) result in the use of capability A only. But, in the case that capability B performs really well and the performance of capability A is not too high, the sensor usage in Figure 3-2 may better. Operational task 2 could be so much more important than operational task 1, that a better performance over operational task 2 would be preferred to covering the total region of operational task 1.

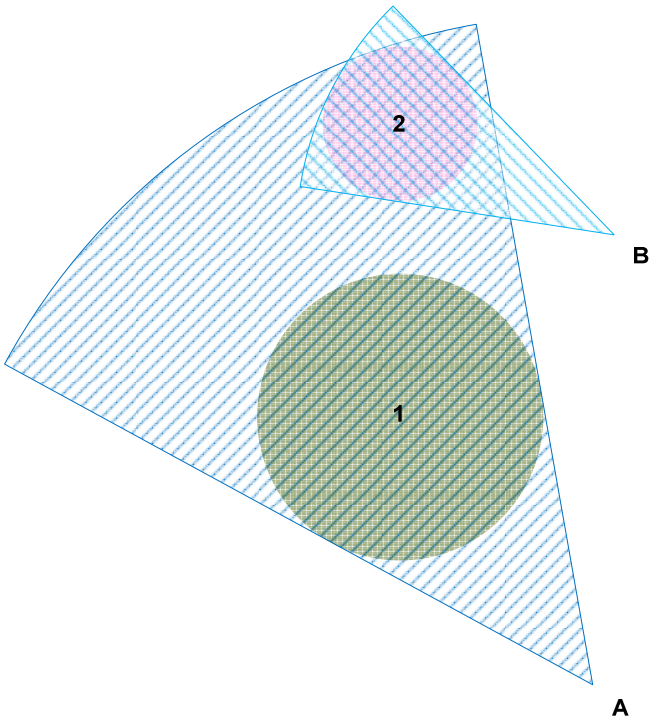


Figure 3-1: The green and pink circle represent the regions of operational tasks 1 and 2, in 2D, and the sectors show how sensor capabilities A and B can and may be used separately for the given tasks.

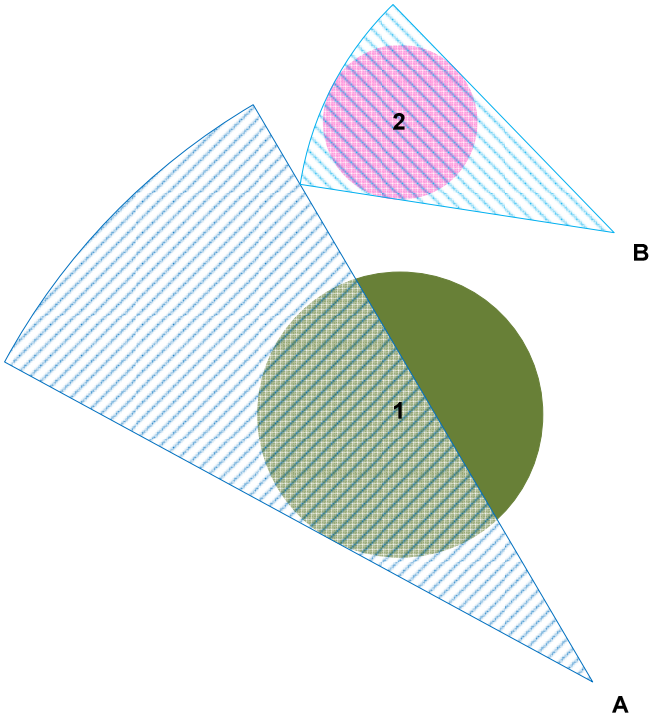


Figure 3-2: The sectors show a possible way that sensor capabilities A and B from Figure 3-1 can be used for operational tasks 1 and 2 simultaneously.

The discussion above shows that by needlessly deploying sensor capabilities, meaningful use of other capabilities can be hindered. Taking the preparation steps in the sensor deployment determination process, as described in Chapter 2, prevents this from happening, as it rules out noncontributing use of sensor capabilities. Furthermore, we can conclude that it takes more than just a simple algorithm to obtain the optimal sensor deployment.

3.3 Problem Situation

Let us have a better look at the problem situation. From now on, only the part of the sensor deployment determination process after the preparation steps will be considered. The situation is as follows.

There is a number of operational tasks that needs to be executed and for each of these tasks we have a capability list. Note that since the lists are constructed from allowable sensor capabilities, the sensor capabilities can each be used separately, with the accompanying parameters. The set of parameters here consists of range, bearing and elevation parameters, as described in Table 3-3. The largest region over which a sensor capability can and needs to be used to maximally contribute to an operational task cannot always be represented by just one set of those parameters, though. Therefore, a sensor capability can have several occurrences in a capability list, each with a different set of parameters. Furthermore, a sensor capability may be in different capability lists, possibly accompanied by sets of parameters representing overlapping regions.

The goal is to determine a set of sensor capabilities accompanied by parameter sets, that represents the optimal sensor usage in the (static) operational situation at hand. Obviously, the chosen sensor usage should be consistent with the provided capability lists. That is, a sensor capability can only be used over a region that is in the union of the regions that are represented by parameter sets in the capability lists that are associated with the concerning sensor capability. Not all regions that can be represented by the parameters are appropriate, though. It is possible to use sensor capabilities over all kinds of different intervals in bearing and elevation, but as far as range is concerned, not any value is possible. In general a sensor capability can only be used over the range that is specified for it, and in some cases some other (specified) range values are possible as well. Actually, no possible range values apart from the maximal range are specified in the simulator system, yet we may come across different range values in the capability lists provided to us. These values will therefore be considered as (additional) possible range values for the sensor capability they were associated with. Furthermore, not all combinations of sensor capabilities are allowed, due to intra- and inter-sensor constraints.

In order to obtain the desired set of sensor capabilities, first, some kind of definition needs to be established for optimal sensor usage. The main issue here is the degree in which the operational tasks are fulfilled. This, of course, depends on the degree in which the separate tasks are fulfilled, which depends on the delivered performance and region coverage. Another issue is efficient sensor usage, which concerns things like effort and costs. When it is clear what the exact criteria for optimal sensor usage are, a solution can be constructed that meets these criteria, or different potential solutions can be tested against the criteria to find the best solution.

3.4 The Important Factors to Consider

Not all of the important factors listed in Table 3-1 to Table 3-4 are present in the existing simulator system (yet). Furthermore, some factors are already processed in precalculation steps (determining the capability lists we are starting out with). Hence, only a part of the factors needs to be considered from here on. For each of the tables above, a discussion follows of which factors can be left out and which still play a role in the process.

Since the relevant sensor capabilities are already checked for whether they can be used for the operational tasks, the type of a task, from Table 3-1, does not need to be considered anymore. Priority, region and target type, however, are needed for determining how well a sensor deployment option fulfills the assigned operational tasks.

A sensor's position, see Table 3-2, is needed for determining to what extent the corresponding sensor capabilities can cover regions of operational tasks. The remaining sensor characteristics, on the other hand, do not have to be taken into account. The orientation of the sensor is already processed in the bearing of its capabilities. Furthermore, the type of the sensor is not being considered for anything at the moment and the switching time and costs are not even implemented in the system yet.

The operational task type from Table 3-3, which was only needed for the check with the type from Table 3-1, can now be left out. The same goes for the sensor load that comes with a sensor capability, which is not implemented in the system yet, and the environment, which was already considered for the determination of the appropriate sensor capabilities. As for the rest of the parameters, the original parameter values that come with the sensor capability have been processed and are not important anymore. The only sensor capability characteristics that still matter are the 'quality of service'-elements, prescribing the delivered performance for the different target types.

Regarding the constraints, blind sectors and EMCON are already taken into account and can be left out of further consideration. The remaining constraints listed in Table 3-4 are barely present in the simulator or even not at all, at the time being, and can also more or less be ignored for now.

Eventually, it comes down to it that starting out with the input capability lists and the desired performance prescribed for every target type (which defines the desired performance for the operational tasks), the following factors can and have to be taken into account in order to achieve optimal sensor usage:

- The priority of the operational tasks.
- The region corresponding with the operational tasks.
- The target type corresponding with the operational tasks.
- The position of the sensors.
- The quality of service of the sensor capabilities.

4 Possible Approaches

From the problem analysis (Chapter 3) it follows that an algorithm needs to be found that, starting out from the given capability lists and taking into account certain factors, determines a set of sensor capabilities and accompanying parameters, representing the (not yet defined) optimal sensor usage. This chapter first discusses which classes of methods could be considered for our problem, after which, they will be reflected upon and judged on their appropriateness. Finally, a suitable approach will be chosen to be used.

4.1 Methods to Consider

When it comes to finding the optimal sensor usage, different kinds of methods can be considered. Since there is a number of sensor capabilities that can be used, we could think in terms of resource allocation. Furthermore, the goal is to make optimal use of these resources, which could ask for some optimization algorithm. Cost functions could be used, or scores could be determined for different sensor deployment possibilities. Moreover, different techniques from normative decision theory could be applied. With the different factors that need to be taken into account, one could also think of rule-based systems or multicriteria decision making. Of course, it is also possible for different kind of techniques to be combined.

4.2 Suitability of Different Methods

The different kind of methods that were just mentioned may not all be equally suited (or even suited at all) for the considered problem. A discussion on the applicability of the different approaches follows.

4.2.1 Resource Allocation

With the resource allocation approach, the available sensor capabilities, with their accompanying parameters, can be seen as resources. Normally, resources are assigned to requests, which might suggest that the sensor capabilities should be allocated to operational tasks. But when a capability is used for a certain task, it does not rule out that the capability contributes to the execution of another task at that same moment. This means that the capability is not really allocated to the one task. A sensor capability can, however, be allocated to a part of the region in which everything takes place. This region of interest, of course, contains the joint region of all operational tasks. When allocating sensor capabilities to regions like this, it may also be possible for some capabilities to be assigned to overlapping regions.

A resource allocation procedure that might be suitable here, is the combinatorial scoring auction [1]. With this procedure, the situation is as follows. Buyers want to procure several items at once and the suppliers offer different quality levels for subsets of items. A scoring rule is used to determine which supplier(s) may take care

of which items. When applying the method for our problem, the different items altogether should enclose the joint region of all operational tasks, while the suppliers relate with possible combinations of sensor capabilities. The scoring rule, which is based on the performance delivered by the different combinations of sensor capabilities over the corresponding regions, is now used to determine which combinations of sensor capabilities should be deployed over which regions. But due to intra-sensor constraints, in general, not every supplier may be selected together with every other supplier, even if they do not have any items in common.

It was made clear in Section 3.2 that using the available sensor capabilities with the given parameters to construct (potential) solutions, will, in general, not give the desired result. Since the combinatorial scoring auction as was just described produces results in this exact way, there is still room for improvement. Section 3.2 also showed that a better result can be achieved, when there is the option to deploy the provided sensor capabilities over only part of the total region that they can cover. The described combinatorial scoring auction procedure can quite easily be adjusted to this by splitting up the regions of the sensor capabilities beforehand, allowing for combinations with the same capability over different regions. Though, there is still the issue of how the region(s) should be split up, in order to obtain the optimal solution.

4.2.2 Optimization

Optimization deals with finding the best possible value(s) of some objective function by systematically choosing arguments from within an allowed set. Since costs are not taken into account in our problem, we can try to maximize some kind of score function, or the result of a scoring rule, that considers the performance of the sensor capabilities associated with the function input. The (potential) solutions to our problem, namely sets of sensor capabilities accompanied by parameters specifying the regions over which they should be deployed, can be taken to be these input arguments. Only a finite number of combinations of the provided sensor capabilities can be made, but each of the capabilities can be deployed over an indefinite number of different regions of indefinite size. This makes our solution space infinitely large and far from discrete. Therefore, it is not possible simply to apply exhaustive search. Moreover, because of the complexity of the search space, even with simulated annealing it is not very likely that a good solution can be found within a reasonable amount of time. And when even simulated annealing does not work, obviously, other optimization methods will not be very effective either.

Limiting the search space by discretizing it significantly simplifies the problem, making it much more suitable to be solved with an optimization method. This can be done, for example, by allowing only a finite number of predefined regions for each sensor capability, over which it can be deployed. The discretization can be done in many different ways. Note, that discretizing our solutions automatically leads to a finite solution space. Like in Subsection 4.2.1, here the quality of the final solution also strongly depends on the choice of regions that are permitted for the sensor capabilities.

An optimization method that slightly relates to our problem is fuzzy optimization with weighted constraints [2]. With this method, preferences for different constraints and goals can be specified by the user and weight factors are used to influence the

nature of trade-off between these various constraints and goals. Currently, we are not really concerned with constraints, but the priorities of the different operational tasks most certainly do express preferences for our goals. Nevertheless, since there are no flexible or soft constraints, there is no need for a fuzzy method.

4.2.3 Normative Decision Making Techniques

Normative decision theory is concerned with identifying the best decision to take. The decision maker is assumed to be fully informed and rational, and able to compute with perfect accuracy. This almost perfectly describes our problem situation. In our case, the best decision is to make optimal use of the available sensors. The decision maker is a (computer) system, that is assumed to be fully informed. Furthermore, it is rational and it can compute with very high accuracy.

Table 4-1 presents different techniques for normative decision making, described by Verhoeff [3], that can be used to choose a solution from a set of potential solutions. These techniques all use different ways, not all equally suitable for our problem, to select one of the options, which may lead to very different solutions. Note that in order to apply normative decision making techniques, like with the approaches discussed earlier, a finite solution space is required. What is left, is the choice of attributes or features of interest, attribute utilities, weights and/or critical levels or criteria, depending on which of the techniques is chosen to be used.

4.2.4 Rule-based Systems

Rule-based systems are used to store and manipulate knowledge. The rules of the system can be used to make simple choices or to deduct useful information, possibly to help making choices. In the STM-simulator application, rules were applied during the preparation steps of the sensor deployment determination process, leaving the appropriate information from which the desired sensor deployment can be chosen. However, it concerns a rather complicated choice to be made, while the possibilities of rule-based systems are limited, especially when it comes to making choices. Therefore, in case of our problem, it would probably be best to look for a more suitable approach.

4.2.5 Multicriteria Decision Making

Multicriteria decision making (MCDM) offers a transparent procedure for dealing with decision making in difficult cases with conflicting interests. During the decision making process, a number of alternatives are identified, as well as some aspects or criteria, on which they are ranked. The procedure finally leads to the best alternative, in accordance with the choices made along the way.

Table 4-1: Listing and description of some normative decision making techniques.

Technique	Description
Addition of Utilities	The utilities of each option's attributes are summed and the option for which the sum is greatest is chosen.
Addition of Utility Differences	For each attribute of interest, the difference between the utility for the attribute for one option and the utility for the same attribute for another option is computed. Then the weighted differences are summed and the option that the sum indicates to have the higher overall relative utility is chosen.
Conjunction	The option that reaches some critical level on all desired attributes is chosen.
Disjunction	The option that reaches some critical level on one or more desired attributes is chosen.
Elimination by Aspects	An (important) attribute is selected and any option that fails to meet some preset critical level for that attribute is eliminated. Repeat the process using the next attribute, and the next, etc., each time eliminating lesser options until a single option remains.
Lexicographic	The option that is best on the most important attribute is chosen. If two or more options are equal on that attribute, the next most important attribute is checked, etc., each time eliminating lesser options until a single option remains.
Number of Superior Features	For two competing options, note which one is superior on each feature of interest. The option that has the largest number of 'superior to' classifications is chosen.
Single Feature Inferiority	For two competing options, the option that is inferior on one feature of interest is eliminated, regardless of the other features.
Single Feature Superiority	The option that is superior on one feature of interest is chosen, regardless of the other features.
Single Feature Difference	Find the feature on which the options differ most, the option that is best on this feature is chosen, regardless of the other features.
Satisficing	The first option that meets or exceeds the minimal criteria for some set of features is chosen.
Satisficing-plus	Options are evaluated on critical features, eliminating all options that do not meet the criteria. Then the cut-offs for the features are changed and the process is repeated; the surviving options are evaluated until a single option remains.

The problem we are dealing with concerns the determination of the sensor deployment that satisfies the given operational tasks best. What makes it complicated, is the fact that improving on one task, may automatically mean decreasing the satisfaction of another task. Using MCDM, the different potential solutions to our problem, namely combinations of sensor capabilities accompanied by regions over which to deploy them, can be seen as the alternatives. However, since a solution is found by evaluating these alternatives, at least a discreet solution space is needed, like with the optimization methods. Furthermore, the operational tasks can be considered as the decision criteria. As for how to judge the tasks on their level of satisfaction, that is a question apart. But again, it concerns an issue with conflicting interests.

Different methods, also varying in the level of freedom for the decision maker, are available for MCDM. A well known, frequently used and rather simple method is the simple multi-attribute rating technique (SMART). With SMART [4], the techniques to be used during the different MCDM procedure steps are specified in quite some detail, leaving only the assessment of attributes (criteria) and alternatives with respect to these attributes to the user. More general is the multi-attribute utility theory (MAUT) [4] approach. MAUT involves assigning (relative) weights to the attributes, which are taken into account in the final evaluation of the alternatives. Yager [5] deals with something similar, namely prioritized MCDM, which is based on prioritization of the decision criteria. This is enforced by giving lower priority criteria weights that are related to the satisfaction of higher priority criteria. In our case, prioritization of criteria is also present: the decision criteria, or operational tasks, are each accompanied by a priority value. However, since not much is given on how to handle task priorities, making the weights alternative-dependent, as is done by Yager, makes it unnecessarily complicated for now.

An example of MCDM applied in a case study is given by Espie et al. [6], who applied it to electricity distribution system planning. With the software system they used, it took about an hour on a PC with an ‘up-to-date’ specification to evaluate all 1728 alternatives (on 6 criteria). Obviously, in an operational situation as we are dealing with (the situations that ask for a STM-system are often more demanding than the described electricity distribution system planning case), having to wait an hour for the system to calculate how to deploy the available sensors is highly undesirable. Anyway, the weight values to be used in the analysis were identified with the help of planning engineers and other stakeholders. Espie et al. mention that different techniques exist for eliciting weight values from individuals, teams and organisations. In case of the STM-system, these techniques may also be of interest.

Something else that might be of interest when it concerns finding the best solution, is interaction. Human interaction can lead to a solution that meets the wishes and expectations of the user more. Wang et al. [7] introduce an iterative method for interactive multicriteria decision making, requiring a reference value to be specified during each iteration step. Although we are not really interested in an interactive method at the moment, such a method could be used to improve the performance of the STM-system in the future.

4.3 Chosen Approach

After having discussed possible approaches to our problem and considering their applicability, it is time to choose the approach to be used in the STM-simulator.

With the suitable approaches we have seen that a discrete solution space, which in our case also means a finite solution space, is required in order to obtain a useful result (or, with some of the methods, even a result at all). Furthermore, with each of these approaches, different potential solutions are evaluated, in a way, to find the desired solution. MCDM, in particular the MAUT-procedure, comprises clear steps that help developing a procedure to evaluate the alternatives, with the possibility to give appropriate weight values to each of the decision criteria. Because of this, MAUT is chosen for handling our problem.

A downside to the use of MCDM, as just mentioned in Subsection 4.2.5, is the amount of time it takes to compute the result. With the real (still to be developed) STM-system, we would not want to wait an hour before taking action when, for instance, a missile is coming our way or when an enemy may be nearby. In case of the STM-simulator, which we are dealing with, however, there are no particular requirements as for that, so the computing time is not a very big issue.

Another thing that we have seen with the suitable approaches, is that with each of them the best of all considered potential solutions is chosen or (the first) one that is good enough. In fact, not each method requires all potential solutions to be considered or for the chosen solution to be the best solution possible. Not requiring our MCDM-procedure to evaluate all alternatives, could save time, possibly even making the chosen approach significantly more practical to be used in a real-world situation. However, we need to keep in mind that there is no use in pursuing this when it decreases the quality of the final result too much. The question here is whether the alternatives to be considered can be chosen in some way that guarantees to include a solution that is good enough.

5 Method

After considering different approaches in the previous chapter, multicriteria decision making (MCDM), in particular the multi-attribute utility theory (MAUT) procedure, has been chosen to be used for determining the optimal sensor usage. Starting out from the given capability lists, a set of sensor capabilities and accompanying parameters needs to be chosen. But as it turned out (see Chapter 4), this (as any other non-discrete) search space is too large to handle. The MAUT-procedure will be used to select a solution from a predetermined, finite set of alternatives.

This chapter gives a general description of MCDM and MAUT, including an overview of the procedure steps, after which these steps will be worked out in detail for our problem. The latter includes, among other things, a discussion of which solution space to consider and the issue of what the optimal sensor usage is.

5.1 Multicriteria Decision Making and Multi-attribute Utility Theory

Multicriteria decision making (MCDM) can be used for problems for which the following applies. Different solutions to the problem exist and the goal is to choose the best solution possible. However, the best solution is not unambiguously defined, it depends on different factors. MCDM consists of four main elements [8]:

1. *The Set of Alternatives*, denoted by X, with its generic element denoted by x, from which we will choose our decision.
2. *The Set of Criteria*, denoted by $f = (f_1, \ldots, f_q)$, with which we are concerned for making a good decision.
3. *The Outcome of Each Choice*, $f(x) = (f_1(x), \ldots, f_q(x))$, measured in terms of the criteria, will also be important for consideration. The totality of all possible outcomes will be denoted by $Y = \{f(x)|x \in X\}$, with y as its generic element.
4. *The Preference Structures of the Decision Maker* will be another important element of a multiple-criteria decision problem. If the preferences over the possible outcomes Y are clearly and perfectly specified, then the decision problem can become easy, for if y* is the best outcome in Y, then some $x^* \in f^{-1}(y^*)$ will be the choice.

Multi-attribute utility theory (MAUT) is a MCDM-method that assigns weights to each of the decision criteria and aggregates these weights and the criteria outcomes. All MAUT-procedures include the following five steps:

1. Define alternatives and value-relevant attributes (criteria).
2. Evaluate each alternative separately on each attribute.
3. Assign relative weights to the attributes.
4. Aggregate the weights of the attributes and the single-attribute evaluations of alternatives to obtain an overall evaluation of alternatives.
5. Perform sensitivity analyses and make recommendations.

5.2 Optimization of Sensor Usage with MAUT

In our case, the problem situation is as follows. The problem concerns the execution of the given operational tasks and all possible ways in which the available sensor capabilities can be deployed, together make up the solution space. Eventually, the best solution depends on satisfaction of each of the individual operational tasks.

Next, the MAUT-procedure steps will be worked out in detail for our problem. Since the determination of a good set of alternatives requires some discussion and the definition of the decision criteria is a completely different issue, these will be handled separately. Furthermore, we will have a look at the criteria weights before going into the evaluation of alternatives on these criteria. Finally, instead of sensitivity analyses and recommendations, the last step is actually just about choosing the final solution. Figure 5-1 gives an overview of the steps that are handled in this section.

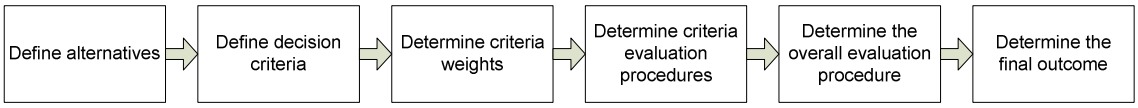


Figure 5-1: Flow chart with the steps to be taken in order to use the MAUT-procedure.

5.2.1 The Set of Alternatives

For determining the sensor usage, we start out with the provided capability lists. Note that when we totally forget about intra- and inter-sensor constraints, there is no need for MAUT or any other procedure for determining the optimal sensor usage, since we could just use all sensor capabilities from the capability lists over the provided regions to obtain the best possible satisfaction of the given operational tasks. As that takes away the whole problem, we need to keep these constraints in mind and assume that, in general, we cannot use the provided sensor capabilities as such. This means that different possible ways of sensor deployment need to be considered. Therefore, as a first step (see Figure 5-2) we determine alternatives that represent possible ways of sensor deployment.

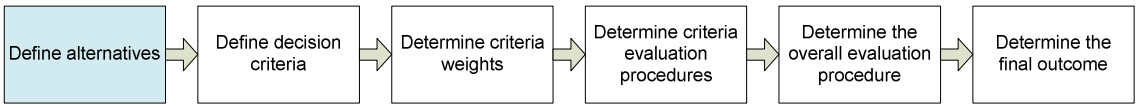


Figure 5-2: First step of the flow chart of Figure 5-1.

For determining the alternatives, let us recall the given capability lists and how to continue from there (Chapter 2 and Section 3.3). For each of the operational tasks, we have a list of allowable sensor capabilities, each accompanied by a set of range, bearing and elevation parameters. The capabilities can each be used separately, with the accompanying parameters, all possibilities and limitations taken into account. In fact, the specified parameters describe to what extent the capability can be employed usefully for the corresponding task. Furthermore, a sensor capability may be in different lists, possibly accompanied by different sets of parameters, that may represent overlapping regions. A set of sensor capabilities needs to be selected from the provided capability lists, and each of the selected capabilities should come

with one or more sets of parameters, representing non-overlapping regions within the total region that is given for the capability in the different lists. The range value(s) for a sensor capability, however, can only be chosen from a (possibly very limited) set of values. From this, we can conclude that our alternatives should represent such sets of sensor capabilities accompanied by sets of parameters.

The next example should give a better insight into capability lists and alternatives. Suppose operational tasks 1 and 2 have been given and sensor capabilities A and B, both belonging to the same sensor, can be used for these tasks. The capability lists associated with the operational tasks are given by Table 5-1 and Table 5-2 and describe to what extent each of the sensor capabilities can be deployed usefully for each of the given tasks. The regions corresponding with these capability lists are shown in Figure 5-3 and Figure 5-4 together with the regions of the operational tasks. To keep it simple, the intervals for range and elevation are kept constant throughout the whole example. Two possible alternatives are given in Table 5-3 and Table 5-4, see also Figure 5-5 and Figure 5-6. Notice that not all of the bearing intervals of the sensor capabilities of the alternatives match or are included in the corresponding intervals in the capability lists. The bearing interval [30°, 70°] for sensor capability A is allowed, however, since it corresponds with a region that is included in the total region, over both tasks, for sensor capability A. Furthermore, if it was given that sensor capabilities A and B could not be used at the same time, then neither of these alternatives would be appropriate. In case the sensor capabilities could be used simultaneously but not over the same region, only alternative II would be appropriate.

Seeing how alternatives can be obtained, an infinite number of different alternatives is possible. Keeping in mind that we wish to have optimal sensor usage, we can say that the gross of the possible alternatives is not very interesting. For instance, have a look at alternative I, from Table 5-3, where the subinterval [3°, 6°] of [0°, 20°] is chosen. Note that infinite other subintervals like that, and thus alternatives, are possible. None of these are really interesting, since the use of the whole interval [0°, 20°] is always more useful.

The following, more elaborate examples give an idea of how to choose the alternatives that are worth considering. Suppose that there is one operational task to fulfill, which is depicted in Figure 5-7 together with the sensor capabilities from the capability lists. (The parameters specifying the use of the sensor capabilities over the third dimension, that is not depicted here, are assumed to be the same throughout the next examples.) Then there is no use in just partially using sensor capability A over region 1 (see the example from the previous paragraph). Furthermore, there is no use whatsoever in partially using either of the sensor capabilities over region 2b. For in case sensor capabilities A and B cannot be used simultaneously over the same region, we would want to use the one with the better performance over the entire region 2b. (Note, that using sensor capability B over region 2b automatically means using it over region 2a as well.) Partial use of either of the capabilities or both could at best lead to the same level of satisfaction here (in case the performance of neither of the capabilities is preferred over the other). When the sensor capabilities can be used simultaneously over the same region, we would at least want to use the sensor capability with the best performance, and also the other capability in case it can contribute anything in the delivered performance, over entire region 2b. Here

again, by partially using either of the capabilities or both, we can at best obtain the same level of performance.

Table 5-1: List of sensor capabilities accompanied by parameters with which they can be deployed usefully for operational task 1.

Operational task 1	
Sensor capability A	Range: 50000 Bearing: [0, 20] Elevation: [20, 40]
Sensor capability A	Range: 50000 Bearing: [30, 50] Elevation: [20, 40]

Table 5-2: List of sensor capabilities accompanied by parameters with which they can be deployed usefully for operational task 2.

Operational task 2	
Sensor capability A	Range: 50000 Bearing: [40, 70] Elevation: [20, 40]
Sensor capability B	Range: 50000 Bearing: [40, 90] Elevation: [20, 40]

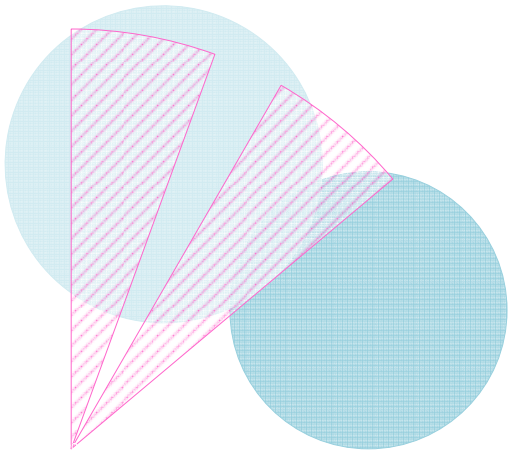


Figure 5-3: The regions corresponding with the capability list from Table 5-1 (in pink) together with the regions corresponding with operational tasks 1 and 2 (on the left and right, respectively), in 2D.

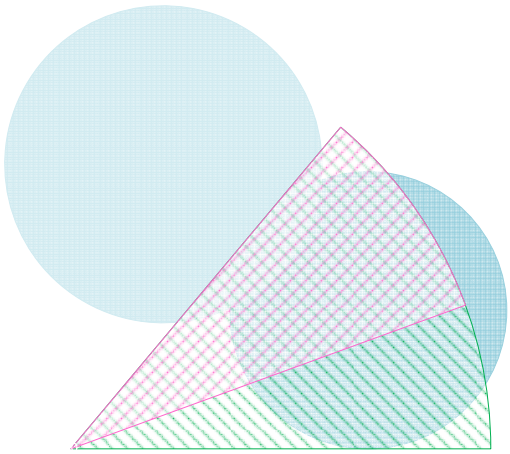


Figure 5-4: The regions corresponding with the capability list from Table 5-2 (in pink and green for sensor capabilities A and B, respectively) together with the regions corresponding with operational tasks 1 and 2 (on the left and right, respectively), in 2D.

Table 5-3: List of sensor capabilities accompanied by parameters that describes the sensor usage of alternative I, derived from the capability lists given by Table 5-1 and Table 5-2.

Alternative I	
Sensor capability A	Range: 50000 Bearing: [3, 6] Elevation: [20, 40]
Sensor capability A	Range: 50000 Bearing: [30, 70] Elevation: [20, 40]
Sensor capability B	Range: 50000 Bearing: [40, 90] Elevation: [20, 40]

Table 5-4: List of sensor capabilities accompanied by parameters that describes the sensor usage of alternative II, derived from the capability lists given by Table 5-1 and Table 5-2.

Alternative II	
Sensor capability A	Range: 50000 Bearing: [0, 20] Elevation: [20, 40]
Sensor capability A	Range: 50000 Bearing: [30, 70] Elevation: [20, 40]
Sensor capability B	Range: 50000 Bearing: [70, 90] Elevation: [20, 40]

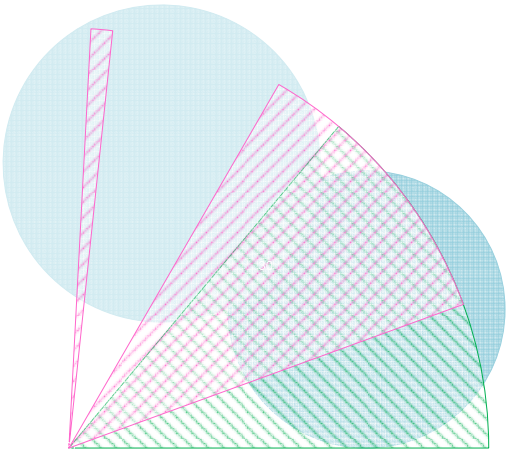


Figure 5-5: The regions corresponding with Alternative I from Table 5-3 (in pink and green for sensor capabilities A and B, respectively) together with the regions corresponding with operational tasks 1 and 2 (on the left and right, respectively), in 2D.

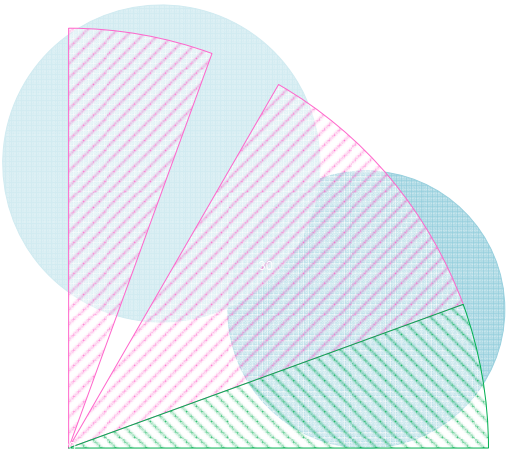


Figure 5-6: The regions corresponding with Alternative II from Table 5-4 (in pink and green for sensor capabilities A and B, respectively) together with the regions corresponding with operational tasks 1 and 2 (on the left and right, respectively), in 2D.

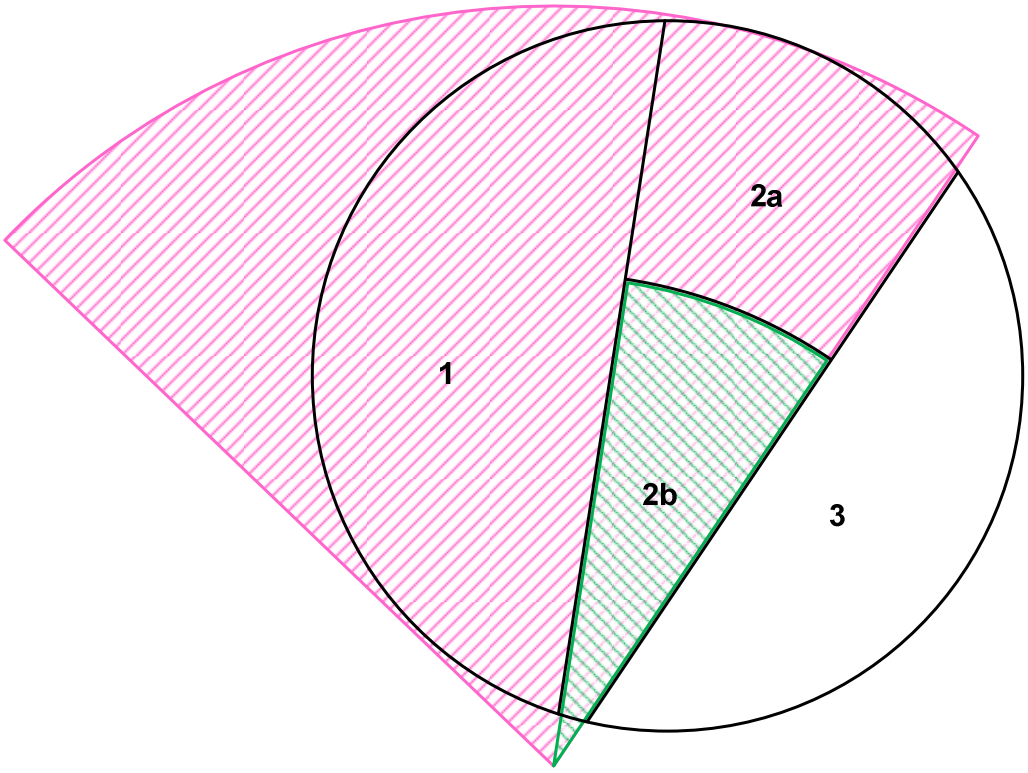


Figure 5-7: The circle represents the region of an operational task, in 2D, and the pink and green sectors show how sensor capabilities A and B, respectively, can be used for the operational task. Based on intervals in bearing (or elevation) over which different combinations of sensor capabilities can be used, the region of the operational task is partitioned into three regions: region 1, the union of 2a and 2b and region 3 (outlined in black). Moreover, each of the resulting regions is partitioned into regions over which different combinations of sensor capabilities can be used (which results in the second region being split up in regions 2a and 2b).

Now suppose that in the situation just described we would come across the possible use of capability A over only part of region 2a and 2b in the capability lists, as shown in Figure 5-8. That would mean that another task is involved that only requires capability A to be deployed over this region. In this case, depending on the performances of both sensor capabilities for the operational tasks, there may be a reason for the sensor capabilities to be deployed over only part of region 2b.

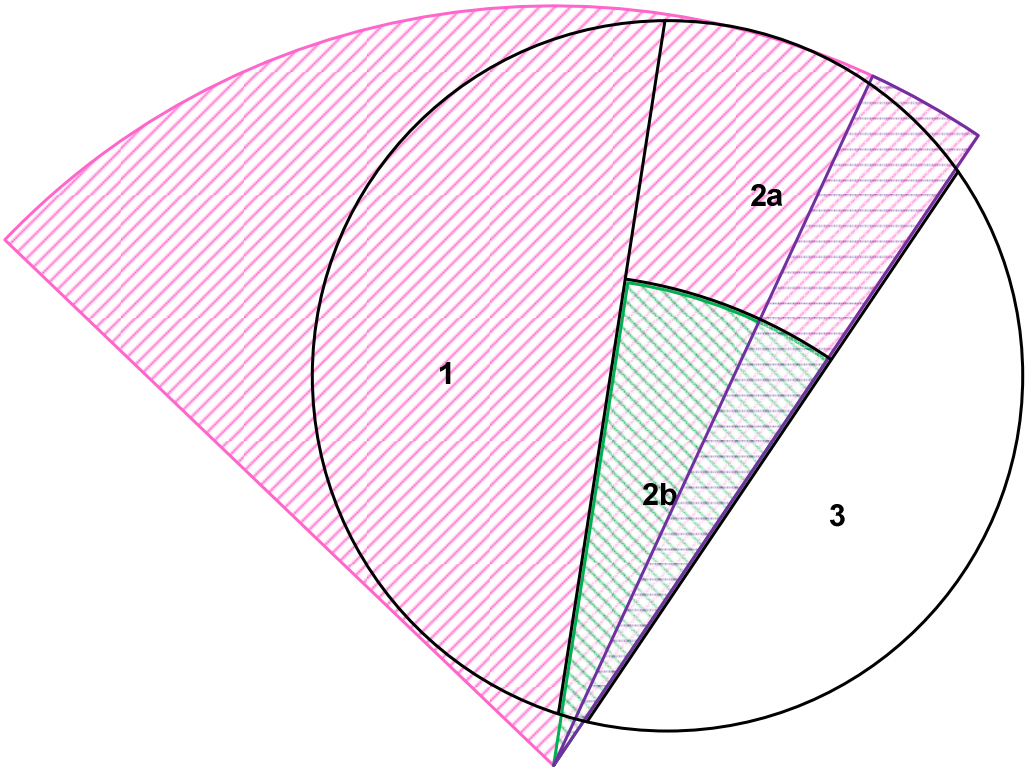


Figure 5-8: The purple sector shows how sensor capability A, in the situation of Figure 5-7, can be used for another operational task (which is not depicted here).

When occurrences of one sensor capability with different sets of parameters in the capability lists are considered as different sensor capabilities, from the previous examples the following can be concluded for sensor capabilities that belong to the same sensor. Suppose that we partition the total region of these sensor capabilities from the capability lists into regions that can be represented with our parameters that, as far as bearing and elevation go, can be covered by different combinations of sensor capabilities. Then for each of these regions, we would want to maximally use each of the (considered) sensor capabilities (that can be used over it) over the region or not at all. When we base the parameter choices for sensor capabilities for alternatives on this, it leaves us with a finite number of alternatives. In fact, we will partition the regions of the sensor capabilities from the capability lists in accordance with the partition made earlier to obtain combinations of sensor capabilities and sets of parameters that can be used as building blocks for alternatives. More precisely, a building block is an object of the form (c, p), where c is a sensor capability and p is a set of parameters. Notice that this partitioning of regions of sensor capabilities may give more than one occurrence of certain combinations of sensor capabilities and parameters. We are only interested in these combinations, though, and not the number of occurrences.

The following procedure computes the building blocks for alternatives for a sensor s as just described:

computeAlternativeBuildingBlocks(s)

1. Let CP = {(c, p) ∈ the union of all capability lists | sensor capability c belongs to sensor s}. (Note, that p is a set of range, bearing and elevation parameters.)
2. Compute region A as the union of the regions represented by the sets of parameters p for which there exists a (c, p) ∈ CP.
3. Partition region A into regions A_1, …, A_n that can be represented by a set of range, bearing and elevation parameters, such that for each A_i there is a different set $CP_i \subseteq CP$ that contains exactly the (c, p) ∈ CP for which the intersection of A_i and the region represented by p is not empty.
4. Let B be an empty set. For each (c, p) ∈ CP, add (c, p_1), …, (c, p_k) to B, where the regions represented by p_1 to p_k make up a partition of the region represented by p, and are each inside region A_i for some i.
5. Return B.

Now, we give an example that illustrates the determination process of building blocks for alternatives. Suppose that for one of the available sensors the regions corresponding with the capabilities given in the capability lists are as depicted in Figure 5-9. Figure 5-10 shows these regions altogether in one picture and also the partitioning of the total region as just described, into three regions. The results of partitioning the regions of Figure 5-9 by these three regions, which can be used as building blocks for our alternatives, are shown in Table 5-5. One of the occurrences of sensor capability B in region 3 can be left out, though, since the part of B1 and B2 in region 3 are exactly the same.

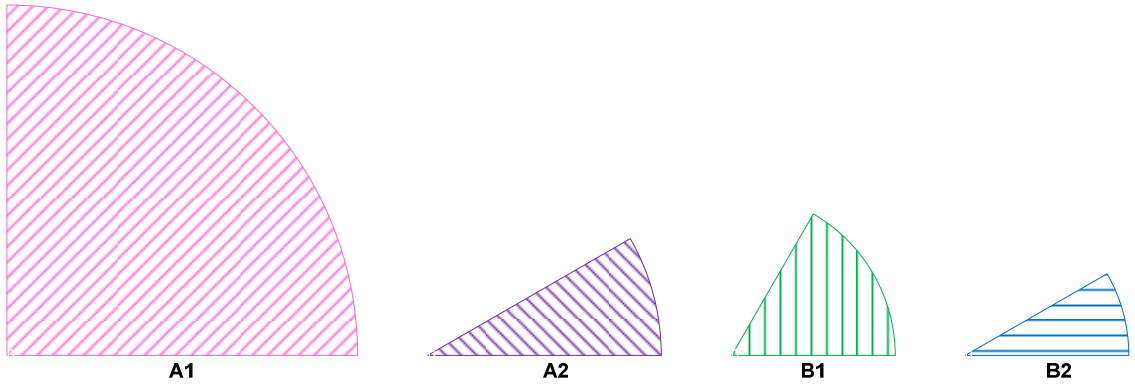


Figure 5-9: Two different ways are shown, in 2D, in which sensor capability A can be used (A1 and A2) and the same for sensor capability B (B1 and B2).

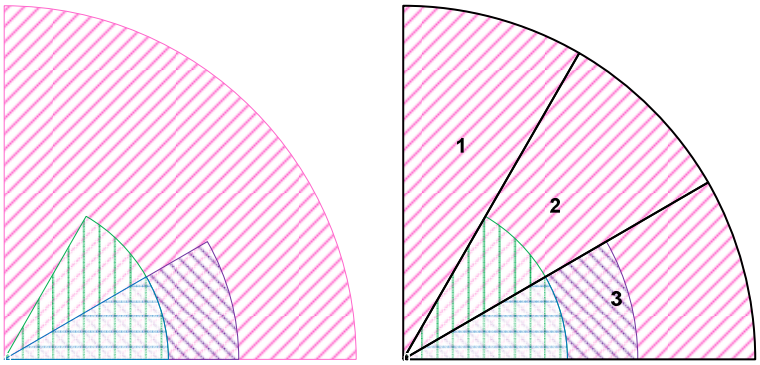
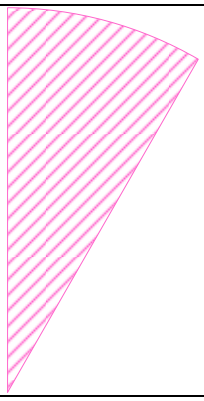
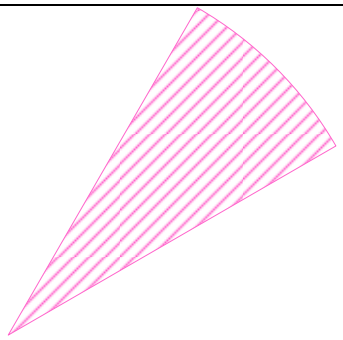
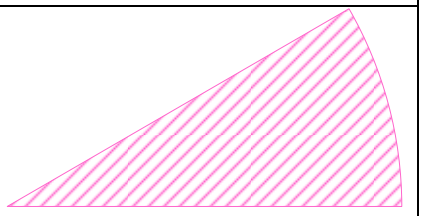
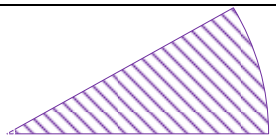
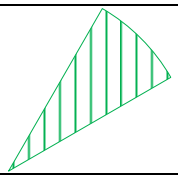
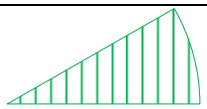
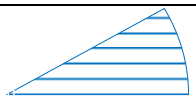


Figure 5-10: On the left, the regions of Figure 5-9 are shown in one picture. On the right, the total region is partitioned by bearing (or elevation) into regions 1 to 3 (outlined in black).

Table 5-5: The results of the regions from Figure 5-9 partitioned by the regions 1 to 3 from Figure 5-10.

	1	2	3
A1			
A2			
B1			
B2			

Note that for the determination of the building blocks as described, only sensor capabilities of a single sensor are considered at a time. In fact, considering sensor capabilities of other sensors when partitioning the region for one sensor can increase the number of alternative building blocks drastically, not to mention the number of possible alternatives. Let us have a look at an example. Suppose that there are two sensors and the partitioning of regions has been done for each of these sensors separately. Figure 5-11 shows two regions corresponding with sensor capabilities of sensor 1 and one region that corresponds with a sensor capability of sensor 2. Note that in the case that the sensor capabilities of both sensors are not allowed to be used over the same region (due to inter-sensor constraints), we may only want to use

sensor capability A over the region on the left of l_a and/or sensor capability B over the region on the left of l_b (see Figure 5-11). This means that when we want to further partition the total region of sensor 1, taking the region of sensor 2 into account, the region of sensor capability B that is completely included in the region of sensor capability A also needs to be considered. Splitting up the regions of sensor capabilities A and B at the corresponding dotted lines gives reason for the other of the two capabilities to be split up along the same line as well (see Figure 5-12), on account of the intra-sensor constraints. When further partitioning the total region of sensor 2 (taking the regions of sensor 1 into account), the region of sensor capability C will, among others, be split up at l_3 from Figure 5-13. This then causes the region of sensor 1 to be split up further, which then again results in the region of sensor 2 having to be further partitioned. It goes on like that for a while, eventually giving a lot of (relatively) rather small regions for building blocks. For the sake of convenience, the sensor capabilities considered for the construction of alternative building blocks for each sensor will be kept to the sensor capabilities of the concerning sensor.

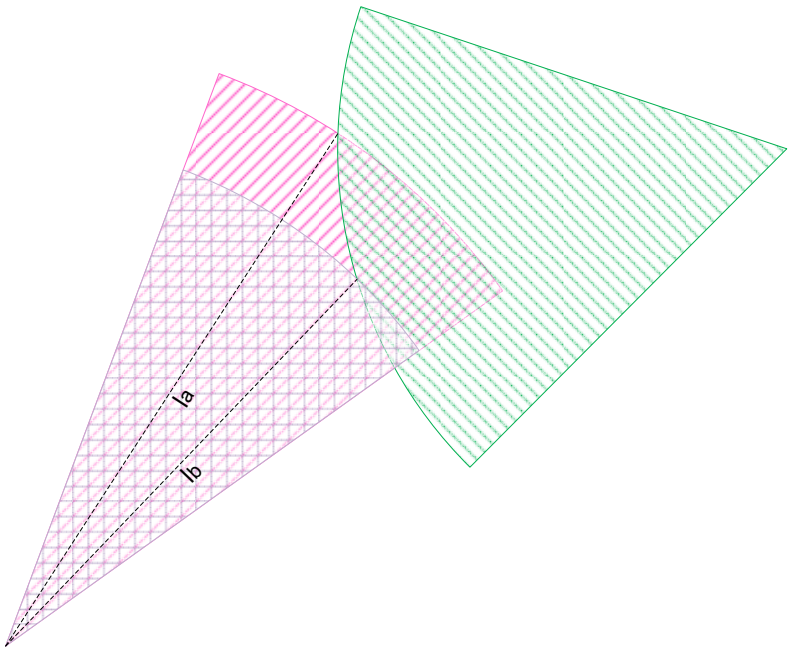


Figure 5-11: The pink and purple regions correspond with sensor capabilities A and B, respectively, of sensor 1 and the green region corresponds with sensor capability C of sensor 2, in 2D. The dotted lines l_a and l_b, starting at sensor 1, split up the regions of sensor capability A and B, respectively, in a part that intersects the region of sensor capability C and a part that does not.

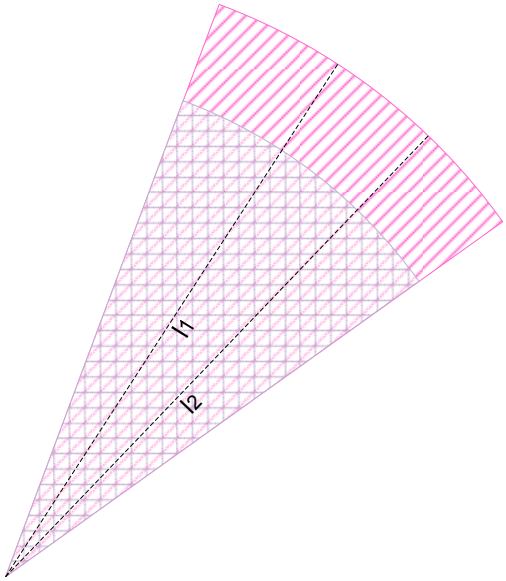


Figure 5-12: The dotted lines show where (both) the regions of sensor capabilities A and B from Figure 5-11 should be split up when sensor capability C, also from Figure 5-11, is considered in the calculation of the alternative building blocks for sensor 1.

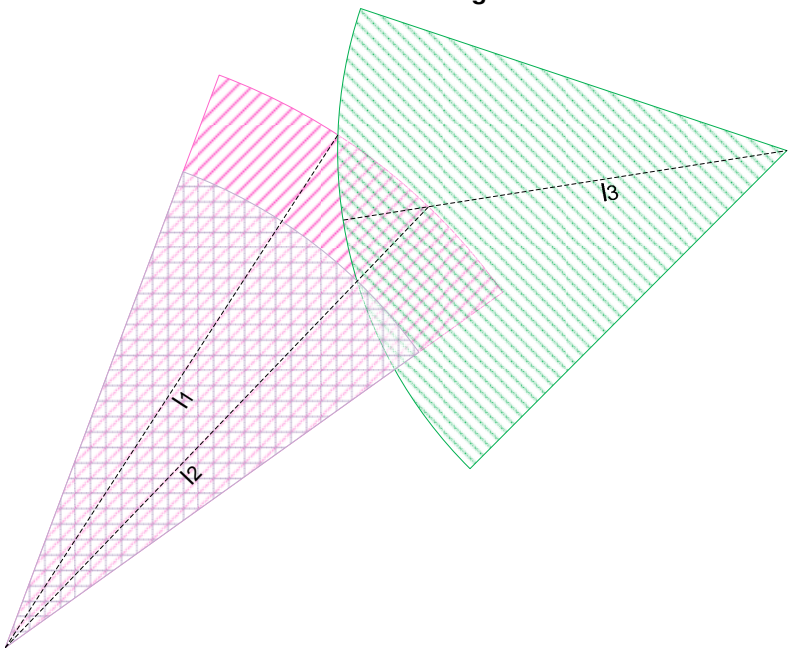


Figure 5-13: The regions from Figure 5-11 are shown together with the dotted lines from Figure 5-12. Dotted line l_3, starting at sensor 2, splits up the region of sensor capability C in a part that intersects the region of sensor capability A on the right of l_2 and a part that does not.

At last, it is time to have a look at the construction of the alternatives to consider. This is simply a matter of determining the building blocks for each of the available sensors and then taking all possible combinations of the set of all building blocks:

computeAlternatives()
1. Let B be the union of the results of computeAlternativeBuildingBlocks(s) for all sensors s.
2. Return the power set of B minus the empty set.

The second step is the point where the intra- and inter-sensor constraints can be taken into account in the future, by only returning the appropriate subsets of B.

Instead of considering the constructed alternatives as such, some alterations can be made. Some of the alternatives determined this way may have building blocks that correspond with regions including or included in other regions corresponding with the same sensor capability. When this occurs, the building blocks corresponding with the regions included within another region can be left out, since they have no influence on the final outcome. In fact, alternatives containing such building blocks can be left out, since for each of these alternatives there exists an alternative without the redundant building block. Furthermore, some of the constructed alternatives may contain building blocks that come from the same sensor capability and that correspond with regions adjacent to each other. In case that the range values corresponding with these building blocks are the same, the building blocks can be combined into one. This could save some computing time later on. Moreover, without unnecessary fragmentation of the regions over which to deploy each of the sensor capabilities, the alternative looks neater and it better portrays real sensor usage.

5.2.2 The Set of Criteria

After having defined the alternatives, it is time to define the decision criteria (see Figure 5-14).

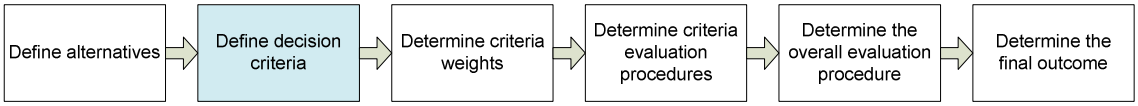


Figure 5-14: Second step of the flow chart of Figure 5-1.

In order to choose suitable criteria for finding the optimal sensor usage, we need to consider the goal to achieve, which is to fulfil the given set of operational tasks as optimally as possible. As follows from Section 3.4, we cannot take costs into account. This leaves just the fulfilment of the set of operational tasks, which depends on the individual operational tasks. Hence, these operational tasks, or their levels of satisfaction, will be considered as our decision criteria.

5.2.3 The Criteria Weights

Now the decision criteria have been defined, we can have a look at a way to weigh them (see Figure 5-15).

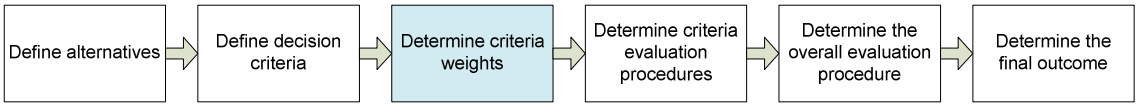


Figure 5-15: Third step of the flow chart of Figure 5-1.

Each of the chosen decision criteria (operational tasks) has a priority value (see Chapter 3), which is very suitable for determining the criteria weight. The possible priority values are given in Table 5-6 (ascending in importance). Now, the question is how to translate task priorities into weight values. Of course, a higher priority should

imply a higher weight. But then various possibilities still exist, from assigning similar weight values to each priority, to giving an operational task with a higher priority such a high weight that the other tasks have only little to no influence at all on the final result. As already mentioned in Subsection 4.2.5, not much, practically nothing actually, is given on how to handle task priorities. Since we do not want to make it unnecessarily complicated, the priority values will simply be assigned the integer weight values from 1 to 6 as in Table 5-6.

Table 5-6: Listing of the possible priority values (ascending in importance) for an operational task and the weights assigned to them.

Priority	Weight
VERY LOW	1
LOW	2
MEDIUM	3
HIGH	4
VERY HIGH	5
TOP	6

5.2.4 The Criteria Evaluations

The next issue (see Figure 5-16) concerns the evaluation of the alternatives on each of the criteria. Since the criteria are all of the same form, namely that of an operational task, only one evaluation procedure is needed here.

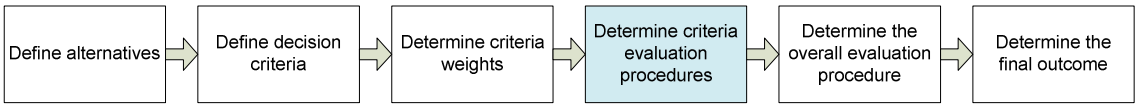


Figure 5-16: Fourth step of the flow chart of Figure 5-1.

To come up with an appropriate procedure to assess an alternative with respect to the satisfaction of an operational task, the factors that are of influence from Section 3.4 need to be considered. As the priority of operational tasks will be used as criteria weights, this leaves the following factors:

- The position of the sensors.
- The region corresponding with the operational tasks.
- The input capability lists.
- The desired performance prescribed for every target type (which defines the desired performance for the operational tasks).
- The quality of service of the sensor capabilities.
- The target type corresponding with the operational tasks.

Starting out with an alternative (set of sensor capabilities and the accompanying parameters) using the sensor positions, the total region coverage of the alternative can be determined. When also considering the region corresponding with the operational task, together with the information from the capability lists on which sensor capabilities can cover this task, the useful region coverage (the coverage of the region of the operational task) can be determined. Furthermore, considering the desired performance prescribed for every target type and the quality of service of the sensor capabilities (of the alternative), the target type that corresponds with the

operational task defines the desired performance and the delivered performance (for the operational task), respectively. In conclusion, to assess an alternative with respect to the satisfaction of an operational task, the degree of both useful region coverage and performance satisfaction can be considered.

Note that, unlike with the desired performance which is the same over the whole region of the operational task (since it depends only on the target type that corresponds with the task and some predefined desired performance), with the delivered performance and with it the performance satisfaction, in general, this is not the case. This is because the delivered performance depends on the used sensor capabilities, which may vary over the region of the operational task. An illustration of this is given by the next example. Suppose that one operational task is given and with the considered alternative two sensor capabilities are used as shown in Figure 5-17. Now, the performance delivered over region 1 depends only on the performance of sensor capability A, whereas the performance delivered over region 2 depends on the performance of both sensor capabilities. As a consequence, performance satisfaction cannot be computed separately for an operational task. The useful region coverage needs to be determined first. Then, only after the region of the operational task has been partitioned into regions that are covered by different combinations of sensor capabilities, can the performance satisfaction be determined (for each of these regions).

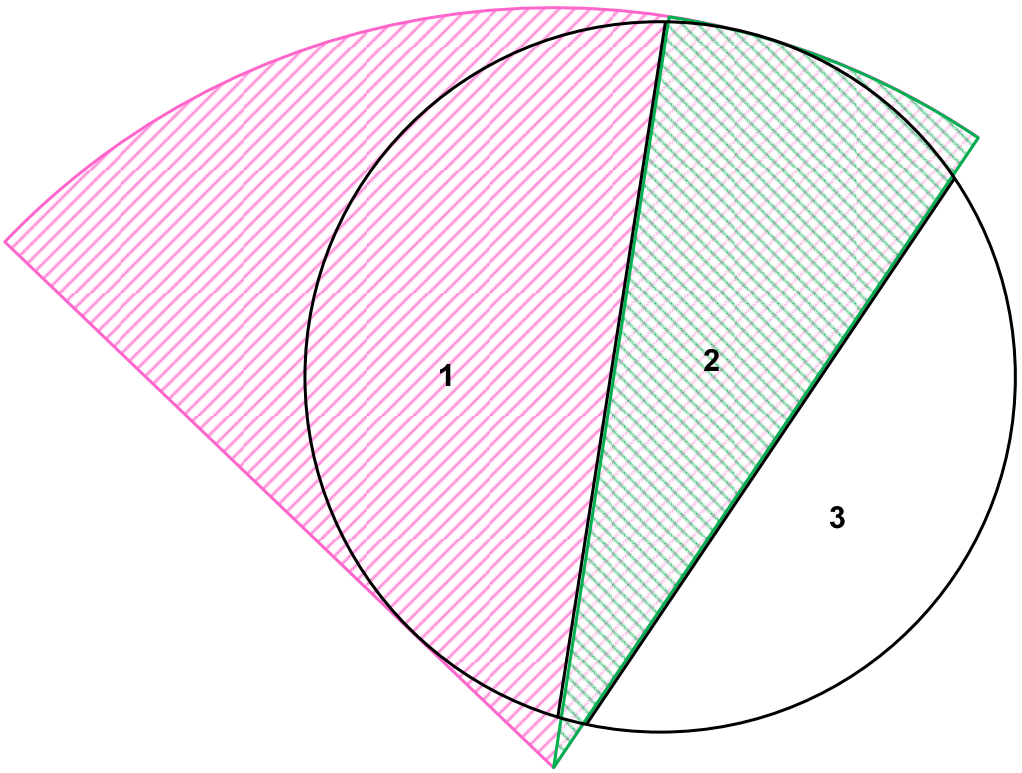


Figure 5-17: The circle represents the region of an operational task, in 2D, and the pink and green sectors show how sensor capabilities A and B, respectively, can be used for the operational task. The region of the operational task is partitioned into three regions, numbered from 1 to 3, that can each be covered by different combinations of sensor capabilities.

Next, the assessment of region coverage and performance satisfaction will be regarded. After this, these will be combined into a procedure that computes the value of an alternative with respect to an operational task.

5.2.4.1 Region coverage

What we are interested in here, is the assessment of the coverage of a part of the region corresponding with an operational task. To compute a score for such a subregion, we could define some real-valued function f(x,y,z) over the total region of the operational task and take the integral of this function over the subregion. Higher function values can be assigned to certain points or regions that are considered more important, such that they will have a higher contribution to the score when included in the covered region. When function f is chosen such that the integral over the whole region is 1, the integral over a region will give an indication of its share in the total region of the operational task (which, of course, is always a number between and including 0 and 1). The simplest case (f(x,y,z) = constant) then comes down to computing the (mathematical) volume that is covered and divide it by the volume of the total region that corresponds with the operational task. Since nothing is specified about a difference in importance of different parts of the region of interest, for the sake of convenience, the simplest function will be used. This means that the following procedure can be used for computing the region score for a subregion A of the region corresponding with operational task OT:

computeRegionScore(A, OT)
1. Compute the volume V1 of region A.
2. Compute the volume V2 of the region corresponding with operational task OT
3. Return V1/V2.

5.2.4.2 Performance satisfaction

Before going into performance satisfaction, let us first have a better look at the desired and delivered performance.

The desired performance for an operational task consists of four values, one for each of the performance factors. The possible values for detection (ascending in quality) are VERY SHORT RANGE, SHORT RANGE, MEDIUM RANGE, LONG RANGE and VERY LONG RANGE. For the factors measurement, resolution and update, the possible values (ascending in quality) are VERY LOW, LOW, MEDIUM, HIGH, VERY HIGH and TOP.

As mentioned earlier, the performance delivered by an alternative may vary over the region of the operational task. For a region that is completely covered by all sensor capabilities that can contribute to the concerning operational task and that are used over the region, there are three possibilities:
1. The region is not covered by any sensor capabilities, which means that no performance is delivered over the region.
2. The region is covered by exactly one sensor capability, which means that the delivered performance over the region is the performance delivered by the concerning sensor capability.

3. The region is covered by more than one sensor capability, which means that the delivered performance over the region depends on the performance delivered by each of the sensor capabilities.

In case 1, of course, the performance satisfaction is 0. In the other two cases, like with the desired performance, the delivered performance consists of four values, one for each of the performance factors. The possible values are also the same as with the desired performance. In case 2 the delivered performance is quite clear, but with case 3, on the other hand, the performances delivered by the used sensor capabilities need to be compared on each of the performance factors. Note that the delivered performance is, of course, defined by the best performance delivered. Hence, the value for each of the performance factors is the highest value offered by the different sensor capabilities (for that factor). Next, a small example in illustration of this. Table 5-7 shows the performances of the two sensor capabilities that cover (part of) the region of an operational task and the eventual delivered performance.

Table 5-7: An example of the performance delivered by two sensor capabilities and the (combined) delivered performance.

	Performance value sensor capability A	**Performance value sensor capability B**	**Delivered performance value**
Detection	SHORT RANGE	MEDIUM RANGE	MEDIUM RANGE
Measurement	VERY HIGH	MEDIUM	VERY HIGH
Resolution	LOW	LOW	LOW
Update	LOW	VERY HIGH	VERY HIGH

Moving on to the performance satisfaction. Since the considered performance consists of four performance factors that can be handled in similar ways, first the assessment of the satisfaction of a performance factor will be considered, after which the assessments of the different factors will be used in a procedure for computing one single performance satisfaction score.

To go from the non-numerical values for a performance factor to a numerical score, we first map combinations of desired and delivered values for the performance factor onto a set of numbers. Suppose the set of possible values for the performance factor (ascending in quality) is $\{v_1, \ldots, v_n\}$. Then the combination of the desired value $v_{desired}$ and delivered value $v_{delivered}$ will be mapped to $x(v_{desired}, v_{delivered}) = \min(delivered/desired, 1)$. When the desired performance (for the concerned performance factor) is satisfied, we have *delivered* >= *desired*, which gives x = 1. When the desired performance is not satisfied, we have *delivered* < *desired* and x < 1 will increase as the delivered performance gets closer to the desired performance. Table 5-8 and Table 5-9 give an overview of the x-values for the different combinations of desired and delivered performance values.

Now, the next step is to map x to a score f(x) for the satisfaction of the performance factor. From our analysis it follows that f(x) should be increasing in x. Furthermore, since partial satisfaction is better than none, f should be non-negative and we can take the score for total satisfaction to be f(1) = 1. The value of the different possible cases of partial satisfaction can be used to determine f. Since, in our case, it is not prescribed how to assess partial satisfaction, we just keep it simple by using f(x) = x.

What is left after computing the scores for each of the performance factors, is to use these scores to determine an overall score for the performance satisfaction. Since it is not given how to handle the different performance factors, we can just take the weighted sum of the computed scores with equal weights ¼. This will give a performance satisfaction score between 0 and 1, where 0 means no satisfaction at all and 1 means total satisfaction.

In short, when a performance p is delivered for operational task OT, the corresponding performance satisfaction score can be computed by the following:

computePerformanceScore(p, OT)
1. If p is ø (no performance delivered), return 0.
2. Determine the desired performance ds for operational task OT.
3. Let s = 0. For each performance factor, find the x-value that corresponds with its value in p and ds in Table 5-8 or Table 5-9, whichever one is appropriate, and add it to s.
4. Return s/4.

Table 5-8: An overview of the different combinations of desired and delivered performance values for performance factor detection, an intermediate numerical value for each combination and the number x it is mapped to.

Detection			
Desired performance value ($v_{desired}$)	**Delivered performance value ($v_{delivered}$)**	**delivered/desired**	**x**
VERY SHORT RANGE (v_1)	VERY SHORT RANGE (v_1)	1	1
	SHORT RANGE (v_2)	2	1
	MEDIUM RANGE (v_3)	3	1
	LONG RANGE (v_4)	4	1
	VERY LONG RANGE (v_5)	5	1
SHORT RANGE (v_2)	VERY SHORT RANGE (v_1)	½	½
	SHORT RANGE (v_2)	1	1
	MEDIUM RANGE (v_3)	1½	1
	LONG RANGE (v_4)	2	1
	VERY LONG RANGE (v_5)	2½	1
MEDIUM RANGE (v_3)	VERY SHORT RANGE (v_1)	1/3	1/3
	SHORT RANGE (v_2)	2/3	2/3
	MEDIUM RANGE (v_3)	1	1
	LONG RANGE (v_4)	1 1/3	1
	VERY LONG RANGE (v_5)	1 2/3	1
LONG RANGE (v_4)	VERY SHORT RANGE (v_1)	¼	¼
	SHORT RANGE (v_2)	½	½
	MEDIUM RANGE (v_3)	¾	¾
	LONG RANGE (v_4)	1	1
	VERY LONG RANGE (v_5)	1¼	1
VERY LONG RANGE (v_5)	VERY SHORT RANGE (v_1)	1/5	1/5
	SHORT RANGE (v_2)	2/5	2/5
	MEDIUM RANGE (v_3)	3/5	3/5
	LONG RANGE (v_4)	4/5	4/5
	VERY LONG RANGE (v_5)	1	1

Table 5-9: An overview of the different combinations of desired and delivered performance values for the performance factors measurement, resolution and update, an intermediate numerical value for each combination and the number x it is mapped to.

Measurement, Resolution en Update			
Desired performance value ($v_{desired}$)	**Delivered performance value ($v_{delivered}$)**	**delivered/desired**	**x**
VERY LOW (v_1)	VERY LOW (v_1)	1	1
	LOW (v_2)	2	1
	MEDIUM (v_3)	3	1
	HIGH (v_4)	4	1
	VERY HIGH (v_5)	5	1
	TOP (v_6)	6	1
LOW (v_2)	VERY LOW (v_1)	½	½
	LOW (v_2)	1	1
	MEDIUM (v_3)	1½	1
	HIGH (v_4)	2	1
	VERY HIGH (v_5)	2½	1
	TOP (v_6)	3	1
MEDIUM (v_3)	VERY LOW (v_1)	1/3	1/3
	LOW (v_2)	2/3	2/3
	MEDIUM (v_3)	1	1
	HIGH (v_4)	1 1/3	1
	VERY HIGH (v_5)	1 2/3	1
	TOP (v_6)	2	1
HIGH (v_4)	VERY LOW (v_1)	¼	¼
	LOW (v_2)	½	½
	MEDIUM (v_3)	¾	¾
	HIGH (v_4)	1	1
	VERY HIGH (v_5)	1¼	1
	TOP (v_6)	1½	1
VERY HIGH (v_5)	VERY LOW (v_1)	1/5	1/5
	LOW (v_2)	2/5	2/5
	MEDIUM (v_3)	3/5	3/5
	HIGH (v_4)	4/5	4/5
	VERY HIGH (v_5)	1	1
	TOP (v_6)	1 1/5	1
TOP (v_6)	VERY LOW (v_1)	1/6	1/6
	LOW (v_2)	1/3	1/3
	MEDIUM (v_3)	½	½
	HIGH (v_4)	2/3	2/3
	VERY HIGH (v_5)	5/6	5/6
	TOP (v_6)	1	1

5.2.4.3 Combining scores

After establishing ways to obtain scores for regions and performance satisfaction, it is time to use these to compute the score for an alternative with respect to the satisfaction of the operational task.

Recall that a performance satisfaction score is always associated with a region that is completely covered by all sensor capabilities that can contribute to the considered task and that are used over it. The contribution of the score associated with such a region to the score for the satisfaction of an operational task, depends on some kind of measure for the region concerned. It can be, for instance, that even with total performance satisfaction the contribution to the final score is quite small, because it only concerns some insignificant region, whereas partial satisfaction could cause a much larger contribution. As a consequence, the contributions to the score for the operational task need to be computed separately for the regions covered by different combinations of sensor capabilities, by combining the performance satisfaction score with a score for the region, after which the score contributions can be summed to obtain the score for the task. For the contribution of a region, we simply multiply the associated scores for performance and region. Note that it does not matter whether not covered regions are included in the sum, since the 0 performance satisfaction will make sure that they do not contribute anything to the score. Furthermore, the region scores for non-overlapping regions that make up the region of the operational task, sum up to 1. In conclusion, it comes down to the score for the operational task being the weighted sum of the performance satisfaction scores, with the region scores as weights. No useful region coverage or performance satisfaction at all gives a score of 0, while total coverage of the region of the operational task with total performance satisfaction gives a score of 1. In between, there are different ways of partial coverage and/or performance satisfaction.

The procedure to compute the score for an alternative *alt*, which is of the form $\{(c_1, p_1), \ldots, (c_n, p_n)\}$, where the c_i are (not necessarily different) sensor capabilities and the p_i are sets of range, bearing and elevation parameters representing the regions over which these sensor capabilities are used, with respect to the satisfaction of the operational task OT looks as follows:

computeOperationalTaskScore(alt, OT)

1. Let alt' be the set of (c, p)-elements $\in$ alt for which there exists some element (c, q) in the capability list of operational task OT.
2. Partition the region of operational task OT into regions $A_1, \ldots, A_m$ such that for each A_i there is a different set C_i that contains exactly the c for which there exists a (c, p) $\in$ alt' for which the intersection of A_i and the region represented by p is not empty.
3. Let $dp_i = \emptyset$, for i = 1, …, m. For each C_i, determine the performance of each of its elements for operational task OT and let the values for each of the performance factors of dp_i be the highest value offered by the sensor capabilities in C_i for that performance factor.
4. Let s = 0. For each region A_i, add the product of results of computeRegionScore(A_i, OT) and computePerformanceScore(dp_i, OT) to s.
5. Return s.

5.2.5 The Overall Evaluation

We have now developed a procedure for computing a score for an alternative with respect to the satisfaction of an operational task. Moreover, an integer weight value from 1 to 6, depending on its priority, has been assigned to each of the operational tasks. The next step (see Figure 5-18) is to use the scores and weights of the separate operational tasks to compute an overall score for the alternative (with respect to our eventual goal, the satisfaction of the given set of operational tasks).

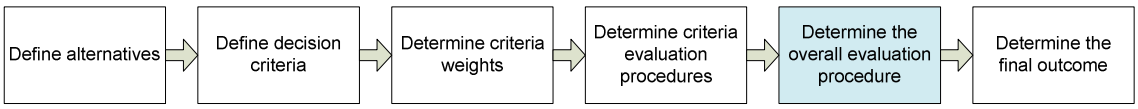


Figure 5-18: Fifth step of the flow chart of Figure 5-1.

For the overall score, the weighted average of the task scores will be taken. The final score will then be between and including 0 and 1, where 0 means no satisfaction at all and 1 means total satisfaction of our goal.

Suppose that operational tasks OT_1, …, OT_n have been given, then the final score for an alternative alt is given by the following:

computeScore(alt)

1. Let w_1, …, w_n be the weights (see Table 5-6) that correspond with operational tasks OT_1, …, OT_n, respectively.
2. Let $s = 0$. For each operational task OT_i, add the product of w_i and computeOperationalTaskScore(alt, OT_i) to s.
3. Return $s/(w_1 + \ldots + w_n)$.

5.2.6 The Final Outcome

Having defined the alternatives and a way to assess each of these alternatives, we can use these to obtain the desired result (see Figure 5-19).

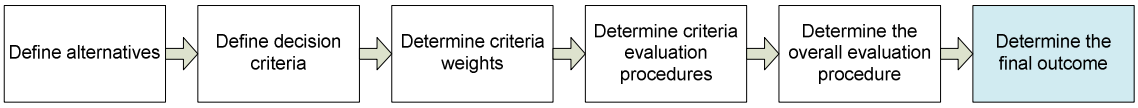


Figure 5-19: Last step of the flow chart of Figure 5-1.

Normally, the last step of the MAUT-procedure is about performing sensitivity analyses and making recommendations. In our case, just one way of sensor deployment, namely the one that scores best on the satisfaction of the given set of operational tasks, needs to be recommended. In other words, from the considered alternatives, we want the one that comes out of the developed evaluation procedure with the highest score. There may be more than one alternative, however, that satisfies this. When this is the case, any alternative associated with the highest score obtained will do. For the sake of convenience, we will take the first alternative that evaluates to this score.

The following procedure performs the whole process of obtaining optimal sensor usage, using the elements described in this chapter:

findOptimalSensorUsage()
1. Let A be the result of computeAlternatives().
2. If A is empty, return ø.
3. Return the first element alt_i of A = {alt_1, …, alt_n}, for which computeScore(alt_i) ≥ computeScore(alt_j) for all j ≠ i.

6 Implementation

In the last two chapters, an approach has been chosen for our problem and, based on that, an algorithm has been worked out for determining the optimal sensor usage within the STM-simulator system. This chapter is concerned with the implementation of this MAUT-based algorithm. Because of the complex nature of our problem (complex structure of the alternatives and decision criteria without some kind of natural scale defined on them), using some existing implementation of the MCDM/MAUT-procedure would require a lot of adjusting or specializing to be done to it. Since the algorithm we developed is actually quite simple, adjusting an existing software implementation like that and making it fit within our system would most probably not be worth the trouble. Therefore, this will not be done.

As mentioned before, the simulator is built in Java, with JESS (Java Expert System Shell) and XML, see also Appendix A. With all relevant elements already being present in the system and the preprocessing done for us, we have nothing to do with JESS or XML anymore. The most important Java classes we are dealing with for the optimization of the sensor usage are grouped in the three packages described below, see also Figure 6-1:

- Demo: contains the class responsible for starting up the application, loading the scenario and creating the graphical user interface (GUI). The Demo class also takes care of all calculations and provides all necessary information to the GUI.
- Beans: contains the classes representing the objects within the scenario.
- SensorTaskingManagement: contains the class for the graphical user interface that allows the user to view and adjust the scenario information, switch between simulation steps and compute the optimal sensor usage.

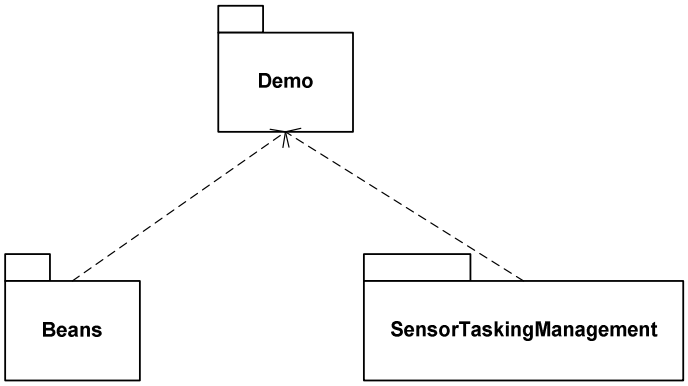


Figure 6-1: A package structure with the most important packages for the optimization of the sensor usage within the STM-simulator.

Now, with a press on a button during a simulation step, the system should compute and show the optimal sensor usage. The most important steps that need to be performed here are the following:

1. Construction of the alternatives to consider.
2. Computing scores for each of the alternatives.
3. Choosing the best alternative.

All this is done by the Demo class, with the help of Beans. This chapter discusses the implementation of these steps.

Instead of implementing the algorithm as developed in the previous chapter, we introduce a few simplifications to make it more manageable, while still maintaining the idea of the algorithm and allowing for our implementation to be more or less easily extensible to match the original algorithm.

- The positions of all sensors of a platform are assumed to be the same as the platform position. This, because the sensor positions are not yet specified in the simulator. Moreover, it significantly reduces the time it takes to compute alternative scores.
- Only two dimensions will be considered in the calculations: altitude and with it elevation will be left out. As a result, we are concerned with 2D-regions, which are easier to work with than regions in 3D. Extending to 3D should be rather simple, though, since elevation can be treated in a similar way as bearing.

6.1 Construction of Alternatives

The construction of the alternatives to consider (see Subsection 5.2.1) basically consists of computing the building blocks and making all possible combinations of these building blocks. Then there is the matter of the extra alterations to the constructed alternatives, described at the end of Subsection 5.2.1. These will be omitted in the implementation for now, since they are not strictly necessary. The most difficult part of the construction of the alternatives, the computation of the building blocks, will be discussed below, followed by some remarks about the forming of the alternatives.

First, let us have a better look at the input we are provided with, which consists of a set of TaskAlternative-objects. Each TaskAlternative corresponds with a combination of a sensor capability and an operational task, see Figure 6-2, and represents a region over which the sensor capability can be used for the operational task. The region is defined by the bearing interval [Bfrom, Bto], elevation interval [Efrom, Eto] and range value Rto.

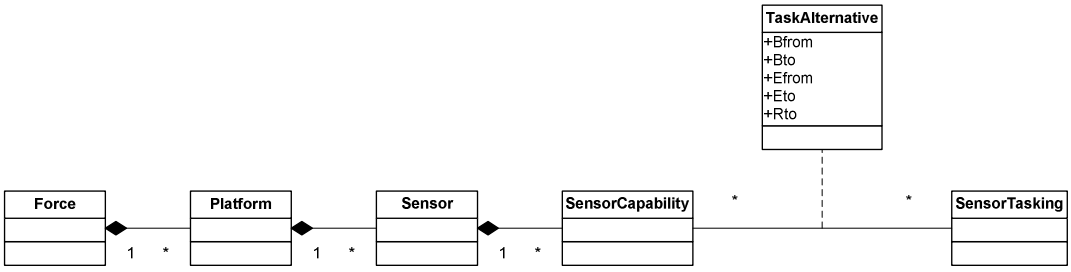


Figure 6-2: Part of the class diagram of the STM-simulator with the classes that are related to the input for the optimization of the sensor usage. (A SensorTasking-object represents an operational task.)

Given that all sensors of a platform are assumed to have the same position, alternative building blocks can be computed at once for all sensor capabilities associated with one platform (instead of one sensor). Furthermore, for the region

over which a sensor capability can be deployed, we now only consider Bfrom, Bto and Rto, which define a sector.

Instead of computing the total region of the sensor capabilities, partitioning it and splitting up the regions associated with the sensor capabilities accordingly, as in computeAlternativeBuildingBlocks() (see Subsection 5.2.1), we use the given TaskAlternative-objects to build a BearingMap2D-structure, from which the alternative building blocks can be extracted in the form of CapabilityAllocation-objects, see Figure 6-3. A BearingMap2D keeps a partition of bearing intervals that can be covered by different combinations of sensor capabilities. With each of these intervals, references to the sensor capabilities that can be used over the interval are kept and with each of these references two lists. First there is a list of range values, over which the sensor capability can be used within the concerning bearing interval. The second list contains references to the operational tasks that the sensor capability contributes to when used over the region defined by the bearing interval and the largest of the range values in the other list, provided that this region intersects the region of the operational task. The reason for keeping the lists of operational tasks here, is that there is no function provided for determining whether a sensor capability can contribute to the execution of an operational task, nor is it clearly specified how to do this. Then for each occurrence of a range value in the BearingMap2D a CapabilityAllocation is created, which contains a reference to a sensor capability, the parameters representing the region (sector) over which the capability can be used and a list of operational tasks which the sensor capability contributes to when used over the region corresponding with the parameters (provided that this region intersects the region of the operational task).

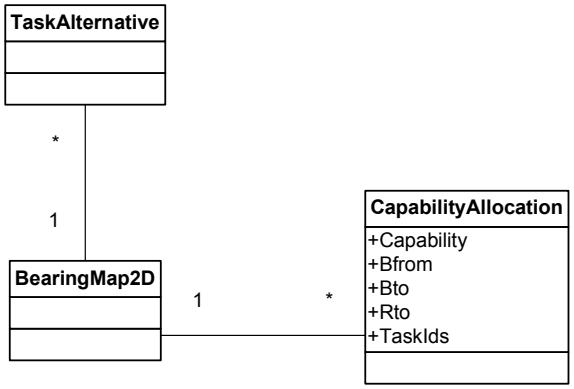


Figure 6-3: Part of the class diagram of the STM-simulator with the classes that are related to the construction of the alternative building blocks.

The created CapabilityAllocation-objects are put into a list, grouped by sensor capability, from which alternatives can be formed. An Alternative just consists of a set of CapabilityAllocation-objects, see Figure 6-4. For now, all combinations of CapabilityAllocation-objects are allowed, but in the future the intra- and inter-sensor constraints and perhaps some other constraints need to be taken into account here. The actual formation of the alternatives is actually not done until they are really needed (when the score needs to be computed for the alternative). The reason for this is that constructing and storing all of the alternatives in advance requires a great deal of memory. Furthermore, this way it saves unnecessary work in case not all

alternatives need to be evaluated (we can stop when an alternative is found that gives a score of 1).

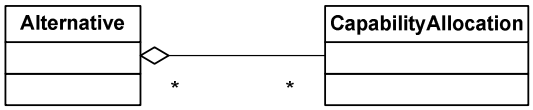


Figure 6-4: Part of the class diagram of the STM-simulator that shows what an Alternative-object consists of.

6.2 Computing Scores

The scores we are interested in are the ones that eventually determine the chosen alternative, which would be the scores for the alternatives with respect to the satisfaction of the given set of operational tasks. Such a score is very easy to compute once the weights of the operational tasks and the scores for the concerning alternative with respect to the individual operational tasks are known (see Subsection 5.2.5). The implementation of the computation of the latter will be discussed below.

As described in Subsubsection 5.2.4.3 the score for an alternative with respect to the satisfaction of an operational task can be computed as a sum of scores associated with the regions of a certain partition of the region of the operational task. To compute these scores, we build a PerformanceMap2D-structure for the combination of the alternative and the operational task, see Figure 6-5, that represents the mentioned partition.

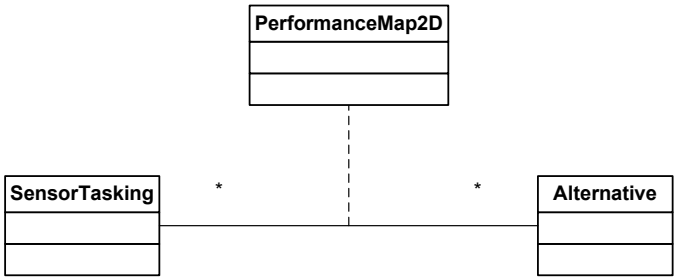


Figure 6-5: Part of the class diagram of the STM-simulator with the classes that are related to the computation of scores for alternatives.

For constructing a PerformanceMap2D, first of all, we represent the 2D-surface that is considered, the surface at a constant distance of the surface or the earth, by a part of a (horizontal) plane. This is done by translating the geographical coordinates of interest to plane coordinates, while placing the first platform (given in the simulation) in the origin. Suppose that the position of this first platform in geographical coordinates is (lat_1, lon_1), in radians, and some other point of interest resides at (lat_2, lon_2). Then the plane coordinates for the second point are computed as follows. First the shortest distance, following the earth's surface, (the great circle distance) between the points (which is always on a plane through the center of the earth) is computed in radians:

$$d = \text{acos}(\sin(lat_1) * \sin(lat_2) + \cos(lat_1) * \cos(lat_2) * \cos(lon_1 - lon_2))$$

The real distance r can be obtained by multiplying d with the radius of the earth. Now, for the position of the second point, we take the point (in the plane) at a distance r from the origin and with an angle from the positive vertical axis that is given by the true course tc from the first to the second point, which is given (in radians) by the following:

```
if (cos(lat1) == 0)
        if (lat1 > 0)
                tc = π
        else
                tc = 2π
else
        if (sin(lon2 - lon1) < 0)
                tc = acos((sin(lat2) – sin(lat1) * cos(d)) / (sin(d) * cos(lat1)))
        else
                tc = 2π – acos((sin(lat2) – sin(lat1) * cos(d)) / (sin(d) * cos(lat1)))
return 2π - tc
```

Classes implementing the Shape interface of the Java AWT library are used to represent the region of the operational task as a circle and the regions associated with CapabilityAllocation-objects as sectors (which may also be circles). The partition of the region of the operational task (that may include partitions of practically any form) is kept as a set of objects from the Area class. For computing this partition, the methods *intersect()* and *subtract()* of class Area are used.

Class Area is not as accurate as we would want, though. In the case that we try to subtract a sector from an overlapping sector with the same circle center and radius, part of the arc of the first sector seems to remain. Moreover, points in the intersection of the two sectors are tested to be inside the resulting Area. See Figure 6-6. This may cause critical errors in subsequent calculations. To make sure we get the desired result in the end, the sector to be subtracted is replaced by a sector with a significant larger radius when the mentioned case occurs.

Besides keeping a partition of the region of the operational task, a PerformanceMap2D keeps track of the delivered performance over each of the partitions. With this, a PerformanceMap2D contains all information needed for computing the desired performance and region scores (for computing the total score for the operational task).

The computation of the performance score is quite straightforward. We could say the same about the region score if only there was a built-in method for computing the area of an Area-object. Unfortunately, this is not the case. On the other hand, it is possible to obtain a bounding rectangle that completely encloses the represented region or an (estimate of the) outline of this region. Moreover, a point or rectangular region can be tested for being (completely) inside the represented region. This allows for different ways to get an estimate of the area. With (an estimate of) the outline being available, Green's theorem can be used for the purpose. A less sophisticated approach is to put the Area on a grid and add up the area of the cells that are completely inside the Area. The method chosen to be used is the Monte Carlo method (used with 10000 points here) that is much easier to implement than Green's

theorem and can give a reasonable estimation, which should suffice for now (since it was not prescribed how to assess the given operational tasks, let alone part of the region of an operational task), within a significantly smaller amount of time than when a grid is used.

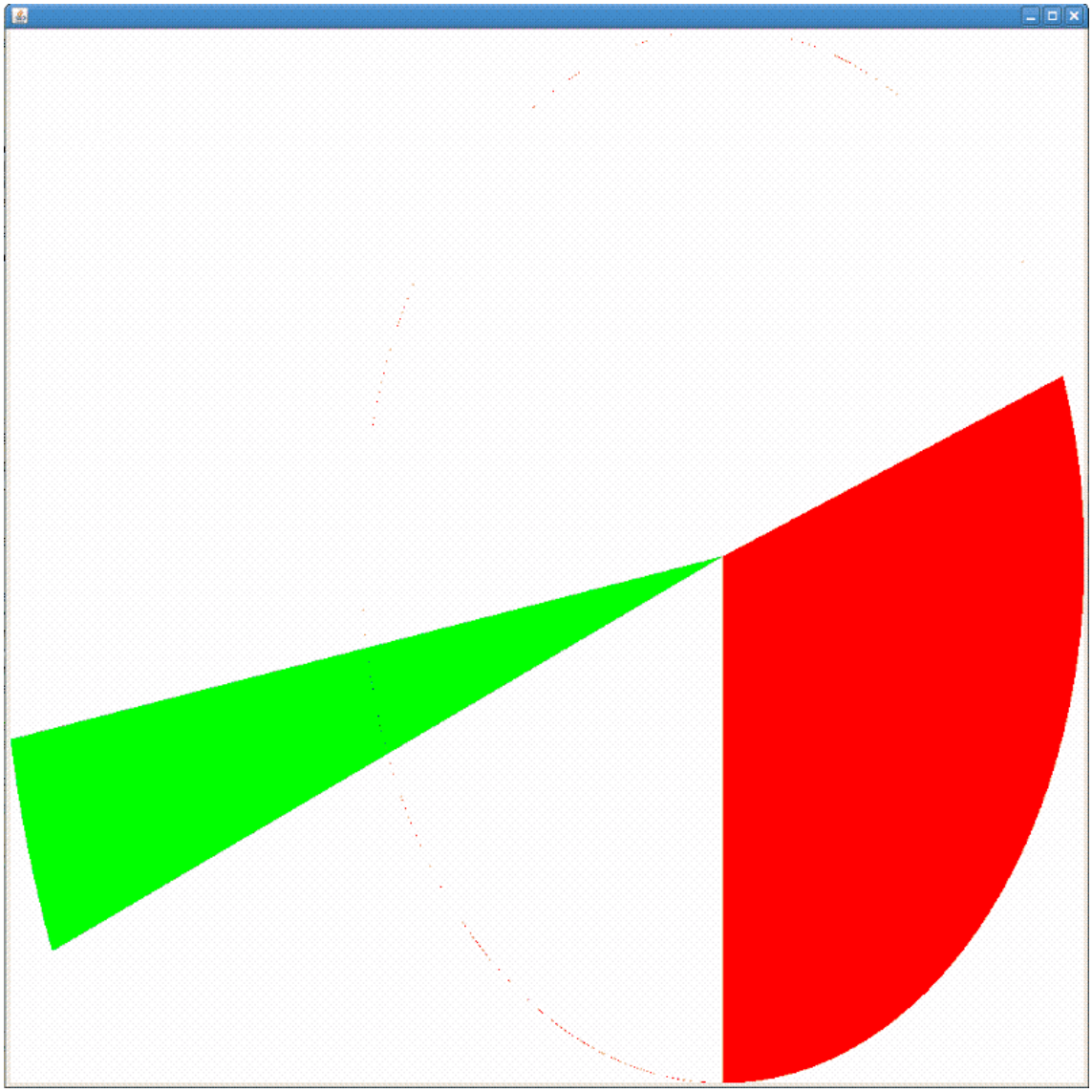


Figure 6-6: In red an Area-object is shown that represents a circle from which different sectors with the same circle center, some with the same radius as the circle and some with a larger radius, have been subtracted. Note that the area includes points from the circle arc. When the region of the intersection of the green sector and the red Area is computed, this gives quite a large number as result.

6.3 Choosing a Solution

As mentioned in Subsection 5.2.6, we want an alternative that gives the highest score. As long as all combinations of building blocks are allowed, it is a sure thing that at least the alternative with all building blocks will give the highest score. However, as followed from the discussion at the beginning of Subsection 5.2.1, we

should keep the intra- and inter-sensor constraints in mind, even though they are not implemented. In other words, we want the simulator to find the alternative with the highest score, presuming that, in general, not all combinations of building blocks are allowed. Still, it is no great feat finding an alternative as such. The number of alternatives to consider can be quite large, though, even in relatively simple simulations. As a result, it takes an unreasonable amount of time to compute scores for all alternatives in most cases.

Different measures can be taken to shorten the time it takes to choose an alternative. A way that is guaranteed to deliver an alternative with the highest score, is distributing the evaluation of alternatives over a number of computers and, in the end, taking the best alternative over all computers. Much less time consuming is the use of satisficing (choose the first alternative with a score that meets or exceeds some chosen value). Even when presuming that, in general, not all combinations of building blocks are allowed, we could say that alternatives with a high number of CapabilityAllocation-objects are much more likely to have (relatively) high scores, whereas alternatives with little CapabilityAllocation-objects are not very likely to have high scores. Taking this into account, we could, for instance, base the choice for a minimum value for satisficing on the score of some alternative with a high (enough) number of CapabilityAllocation-objects. Besides being more likely to have high scores than alternatives with little CapabilityAllocation-objects, alternatives with a high number of CapabilityAllocation-objects are also more likely to have redundant CapabilityAllocation-elements. Bearing this in mind, perhaps some possibly ordered set of alternatives, with only alternatives with a number of CapabilityAllocation-objects between some chosen numbers, could be chosen to be considered. Then, there is always the possibility just to consider some number of alternatives chosen randomly from the available alternatives. For the rest, several of the options just mentioned could be combined.

Recall that the created CapabilityAllocation-objects are put into a list, grouped by sensor capability. Notice that each combination of CapabilityAllocation-elements, or Alternative, can be represented by a binary code such that exactly the elements in the list at the positions with a ‘1’ are considered. We define a numbering on the set of alternatives by associating alternatives with binary codes as such. With this, not only can alternatives with little and/or those with a high number of CapabilityAllocation-objects be left out of consideration quite easily, but the set of alternatives to consider can also be partitioned very easily for parallel processing. For now, two options are implemented in the system. First, there is the option to find the first alternative with the highest score out of all alternatives between two specified numbers. The other option gives the first alternative with the highest score out of some specified number of random alternatives. Here an object from the Java class Random is used, with seed 10.

7 Results

Now that the implementation has been discussed, it is time to have a look at some results. For this, first the scenario in the simulator will be introduced, after which we will have a look at the determination of the sensor usage in two different situations.

Figure 7-1 gives an overview of the platforms in the simulator. The platforms either belong to the own force or the enemy force. The platforms of the enemy force are listed in Figure 7-2. Ofcourse we have more information on the own force, which consists of the frigates HNLMS VAN BUUREN and HNLMS Ho Ho. The sensors of the HNLMS VAN BUUREN are listed in Figure 7-3 and their sensor capabilities are given in Figure 7-4 to Figure 7-6. The HNLMS Ho Ho has only one sensor, with one sensor capability, see Figure 7-7 and Figure 7-8.

Figure 7-1: A view in Google Earth of the platforms in the STM-simulator at their start positions. The white path starting at the HNLMS VAN BUUREN shows its trajectory throughout the simulation.

Name	Type
COBRA	Missile Site
PUMBA	ASCM
Moby Dick	Carrier

Figure 7-2: A listing of the platforms of the enemy force in the STM-simulator.

Name	Type	Class
APAR	MFR	SC IV
SMART-L	SURV Radar	SC III
SIRIUS	*NONE*	SC III

Figure 7-3: A listing of the sensors of the HNLMS VAN BUUREN.

Id	Name	Oper. task type
2	LIMITED VOLUME SEARCH	SurveillanceTask
3	HORIZON SEARCH	SurveillanceTask
4	TARGET TRACK	TrackingTask

Figure 7-4: A listing of the sensor capabilities of the Apar.

Id	Name	Oper. task type
11	AIR SURVEILLANCE	SurveillanceTask
12	AIR ALERT	SurveillanceTask

Figure 7-5: A listing of the sensor capabilities of the Sirius.

Id	Name	Oper. task type
6	LONG RANGE SURVEILLANCE	SurveillanceTask
7	LOCAL AREA DEFENCE	SurveillanceTask
8	SURFACE SURVEILLANCE	SurveillanceTask
9	SURFACE FIRE CONTROL	GunFireSupportTask

Figure 7-6: A listing of the sensor capabilities of the Smart-L.

Name	Type	Class
S1850mod	SURV Radar	SC III

Figure 7-7: A listing of the sensors of the HNLMS Ho Ho.

Id	Name	Oper. task type
42	LONGISH RANGE SURVEILLANCE	SurveillanceTask

Figure 7-8: A listing of the sensor capabilities of the S1850mod.

In the simulator, six different steps can be simulated, in which the HNLMS VAN BUUREN traverses along the path given in Figure 7-1. Operational tasks, platform positions and the availability of the platforms, sensors and sensor capabilities of the own force are specified (and can be changed, see Appendix A) separately for each simulation step. By default all sensor capabilities in the system are available throughout the entire simulation. Table 7-1 gives an overview of some figures that give an indication of the size of the problem within the considered scenario. The number of alternatives during each step is determined by the number of available building blocks (not shown here) for the construction of these alternatives. Note that even with this (relatively) simple scenario, we come across very large numbers of alternatives.

Table 7-1: An overview of the number of operational tasks, TaskAlternative-objects and alternatives during each of the steps of the STM-simulator with the default settings.

step	Number of operational tasks	Number of TaskAlternative-objects	Number of alternatives
0	1	6	2^{12}
1	2	10	2^{26}
2	3	13	2^{39}
3	5	20	2^{68}
4	5	19	2^{73}
5	5	19	2^{77}

Next, two small examples will be handled. First the different steps of the process of determining the sensor usage will be demonstrated in/with a small example. Simulation step 0 will be used for this. Furthermore, we will examine scores for different alternatives and the time it takes to compute these scores. Then we will have a look at a somewhat larger example, simulation step 1, with more than one operational task. We will show how an alternative contributes to each of the operational tasks, after which scores and computing time will be examined again.

7.1 Simulation step 0

At step 0 the platforms are in the positions as shown in Figure 7-1. The positions of the platforms of the own force are given in Table 7-2 and the set of operational tasks to be fulfilled is given in Figure 7-9. The "Area"-field represents the region to be covered, which is actually the circle with a radius of 50000 m around the HNLMS VAN BUUREN. Furthermore, the given TaskAlternative-objects are listed in Figure 7-10 and Figure 7-11 shows all the elements just mentioned together in one picture.

Table 7-2: The geographical positions of the platforms of the own force during step 0 of the STM-simulator.

	HNLMS VAN BUUREN	HNLMS Ho Ho
Lattitude	49.7°	50.2°
Longitude	-2.9°	-0.5°

Id	Task Name	Task Type	Expected Target Type	Priority	Area
11	Missiles 360	SurveillanceTask	SeaSkimmer	high	HNLMS VAN BUUREN

Figure 7-9: A listing of the operational tasks to be fulfilled during step 0 of the STM-simulator.

```
Capability: 2(HNLMS VAN BUUREN)TaakId's: [11]Bfrom: 0.0Bto: 360.0Rto: 50000.0
Capability: 3(HNLMS VAN BUUREN)TaakId's: [11]Bfrom: 0.0Bto: 360.0Rto: 50000.0
Capability: 6(HNLMS VAN BUUREN)TaakId's: [11]Bfrom: 83.02737426757812Bto: 360.0Rto: 50000.0
Capability: 6(HNLMS VAN BUUREN)TaakId's: [11]Bfrom: 0.0Bto: 43.02737808227539Rto: 50000.0
Capability: 12(HNLMS VAN BUUREN)TaakId's: [11]Bfrom: 0.0Bto: 360.0Rto: 50000.0
Capability: 42(HNLMS Ho Ho)TaakId's: [11]Bfrom: 236.88629913330078Bto: 269.07494354248047Rto: 230362.53125
```

Figure 7-10: A listing of the TaskAlternative-objects given at step 0 of the STM-simulator. Besides the information in a TaskAlternative, for each TaskAlternative also the platform is given from which the associated sensor capability originates.

Since only one TaskAlternative is associated with the HNLMS Ho Ho, we can use this directly as an alternative building block. For the rest of the building blocks, first we need to partition the region covered by the TaskAlternative-objects associated with the HNLMS VAN BUUREN. From Figure 7-10 and Figure 7-11 it is clear that the total region that can be covered is the circle with a radius of 50000 m that is also the region of the operational task to be executed. Furthermore, we can easily see that the region should be partitioned in three sectors with roughly the following bearing intervals: [0°, 43°], [43°, 83°] and [83°, 360°]. This means that all TaskAlternative-objects associated with the HNLMS VAN BUUREN should be partitioned at the bearing-values 43° and 83° (it is not necessary to partition at 0° or 360°, since those are by definition never in the interior of an interval) as far as possible. The result is shown in Figure 7-12.

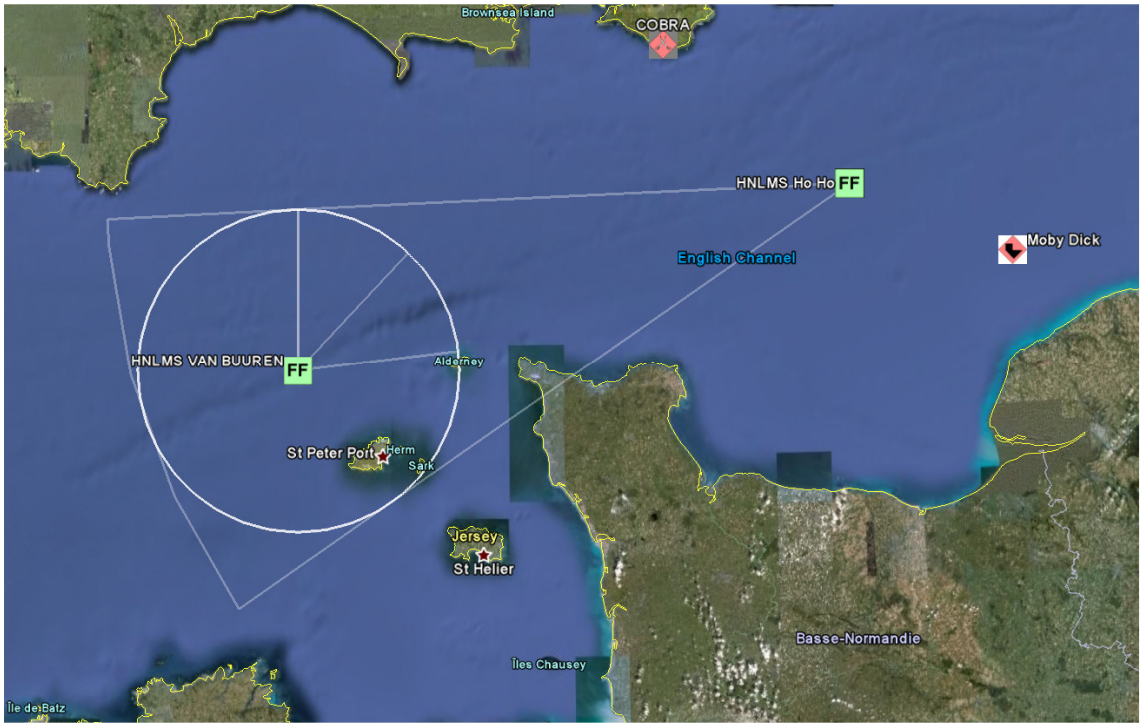


Figure 7-11: A view in Google Earth of the participating platforms of the own force, the operational tasks and TaskAlternative-objects at step 0 of the STM-simulator.

```
Capability: 6(HNLMS VAN BUUREN)TaakId's: [11]Bfrom: 0.0Bto: 43.02737808227539Rto: 50000.0
Capability: 6(HNLMS VAN BUUREN)TaakId's: [11]Bfrom: 83.02737426757812Bto: 360.0Rto: 50000.0
Capability: 3(HNLMS VAN BUUREN)TaakId's: [11]Bfrom: 0.0Bto: 43.02737808227539Rto: 50000.0
Capability: 3(HNLMS VAN BUUREN)TaakId's: [11]Bfrom: 43.02737808227539Bto: 83.02737426757812Rto: 50000.0
Capability: 3(HNLMS VAN BUUREN)TaakId's: [11]Bfrom: 83.02737426757812Bto: 360.0Rto: 50000.0
Capability: 2(HNLMS VAN BUUREN)TaakId's: [11]Bfrom: 0.0Bto: 43.02737808227539Rto: 50000.0
Capability: 2(HNLMS VAN BUUREN)TaakId's: [11]Bfrom: 43.02737808227539Bto: 83.02737426757812Rto: 50000.0
Capability: 2(HNLMS VAN BUUREN)TaakId's: [11]Bfrom: 83.02737426757812Bto: 360.0Rto: 50000.0
Capability: 12(HNLMS VAN BUUREN)TaakId's: [11]Bfrom: 0.0Bto: 43.02737808227539Rto: 50000.0
Capability: 12(HNLMS VAN BUUREN)TaakId's: [11]Bfrom: 43.02737808227539Bto: 83.02737426757812Rto: 50000.0
Capability: 12(HNLMS VAN BUUREN)TaakId's: [11]Bfrom: 83.02737426757812Bto: 360.0Rto: 50000.0
Capability: 42(HNLMS Ho Ho)TaakId's: [11]Bfrom: 236.88629913330078Bto: 269.07494354248047Rto: 230362.53125
```

Figure 7-12: A listing of the alternative building blocks constructed at step 0 of the STM-simulator. Besides the information in a CapabilityAllocation, for each CapabilityAllocation also the platform is given from which the associated sensor capability originates.

Now, let us have a look at an alternative that is formed from the building blocks just created, see Figure 7-13. When deploying the sensors according to this alternative, it actually looks very much like Figure 7-11. When we want to compute the score for the alternative, however, everything needs to be mapped on a flat surface, which gives the situation in Figure 7-14. The error that is caused here is clearly visible, the sector originating from the HNLMS Ho Ho does not enclose the entire region of the operational task as in Figure 7-11. Note that the alternative needs to be evaluated for only one operational task. Figure 7-15 shows a partition of the region of the operational task such that the partitions are exactly the regions that are covered by different combinations of CapabilityAllocation-objects of the alternative (that can contribute to the operational task). Moreover, for each of the partitions an estimate of its area, the performance score and its contribution to the total score for the concerning alternative are given. Obviously, the first subregion is not covered by any sensor capability, hence the performance score of 0. The computation of the

performance score for the second subregion is not that complicated either, since it only covered by one CapabilityAllocation. The third and fourth subregions are each covered by more than one CapabilityAllocation-objects, though. Figure 7-16 shows the intermediate results in the computation of the performance score for the third subregion. First the performances of the concerning CapabilityAllocation-objects for the operational task are considered for determining the delivered performance, after which the score for each of the performance factors is determined from the delivered and desired performance. For completion, the total score for the alternative is given in Figure 7-17.

Capability-inzet: 9Capability: 6(HNLMS VAN BUUREN)TaakId's: [11]Bfrom: 0.0Bto: 43.02737808227539Rto: 50000.0
Capability-inzet: 13Capability: 12(HNLMS VAN BUUREN)TaakId's: [11]Bfrom: 43.02737808227539Bto: 83.02737426757812Rto: 50000.0
Capability-inzet: 18Capability: 42(HNLMS Ho Ho)TaakId's: [11]Bfrom: 236.886299133300078Bto: 269.07494354248047Rto: 230362.53125

Figure 7-13: An alternative constructed from the building blocks in Figure 7-12.

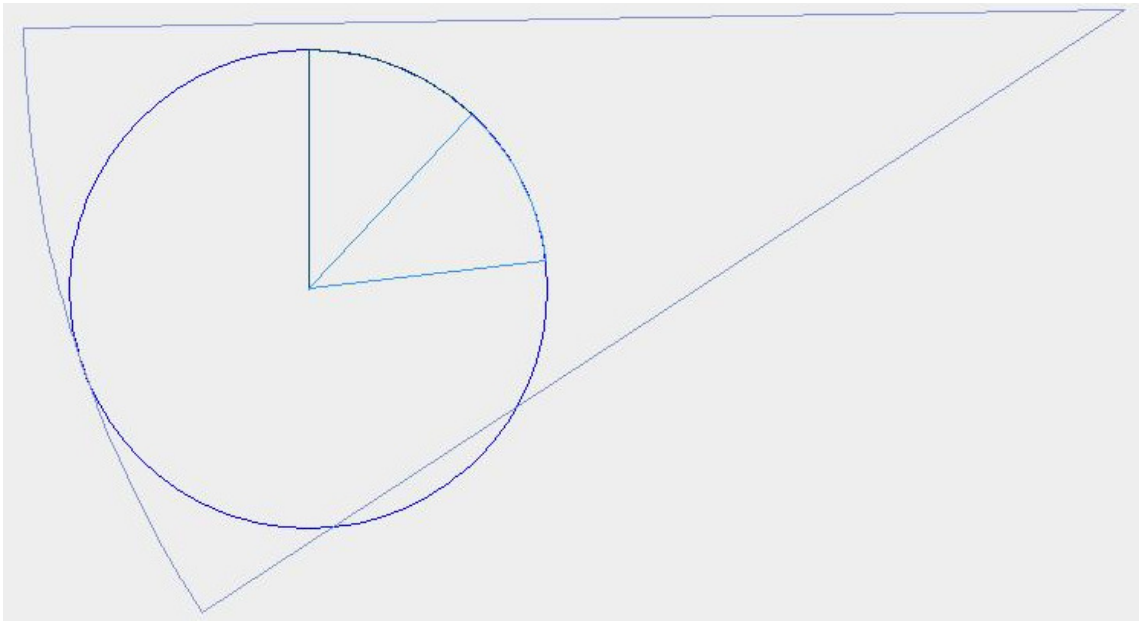


Figure 7-14: The alternative of Figure 7-13 and the operational task at step 0 of the STM-simulator mapped on a flat surface.

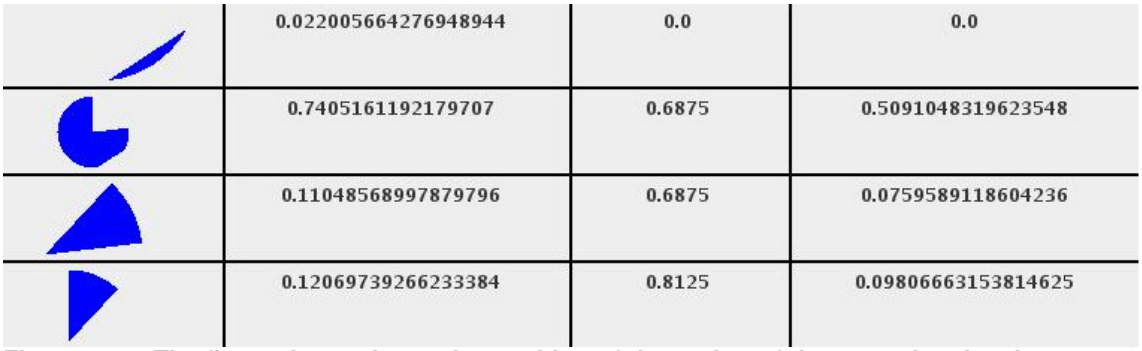
	0.022005664276948944	0.0	0.0
	0.7405161192179707	0.6875	0.5091048319623548
	0.11048568997879796	0.6875	0.0759589118604236
	0.12069739266233384	0.8125	0.09806663153814625

Figure 7-15: The first column shows the partition of the region of the operational task at step 0 of the STM-simulator based on Figure 7-14. The rest of the columns contain for each of the partitions, from left to right, an estimate of its area, the performance score and its contribution to the total score for the concerning alternative.

Performance factor	Capability-inzet 13	Capability-inzet 18	Geboden	Gevraagd	Score	Weight
Detection	VERY SHORT RANGE	MEDIUM RANGE	MEDIUM RANGE	LONG RANGE	0.75	0.25
Measurement	VERY LOW	MEDIUM	MEDIUM	HIGH	0.75	0.25
Resolution	VERY LOW	LOW	LOW	HIGH	0.5	0.25
Update	VERY LOW	MEDIUM	MEDIUM	HIGH	0.75	0.25

Figure 7-16: The delivered performance for the third partition of Figure 7-15 ("Geboden") is determined from the performances of the CapabilityAllocation-objects that cover it (Capability-inzet 13 and 18) for the operational task. The desired performance for the operational task is given under "Gevraagd". Furthermore, the score for each of the performance factors, determined from the delivered and desired performance, is given and its weight.

Task ID	Score	Weight
11	0.6831303753609248	0.6666666666666666

Totale score: 0.6831303753609248

Figure 7-17: The scores of the alternative given in Figure 7-13 for each of the operational tasks in Figure 7-9 together with the weights of these tasks and the total score for the alternative.

A score of 0.68 does not seem very high considering that we would preferably obtain a score of 1. On the other hand, we cannot expect that all operational tasks can be perfectly fulfilled with the available resources in every situation. Furthermore, the quality of service used for each of the sensor capabilities and the desired performances that are used at the moment are not genuine values and might be chosen very poorly. The scores for some other alternatives are given in Table 7-3. The alternative with all building blocks, that should give the highest score, gives a score of 0.9317569255897913. Notice that higher scores are obtained by some other alternatives. This is caused by the approximations of the areas not being accurate enough. The difference in the scores is only 0.3%, though. We can see that an alternative with a score of 0.93 can already be found before the 1000th alternative. It is even possible to find such a high score by evaluating only 100 or 10 alternatives. Many different alternatives may have quite high scores. In fact, the scores for the last n alternatives in Table 7-3 are obtained by different alternatives. Moreover, different seeds give rise to a score of 0.93 within 10 random alternatives.

Table 7-3: The highest scores for different sets of alternatives at step 0 of the STM-simulator.

n	Highest score for the first n alternatives	Highest score for the last n alternatives	Highest score for n random alternatives
10	0.6742031590469111	0.9346025102392764	0.9346025107920549
100	0.8489237304566698	0.9346025102392764	0.9346025107920549
1000	0.9346025107920549	0.9346025102392764	0.9346025107920549

Evaluating all (4095) alternatives to find the first alternative with the highest score takes over 13 minutes (counting in all operations after the computation of the alternative building blocks). Table 7-4 shows the time it takes to find the scores given in Table 7-3.

Table 7-4: The time it takes to find the scores given in Table 7-3.

n	Time to compute the highest score for the first n alternatives	Time to compute the highest score for the last n alternatives	Time to compute the highest score for n random alternatives
10	1 s	3 s	2 s
100	13 s	28 s	20 s
1000	171 s	228 s	203 s

7.2 Simulation step 1

At simulation step 1 the HNLMS VAN BUUREN has moved a bit along the path shown in Figure 7-1. The new platform positions are given in Table 7-5 and the corresponding operational tasks in Figure 7-18. The first of the two tasks seems to have stayed the same as in step 0, but since the position of the HNLMS VAN BUUREN has changed, the region to be covered has shifted along with it. The region to be covered for the second task is the circle with a radius of 1200 m around the COBRA. The given TaskAlternative-objects are listed in Figure 7-19 and Figure 7-20 gives a graphical view of the situation just described.

Table 7-5: The geographical positions of the platforms of the own force during step 1 of the STM-simulator.

	HNLMS VAN BUUREN	HNLMS Ho Ho
Lattitude	50.1°	50.2°
Longitude	-1.6°	-0.5°

Id	Task Name	Task Type	Expected Target Type	Priority	Area
11	Missiles 360	SurveillanceTask	SeaSkimmer	high	HNLMS VAN BUUREN
12	COBRA	SurveillanceTask	SSM	top	COBRA

Figure 7-18: listing of the operational tasks to be fulfilled during step 1 of the STM-simulator.

```
Capability: 2(HNLMS VAN BUUREN)TaakId's: [11]Bfrom: 0.0Bto: 360.0Rto: 50000.0
Capability: 2(HNLMS VAN BUUREN)TaakId's: [12]Bfrom: 17.909788250923157Bto: 20.47358500957489Rto: 54839.72265625
Capability: 3(HNLMS VAN BUUREN)TaakId's: [11]Bfrom: 0.0Bto: 360.0Rto: 50000.0
Capability: 3(HNLMS VAN BUUREN)TaakId's: [12]Bfrom: 17.909788250923157Bto: 20.47358500957489Rto: 54839.72265625
Capability: 6(HNLMS VAN BUUREN)TaakId's: [11]Bfrom: 83.02737426757812Bto: 360.0Rto: 50000.0
Capability: 6(HNLMS VAN BUUREN)TaakId's: [11]Bfrom: 0.0Bto: 43.02737808227539Rto: 50000.0
Capability: 6(HNLMS VAN BUUREN)TaakId's: [12]Bfrom: 17.909788250923157Bto: 20.47358500957489Rto: 54839.72265625
Capability: 12(HNLMS VAN BUUREN)TaakId's: [11]Bfrom: 0.0Bto: 360.0Rto: 50000.0
Capability: 42(HNLMS Ho Ho)TaakId's: [11]Bfrom: 223.85340118408203Bto: 307.49657440185547Rto: 124983.5078125
Capability: 42(HNLMS Ho Ho)TaakId's: [12]Bfrom: 307.4638552069664Bto: 309.37336403131485Rto: 73216.546875
```

Figure 7-19: A listing of the TaskAlternative-objects given at step 1 of the STM-simulator. Besides the information in a TaskAlternative, for each TaskAlternative also the platform is given from which the associated sensor capability originates.

An alternative to be considered at this simulation step is given in Figure 7-21 and Figure 7-22. As opposed to simulation step 0 with only one operational task, here the total score for the alternative, see Figure 7-23, actually needs to be computed from the scores for the operational tasks. With this alternative each CapabilityAllocation-object contributes to the execution of exactly one operational task, but in general the number of operational tasks a CapabilityAllocation-objects can contribute to is not limited to one. In fact, the alternative given in Figure 7-24 and Figure 7-25 has a CapabilityAllocation-object that contributes to both operational tasks.

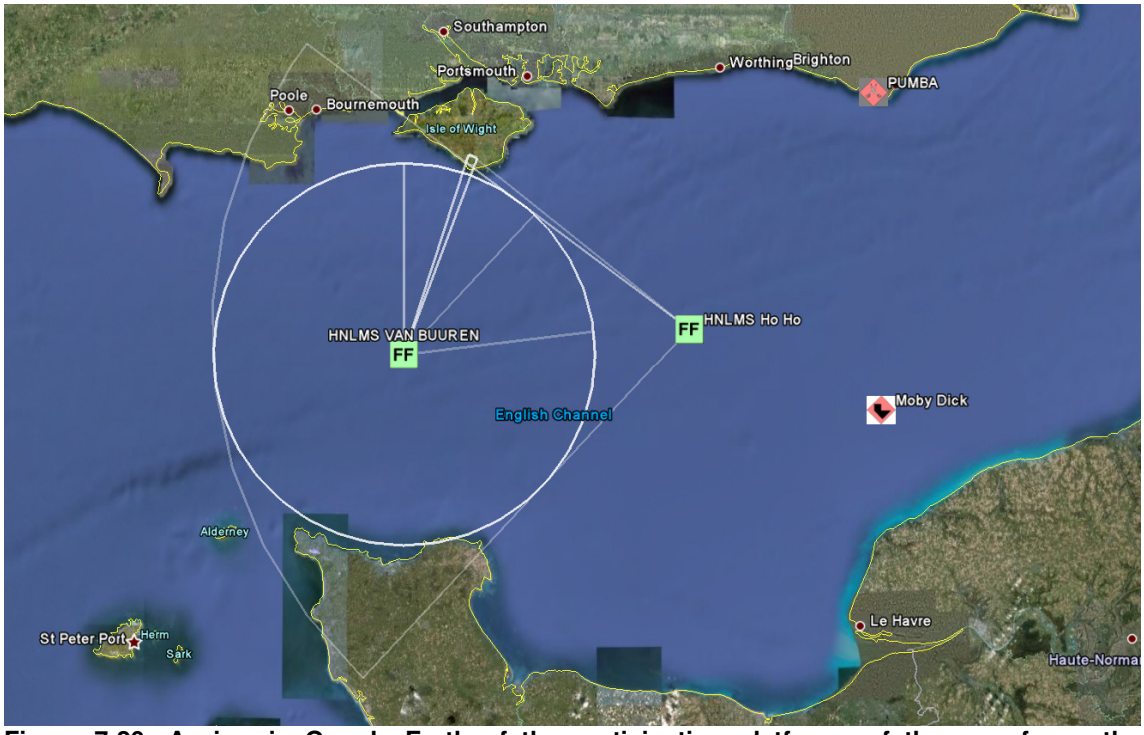


Figure 7-20: A view in Google Earth of the participating platforms of the own force, the operational tasks (one of the operational tasks involves a relatively small region around the COBRA which has been left out of the view to make the region of the operational task visible) and TaskAlternative-objects at step 1 of the STM-simulator.

```
Capability: 42(HNLMS Ho Ho)TaakId's: [11]Bfrom: 223.85340118408203Bto: 307.4638552069664Rto: 124983.5078125
Capability: 42(HNLMS Ho Ho)TaakId's: [12]Bfrom: 307.49657440185547Bto: 309.37336403131485Rto: 73216.546875
```

Figure 7-21: An alternative at step 1 of the STM-simulator.

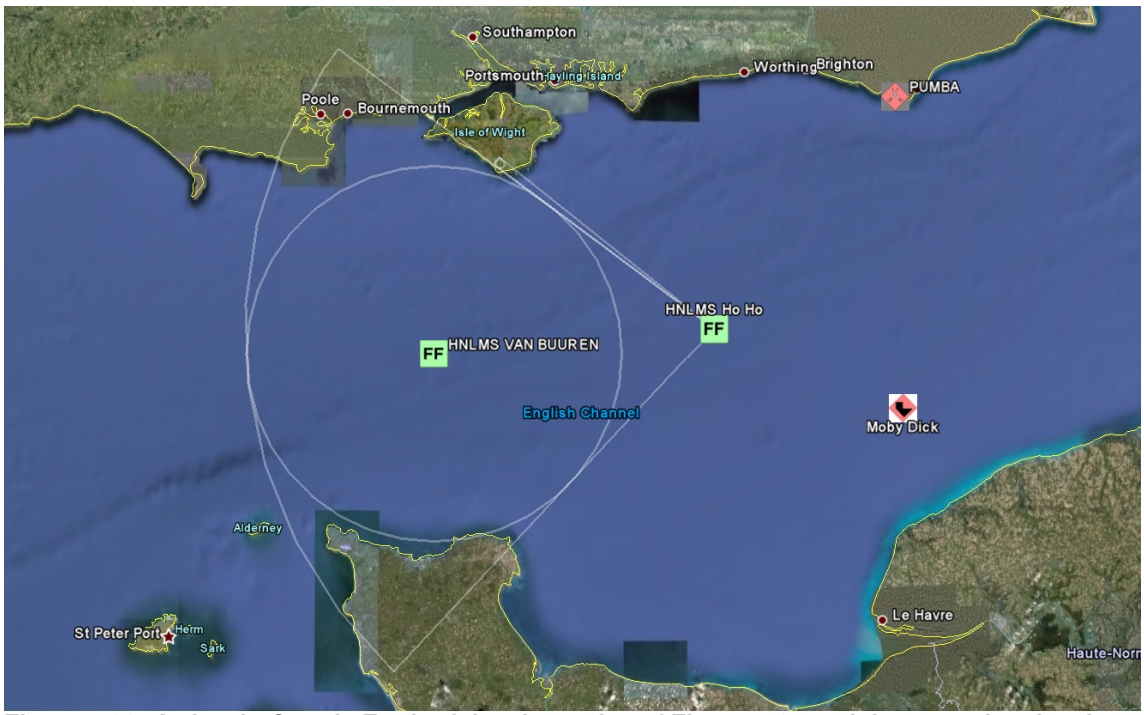


Figure 7-22: A view in Google Earth of the alternative of Figure 7-21 and the operational tasks at step 1 of the STM-simulator.

Task ID	Score	Weight
11	0.683912663707338	0.6666666666666666
12	0.3965640985778726	1.0

Totale score: 0.5115035246296589

Figure 7-23: The scores of the alternative given in Figure 7-21 for each of the operational tasks in Figure 7-18 together with the weights of these tasks and the total score for the alternative.

Capability: 3(HNLMS VAN BUUREN)TaakId's: [11, 12]Bfrom: 17.909788250923157Bto: 20.47358500957489Rto: 54839.72265625
Capability: 42(HNLMS Ho Ho)TaakId's: [12]Bfrom: 307.49657440185547Bto: 309.37336403131485Rto: 73216.546875

Figure 7-24: An alternative at step 1 of the STM-simulator.

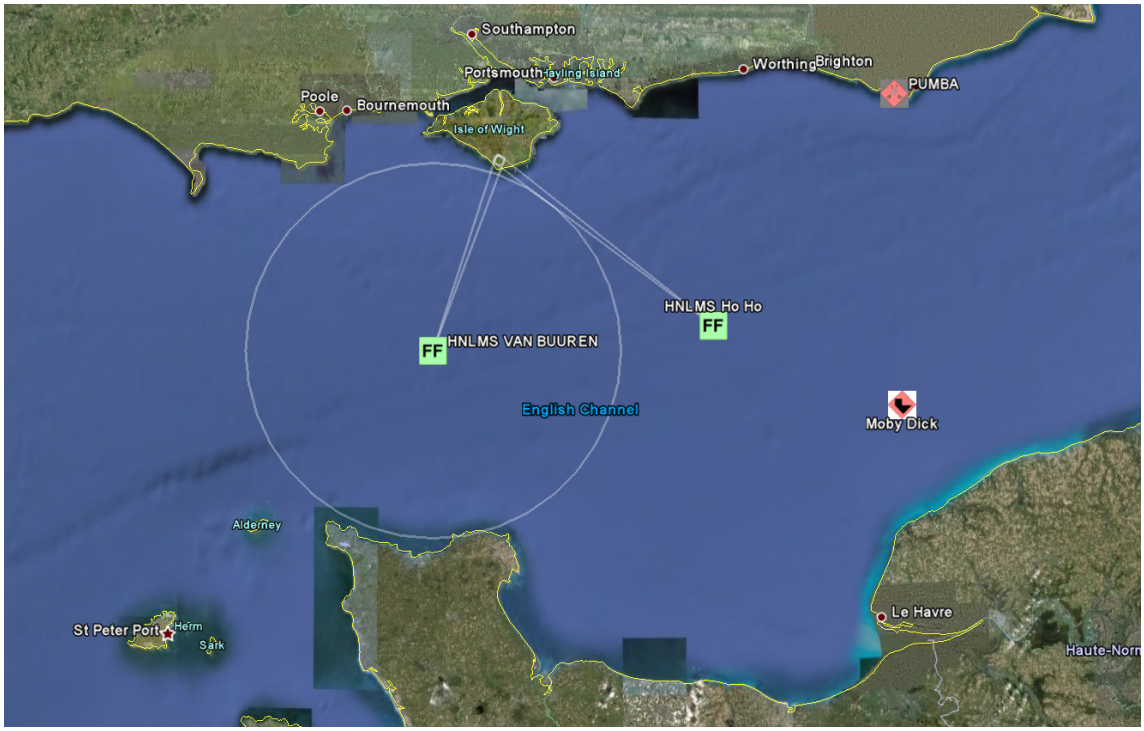


Figure 7-25: A view in Google Earth of the alternative of Figure 7-24 and the operational tasks at step 1 of the STM-simulator.

As with simulation step 0, there are alternatives, see Table 7-6, that give higher scores (< 0.1% higher) than the one with all of the building blocks, which gives a score of 0.935163427017035. Here again, only a small number of alternatives needs to be evaluated to obtain a high score.

Table 7-6: The highest scores for different sets of alternatives at step 1 of the STM-simulator.

n	Highest score for the first n alternatives	Highest score for the last n alternatives	Highest score for n random alternatives
10	0.5115035246296589	0.9358946691403647	0.9340516624939362
100	0.5222818563768452	0.9358946691403647	0.9346416304281571
1000	0.5799358326642978	0.9358946691403647	0.9358946693614763

When we compare the time needed to evaluate alternatives during this step, see Table 7-7, with that of the previous step (Table 7-4), we see that the difference is quite small for the first n alternatives. This might be a bit unexpected, since the alternatives contain the exact same amount of building blocks and we now have

twice as many operational tasks to evaluate these alternatives on. On the other hand, the second operational task only concerns a very small area that can only be covered by a few CapabilityAllocation-objects, see Figure 7-19 and Figure 7-20. With the other subsets of alternatives that were evaluated, it takes more than twice as long now. Since the alternatives have more building blocks (on average) and there are twice as many operational tasks at step 1, this is not that surprising. Assuming that the 1000 random alternatives that were considered are representative for the whole set of alternatives, it would take about 326 days to evaluate all 2^{26} alternatives. When looking at the number of operational tasks and TaskAlternative-objects, we could say that the size of the problem at step 1 is about twice that of the problem at step 0. Just like that, the problems at the last three simulation steps (see Table 7-1) are about twice as big as the problem at step 1. When assuming that it also takes about twice as long, on average, to evaluate an alternative during these steps, it means that evaluating all alternatives in the STM-simulator like this will take over a billion (10^{12}) years for sure.

Table 7-7: The time it takes to find the scores given in Table 7-6.

n	Time to compute the highest score for the first n alternatives	Time to compute the highest score for the last n alternatives	Time to compute the highest score for n random alternatives
10	1 s	7 s	5 s
100	14 s	67 s	43 s
1000	200 s	638 s	420 s

8 Discussion and Future Work

This thesis considered the problem of finding the optimal sensor usage in the STM-simulator. After observing the most important factors and analyzing the problem situation, different approaches were considered, of which MCDM was found to be the most suitable on account of the clear steps it comprises. The MAUT-procedure, chosen to be used, has been worked out for the considered case and implemented in the system.

We have examined two simulation steps and have found that with the largest steps of this quite simple scenario in the simulator we come across so many alternatives that it would take more than a year, very mildly put, to evaluate them all. On the other hand, very high scores (similar to the highest possible score, given the circumstances) can be found by considering only a small selection of the alternatives.

Several assumptions have been made to simplify the problem and make the implementation more manageable, while still maintaining a proper idea of the problem situation and chosen algorithm. First of all, there is the issue of the considered alternatives. These were formed by combinations of building blocks of the form (c, p), where c is a sensor capability and p is a set of parameters. To include all alternatives worth considering, in principle, sensor capabilities of other sensors also need to be taken into account when determining the alternative building blocks for a certain sensor (see pages 32 and 33 for details). This was omitted, however, since it would make the problem significantly larger. As a consequence, the actual optimal sensor usage might not be among the considered choices. Changing the determination process of the alternative building blocks for one sensor to make it consider sensor capabilities of other sensors is not very difficult. It does demand for a whole lot more computing time when computing the sensor usage, though. As for the implementation, only two dimensions were considered. In theory, extending to 3D is rather simple since elevation (which was left out) can be treated in a similar way as bearing. Nevertheless, in practice it would be a lot easier if a Java class was available for representing regions in 3D. Furthermore, the considered surface around the earth, being an irregular shape approximating an ellipsoid, is approximated by a flat, rectangular surface. With this, the region of interest is presumed to be (relatively) small (compared to the surface of the earth), which is a valid assumption with current military operations. It is evident that this simplification for determining the region coverage is made at the cost of accuracy. It results in noticeable faulty positions with regard to one another (except for positions with regard to the first platform that is used as the reference point), which also means that the coverage regions are determined inaccurately. When it comes to the point that accuracy becomes critical, the region coverage needs to be determined in some other way. Another issue that appeared to cause inaccurate results is the use the Monte Carlo method for the computation of areas. For more accurate results, more points could be used or another method should be considered.

As demonstrated by the results, the number of alternatives in the simulation is too large to deal with within a reasonable amount of time. The implementation can be made more efficient, though, by using the PerformanceMap2D for an operational task and the alternative with all of the building blocks as a basis for the rest of the

PerformanceMap2D-objects for the same operational task. However, this still would not make a large enough difference. Even if it would only take 1 ms per alternative and we would have a billion times the computing power we have now, it would still take thousands of years to go through all simulation steps. Several possible measures were mentioned in Section 6.3 for keeping the computing time down and two very simple options have been implemented. Using more computing power, possibly by distributing the evaluation of alternatives over a number of computers, is a very good solution for finding the best alternative within a significantly shorter amount of time. Another workable solution that can perform very well is using some temporary sensor deployment while the system continues evaluating alternatives, and changing the sensor usage when a significantly better alternative is found. Other solutions could also be implemented. It is, basically, a trade-off between speed and quality. Of course, different options could also be combined. Another way of keeping the required computing time limited, is to avoid having too many alternatives. This can be done by not partitioning the regions over which sensor capabilities can be used or only partitioning them to a certain extent, when making the building blocks for alternatives. The extent to which these regions should be partitioned is a question of whether the added value is worth the additional computing time. What could also help when it comes to efficiency and reducing the computing time required, are leaving out the alternatives with redundant building blocks and joining the building blocks of the considered alternatives before evaluating them, as mentioned in Subsection 5.2.1. Besides speed and efficiency, there are other factors that can be taken into account for improving the performance of the simulator. One such factor still to be implemented is the radar horizon (which lies somewhat farther than the visual horizon on account of the transmitted waves being bent to some degree around the earth). Even when a sensor had infinite range, it could not cover regions past the radar horizon.

For the rest, there remain some matters that still need to be established to allow for better results to be obtained. Among these are functions for the region coverage and performance scores and weights to be used for computing scores that reflect the vision of the Royal Netherlands Navy. Furthermore, the creation of alternatives from the constructed building blocks can result in quite fragmented sensor (capability) usage. This leads to the question of to what extent this is desirable. Then there is the influence of meteorological factors on the sensor usage of the effect of it that is still to be determined.

Finally, there are several elements left to be implemented and, of course, accounted for in the determination of the sensor usage to make the simulation complete. These include the switching time and costs of a sensor. To determine whether a switch of modes is required, the current mode of the sensor also needs to be known. Furthermore, the sensor load corresponding with each of the sensor capabilities needs to be specified, the form of preference rules needs to be defined and the intra- and inter-sensor constraints and geographical information need to be added in the simulator.

In conclusion, using the MAUT-procedure in the STM-simulator does not guarantee to give the optimal sensor usage (within a reasonable amount of time). However, when it is clear how different ways of sensor deployment are to be rated and all

relevant factors are taken into account, results can be found that come quite close to being optimal.

References

[1] Rudolf Muller, Andres Perea, Sascha Wolf, *Combinatorial Scoring Auctions*, Research Memoranda 020, Maastricht: METEOR, Maastricht Research School of Economics of Technology and Organization, June 2007.
[2] U. Kaymak en J.M. Sousa, *Weighted Constraints in Fuzzy Optimization*, Erasmus Research Institute of Management, March 2001.
[3] Thomas Verhoeff, *Decision Support for Naval Mission Planning*, Hengelo: Thales Nederland B.V., October 2007.
[4] Detlof von Winterfeldt en Ward Edwards, *Decision Analysis and Behavioral Research*, Cambridge [etc.]: Cambridge University Press, 1986.
[5] Ronald R. Yager, *Modeling Prioritized Multicriteria Decision Making*, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 34(6):2396—2404, 2004.
[6] P. Espie, G.W. Ault, G.M. Burt en J.R. McDonald, *Multiple criteria decision making techniques applied to electricity distribution system planning*, Generation, Transmission and Distribution, IEE Proceedings, 150(5):527—535, 2003.
[7] Xiaomin M. Wang, Zhilin L. Qin en Yuda D. Hu, *An Interactive Algorithm for Multicriteria Decision Making: The Attainable Reference Point Method*, 2001.
[8] Po-Lung Yu, Yoon-Ro Lee, Antonie Stam, *Multiple-criteria decision making: concepts, techniques, and extensions*, New York, N.Y.; London: Plenum, 1985.

Appendix A: The STM-Demonstrator

Within the Above Water Systems - Lead System Integration department at Thales Nederland, several demonstrators have been built to show possible future projects. One of these demonstrators is a simulator for naval missions that gives the user a recommendation on the usage of the available sensor suite in different situations. Using the graphical user interface, the system output can be viewed in different levels of detail.

The activity diagram of the STM-demonstrator is discussed below, followed by a description of the actual application.

Activity Diagram

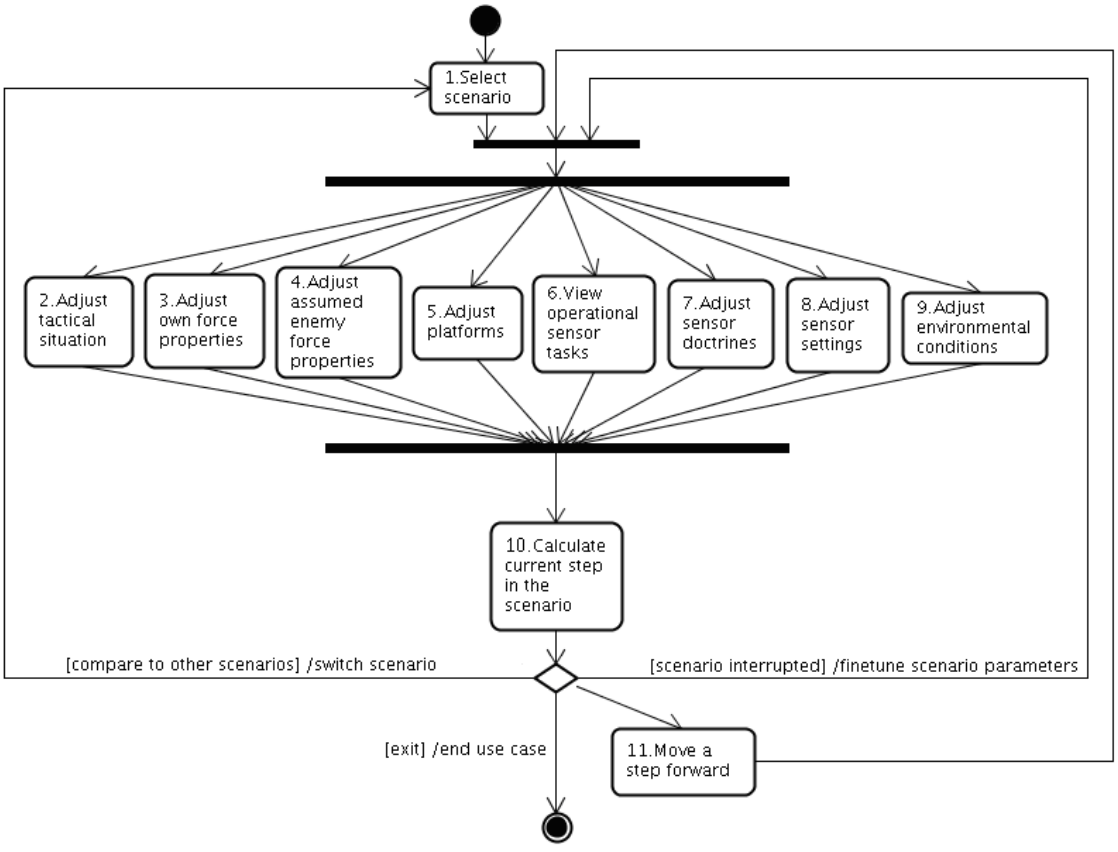


Figure 8-1: STM-demonstrator activity diagram.

The activity diagram in Figure 8-1 gives an overview of the possible user actions with the STM-demonstrator, aside from examining scenario information and the system output. A description of the activity diagram events follows.

1. Select scenario
The user selects one of the available scenarios.

2. Adjust tactical situation
The user activates or deactivates EMCON plans and/or ROE. They can also view operational messages and events.

3. Adjust own force properties
An overview of the own platforms and events is shown. The user can activate or deactivate each of the own platforms.

4. Adjust assumed enemy force properties
An overview of the assumed enemy platforms and events is shown. The user can activate or deactivate each of the enemy platforms.

5. Adjust platforms
A detailed view of the own platforms and their weapons, sensors and events is shown. The user can activate or deactivate each of the sensors.

6. View operational sensor tasks
An overview of all current sensor tasks is shown. The user can activate or deactivate each of these tasks.

7. Adjust sensor doctrines
Sensor doctrines are shown, ordered by doctrine group. The user can activate or deactivate sensor doctrine groups and/or sensor doctrines.

8. Adjust sensor settings
For each sensor of the own platforms, sensor settings are shown. The user can activate or deactivate each of these sensor settings.

9. Adjust environmental conditions
Environmental settings such as sea state, humidity and weather type are shown. The user can change these settings.

10. Calculate current step in the scenario
The user starts the process of determination of the sensor usage, which will be shown once it is determined.

11. Move a step forward
The user moves to the next step in the scenario. The appropriate information corresponding to this (next) step will be shown.

After calculating the sensor usage for a certain situation (step 10 of the activity diagram), at the diamond, the user can choose from four options:

- Change the scenario settings (steps 2 to 9), to recalculate the sensor usage.
- Choose another scenario (step 1) and continue from there.
- Move on to the next step in the scenario (step 11).
- Quit the simulation tool.

The Demonstrator

The simulation tool is built in Java, with JESS (Java Expert System Shell) and XML. It is being used on a Pentium IV-class Dell laptop with 512 MB RAM, under Fedora.

Figure 8-2 shows the interface of the demonstrator, which consists of four segments:
1. *Illustration of the current situation;* shows an image corresponding with the current step in the scenario.
2. *Scenario control;* shows the scenario and current step, the button that starts calculating the sensor usage and the button for going to the next step.
3. *Situation, rule and doctrine control;* shows several tabs with all information about units in the current situation, environment, weather, rules and doctrines.
4. *Sensor deployment;* the sensor deployment can be viewed here after calculation.

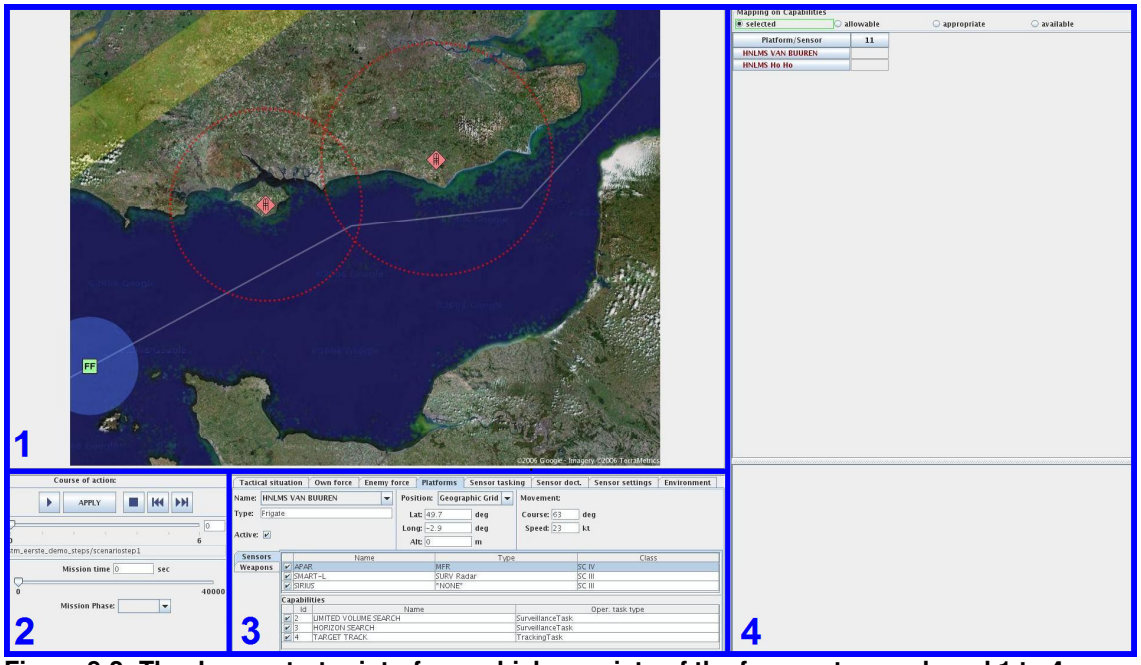


Figure 8-2: The demonstrator interface, which consists of the four parts, numbered 1 to 4, outlined in blue.

The scenario is described in XML-files. This includes scenario steps, geography of the theatre, tactical situation (including ROE and EMCON), information about own force and enemy force (such as units, trajectory, sensor suite and weapon suite), operational sensor tasking, doctrines and environmental conditions. For the rules that come with this information, JESS is being used. These data and rules, which can be viewed in detail in area 3 from Figure 8-2, are used to determine the sensor deployment.

Area 4 from Figure 8-2 allows the user to view the determined sensor deployment in different levels of detail. The table displayed there can be expanded to show exactly which sensor functions, from which sensors, (partly) cover which of the operational tasks. Furthermore, the user can view the parameters values that the sensor functions are deployed with.