# Rendezvous Planning for Multiple Autonomous Underwater Vehicles using a Markov Decision Process

Veronika Yordanova[1,*], Hugh Griffiths[2], Stephen Hailes[3]

[1]Atlas Elektronik GmbH
[2]University College London, Department of Electronic and Electrical Engineering
[3]University College London, Department of Computer Science
[*]veronika.yordanova.11@ucl.ac.uk

**Abstract:** Multiple Autonomous Underwater Vehicles (AUVs) are a potential alternative to conventional large manned vessels for mine countermeasure (MCM) operations. Online mission planning for cooperative multi-AUV network often relies or predefined contingency on reactive methods and do not deliver an optimal end-goal performance. Markov Decision Process (MDP) is a decision-making framework that allows an optimal solution, taking into account future decision estimates, rather than having a myopic view. However, most real-world problems are too complex to be represented by this framework. We deal with the complexity problem by abstracting the MCM scenario with a reduced state and action space, yet retaining the information that defines the goal and constraints coming from the application. Another critical part of the model is the ability of the vehicles to communicate and enable a cooperative mission. We use the Rendezvous Point (RP) method. The RP schedules meeting points for the vehicles throughput the mission. Our model provides an optimal action selection solution for the multi-AUV MCM problem. The computation of the mission plan is performed in the order of minutes. This quick execution demonstrates the model is feasible for real-time applications.

## 1. Introduction and Background

Mine Countermeasures (MCM) is the problem of finding and disposing of naval mines. Mine hunting is a common method used for MCM. It relies on detecting and classifying a target on the sea bottom, using a sonar sensor, followed by an appropriate disposal procedure. Often a manned surface vessel is employed in the mine hunting task. There are disadvantages of this traditional method. For instance, the personnel are at risk, the platforms require complex design and are expensive, and the procedure is time consuming and inflexible, as the sensor is coupled to the platform.

Recent advances in AUVs have made them a viable option for a safe, and potentially cost-effective and time-efficient alternative to conventional mine-hunting platforms. Many AUV designs have reached maturity, which allows a considerable amount of research to focus on robotic networks underwater [1], [2], [3], [4]. A distributed system allows for the nodes to be dynamically tasked with different functions during the mission. It brings robustness in the case of a node malfunctioning, and could improve time and area-coverage performance. Utilising multiple robots has applications in science, industry and defence, such as adaptive sampling, surveying deep waters for oil and gas deposits, and surveillance. This study focuses on the MCM problem and the remainder of the paper will consider the implications of using an AUV network for mine hunting. However, the methods and solutions are transferable and can be utilised in other applications in the underwater domain.

One common problem experienced in AUV applications is communications. The physical wave propagation limits in water exclude the electromagnetic spectrum from most common shallow water applications. Even though light can propagate for larger distances in deep waters, in shallow waters, transferring data is possible for up to a few metres, where blue-green laser is a promising technique [5]. If a connection requires larger distance, acoustic modems are the common choice with which to equip an AUV. However, there are hard limitations imposed by the physics of the sound wave propagation and the ability to transfer data underwater is very different from terrestrial capabilities. The major difficulties arise from the slow sound speed in water, attenuation with increasing the distance and frequency, and variability of the channel quality. This brings limited bandwidth, low data-rate, and loss of connectivity [6]. A considerable amount of research has focused on the underwater communication problem and many new solutions are pushing the boundaries of acomms [7]. However, the underwater channel is so adverse, that often networks are considered impractical. Even for a single vehicle operation, the unstable and unreliable channel means a constant risk of losing the platform when operating in autonomous mode. Ability to communicate is vital for collaboration, since this is the major advantage a network brings.

The data exchange problem is solved by bringing the nodes together. This approach has been investigated before for the application of multi-vehicle MCM [8], [9]. The Rendezvous Point (RP) method relies on adaptively scheduling meeting points for the AUVs performing an MCM mission. The disadvantage is that the vehicles need to travel to the RPs and hence some resource is lost. The advantage is that by bringing the nodes together, a reliable connection is available and the agents can adapt their next meeting point to the emerging workload, as well as reallocate their tasks. In [9], a numerical simulation evaluates the loss and the benefits of applying the RP method. It concludes that by enabling adaptive reallocation, the vehicles can be utilised at constant rate, which makes the meeting loss acceptable. However, the RP method described above relies on heuristics rules which only allow for myopic decision making. In order to move from an adaptive to intelligent planning, the Markov Decision Process (MDP) framework is used in this work.

Section 2 explains the MDP framework, gives the theoretical formulation and notation, and provides some recent advancements in applying the method. It also points out the contributions of the paper by applying MDP to the multi-vehicle MCM problem. Section 3 deals with modelling the multi-vehicle MCM scenario and confining it to the MDP framework. It explains the assumptions and limitations of the approach. Section 4 focuses on the solution method for our MDP, value iteration, and provides the simulation results and analysis. Finally, Section 5 concludes with a summary and further possibilities for expansion and application of the method.

## 2. Mission Planning for MCM Problem Formulation

In this section we explain the typical stages of a mine hunting operation, using AUVs for MCM, the rendezvous point method, and the scenario used for the simulations. We also provide the theoretical formulation of the MDP framework. Combining the application and the method, we managed to model an intelligent planning scheme for MCM using multiple AUVs. The main contributions are stated at the end of this section.

### 2.1. Application Specifications and Constraints - MCM scenario

The five phases in mine hunting are [10]:

- Search - Scan an area for mine-like objects using a sonar sensor.
- Detection - A contact is acknowledged based on the sensory data and its location recorded.
- Classification - Put the detected contact into one of two categories: mine-like object (MLO) or non-mine like object.
- Identification (ID) - Determine the type of the MLO
- Neutralisation - Depending on the location of the MLO and the purpose of the mission, a target is considered neutralised if it can be successfully localised and avoided, destroyed or disabled.

Since this work is concerned with the planning problem, some important MCM parameters and capabilities are treated as a black box and assumptions are made about their performance. The constraints we take into account from the MCM problem are:

- Search: 1) vehicles move in a typical lawnmower pattern to secure complete area coverage and 2) the sensor defines the swath width, and hence the distance between the lawnmower legs, and the speed of the vehicle, in order to collect undistorted data. The limitations coming from currents and navigation error are ignored. Research on optimal coverage, navigation and sonar imagery signal processing is ongoing but these problems are not addressed in the current work.
- Detection: The probability of detection is an important performance measure in MCM, however for this work we assume deterministic detection output passed to the MDP planner.
- Classification, Identification and Neutralisation: The classification and identification steps require collecting further information of the detected target. Depending on the available sensor, or restrictions based on established doctrines, this includes relocation of the target and collecting additional imagery. While we are not concerned with the processing of the sensory data, or the accuracy of target positioning and the collected data quality, we assume all recorded targets during the detection phase need to be relocated to allow additional data collection. This procedure defines another path planning specification for our scenario. The Neutralisation phase is not considered in this work, as usually a Remotely Operated Vehicle (ROV) is used..

The MCM scenario we are studying uses 3 AUVs. For the limited battery time they have, their goal is to maximise the area they can search with the constraint that every detection has to be revisited. Two tasks are defined: search and ID (relocating targets and collecting additional data to help classify and identify the object). Following the RP method idea, the vehicles meet so they can exchange target locations, decide how to allocate tasks and appoint time and location for their next meeting point. The RP schedule and the MCM application goal constraints define the input functions for the MDP framework.

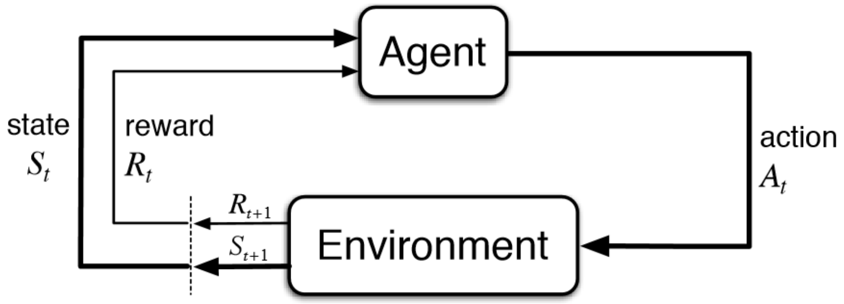

***Fig. 1.*** *Reinforcement learning: agent-environment interaction [14].*

## 2.2. *Method Basics and Advancements: Markov Decision Process and Rendezvous Point*

MDP is a common framework used to model intelligent decision making. The two problems it solves are reinforcement learning (RL) and planning. The recent success story of DeepMind's Alpha Go agent applying RL to the game of Go [11], and the rapid advancement of deep learning methods [12] that allow processing of large state spaces, has proven the ability of modelling complex problems relying on the classical MDP framework variations. The difficulty of RL in real life is that agents need to develop an understanding of the environment based on multiple sensor inputs and use it to transfer current skills to new situations. On the other hand, planning assumes the environment representations are already derived and the agent can proceed to finding the optimal action policy. In both instances, modelling the agent-environment interaction often creates huge state and action spaces which leads to unmanageable computation load. This is especially true for problems that require real-time processing or have limited resources [13]. Figure 1 gives the basic elements of a RL agent. This structure is used for modelling the multi-AUV MCM problem and the notation used further is extracted from [14].

Reducing the complexity of an MDP model and creating a manageable state and action space is a well known problem and has been described in [16]. There are different methods proposed in literature on how to handle MDP complexity. Using domain knowledge to take advantage of a problem's special features has been recognised and used in [17], [14]. A complex problem of multiple elevators' scheduling was simplified by carefully defining the performance measures, while retaining the application's goals. This is also the concept used for this paper - using the domain knowledge of the application allowed to confine the problem into a manageable MDP model. Most recent methods aim to reduce MDP complexity using approximation techniques that do not go through exhaustive state/action space search as the solution used in this paper. The choice of states to be processed can be based on predefined weights or constraints due to time [18]. One issue with such approximate techniques is that the solution is near optimal and can lead to a local maximum solution, instead of a global one. Sometimes, a heuristic approach might allow for a greater control over the model by defining how deep the search space need to be explored in order to make the solution reliable for the application's needs. One such example was studied in [19]. Another method proposes a tradeoff of reducing dependence on state space size but on the expense of increasing the horizon space dependence [15]. A summary of multiple methods dealing with reducing MDP complexity can be found in [20]. While a classic optimal solution, value iteration, was chosen for this work as a proof of concept, all of the methods described in this section or in

[20] might bring a significant improvement to reducing the complexity and processing time of the MDP model. Recent advancements in the field are reached by using learning instead of providing a transfer and reward functions (described in sections 3.2 and 3.3), as well as using Partially Observable Markov Decision Process (POMDP) [21], where the state space is non-deterministic, making the modelling more realistic by taking into account noisy sensors.

The MDP model in this work is defined using the Rendezvous Point (RP) method idea from [9]. The classic Rendezvous problem is trying to find what strategy multiple agents should choose in order to maximise the probability of meeting each other. The RP method uses some of the basic principles from the original Rendezvous problem, such as no active communication between agents, sensing region restrictions, collective behaviours, meeting at an unspecified location [22]. The difference is that the RP goal is not for the nodes to ”discover” the rendezvous, but to ”agree” on it. Where and when to meet is unknown at the beginning of the mission, but the agents decide dynamically how and when to schedule this. Thus, the RP method differs from previous work and techniques suggested for the original Rendezvous problem.

The method we propose in this paper, MDP using RP, is relevant to applications and approaches in collaborative robotics, cooperative exploration, search and rescue, swarms, sensor networks. Often, rule based or reactive solutions are used for changing behaviours or tasks of the vehicles. This paper suggests simulation results and modelling for optimal multi-vehicle task allocation.

### *2.3. Contributions*

The main contribution of this paper is the representation of a multi-AUV MCM planning problem with a very small state space MDP. This allowes using an optimal solution method in low computation time. As a result, this MDP model has a potential to be applied to an underwater robotic network and provide a real-time output for the agents. Some key points of the model are:

- State space mapped from physical space to time to reduce dimensionality
- Action space is defined in a way that brings the problem from multi-agent to a single agent by explicit task allocation for all vehicles.
- The interval between the elements of $t$ is not always the same, as in most MDP formulations, however, the problem is still discrete and the framework applicable.

Some heavy assumptions in this model are made: 1) assuming the environment dynamics are known; 2) using a uniform distribution to represent the number and position of the expected targets; 3) approximate expected shortest path calculation; 4) sensor is treated as black box with consistent specifications.

## 3. Modelling Multiple vehicle MCM planning with MDP

There are four components to define an MDP - state, action, transition probability and reward. Each of them is defined as an input function for the MDP and is further explained in this section. The RP scheduling fits into the framework as the agents interact with the environment at a discrete sequence of steps: $t = 0, 1..T$. Every t-step is an RP.

| Action Number | N search veh | RP duration | ID time |
|---|---|---|---|
| 0 | 0 | $id/3$ | $id$ |
| 1 | 1 | $id/2$ | $id$ |
| 2 | 2 | $id$ | $id$ |
| 3 | 3 | $RP_{max}$ | 0 |
| 4 | 2 | $RP_{max}$ | $RP_{max}$ |
| 5 | 1 | $RP_{max}$ | $2 * RP_{max}$ |
| 6 | 0 | $RP_{max}$ | $3 * RP_{max}$ |

**Table 1** Action space

## *3.1. State and Action Space Model*

A state $s$ characterises where the agents are in the environment. The set of all possible states is defined by $s_t \in S$. In order to represent the multi-vehicle MCM problem with a manageable state space, we have created a mapping from physical space (the updated location of the vehicles and the detected targets) to time. The information retained in the state to fully define the agent's current relationship with the environment, is kept in a 2D state space structure.

The first dimension represents the workload. It is defined by how much time a vehicle requires to revisit all detections, so that additional data is collected, and enable classification and identification. The state is updated on every RP interval. For convenience, for the rest of the paper, this state will be referred to as 'ID time' and the process of reallocating a target 'identified'. Once a detection is revisited, it is removed from the list of workload. The ID time is additive in a way that the workload can accumulate over time, if the agents decide to postpone identifying detections. An upper limitation of 300 minutes is given to maintain a reduced state space, $id \in [0, 300)$. Using time instead of space to represent the workload means that the location of targets no longer needs to be tracked in the state space. This enables us to reduce the problem from 2D space representation to 1D time representation.

The second dimension contains the mission time limit information. It is given by the battery of the AUVs, and for our scenario this limit is 10 hours of operation with a state space from 0 to 600 minutes $battery\ state \in [0, 600)$. Thus, the full state space is defined by a tuple $s = (id, battery)$. The battery time state abstraction means that the location of the vehicles in physical space does not need to be retained in the model's state space. Their location is defined by the time they spend on search, as the pattern of their movement is known (lawnmower in a confined area), as well of their speed (a constant speed of 2 $m/s$ is assumed and all environmental disruptions are ignored). This way, the vehicles' position is retained in 1D time vector, instead of 2D x-y coordinates. The depth information is ignored for this model.

The states in an MDP can be terminal and non-terminal. A terminal state triggers termination of the episode, or the mission. In our simulation, terminal states are all states that have reached the battery state limit: $s(:, 599)$.

Action is the way the agent interacts with the environment. The set of all actions is a function of the state: $a_t \in A(s_t)$. The scenario problem we are looking into includes three agents, or three collaborating AUVs. In order to simplify the model, instead of considering multi-agent system, the action space explores all combinations of dividing the search and ID tasks between the 3 agents, as well as varying the RP time. They are defined as in Table 1 and example of the AUV paths are shown in 2.

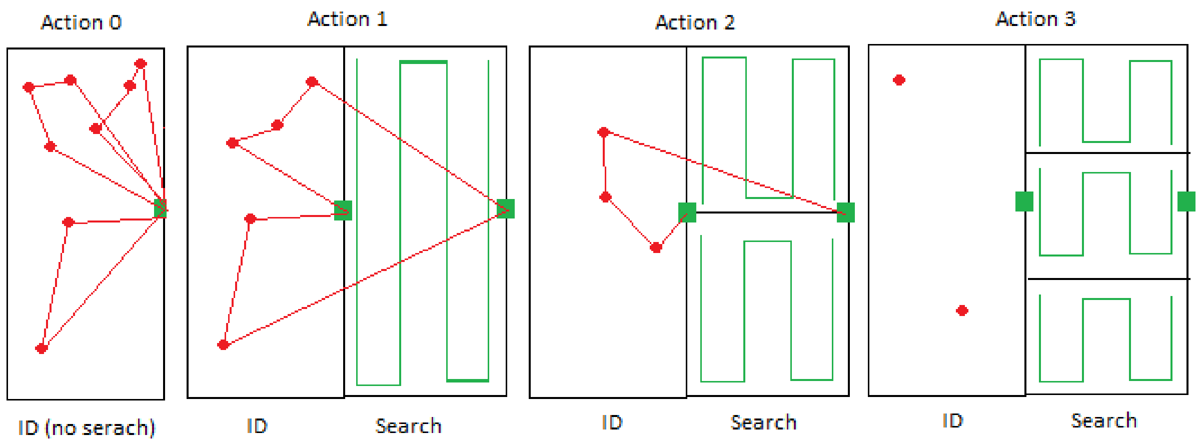

***Fig. 2.*** *AUV path examples illustrating Actions 0,1,2 and 3 from Table 1. Actions 4,5 and 6 are similar to Actions 2,1 and 0, respectively, but the RP duration is based on the ID time, not the predefined value.*

There are 7 actions in our model and a sequence number for each is given in the first column of Table 1. The second column is the number of vehicles allocated for the search task during the RP interval. The third column is the RP duration. There are two options for this parameter. It can be a predefined interval: in the model, a 1 $h$ interval is chosen - this is assumed as an interval at which an operator would like to get a confirmation of the status of the mission, if there is a surface node to assist the mission. However, the RP duration can be based on the requirements of the ID process. This is done by adapting the RP duration so that that all known targets up to date can be identified. It also varies depending on the number of vehicles that are tasked with the ID job. The RP interval choice is a tradeoff between distributing the resource more efficiently (length based on ID time, or actions 0, 1 or 2), or following strict procedure of regular meeting intervals ($RP_{max}$, or actions 3, 4, 5 or 6). The final column in the table is the ID time that is spent on the ID task.

Some limitations were imposed on the actions in order to comply with the index limits of the state space. The first one is due to how our expected target model is tied to the search space, as explained later in Figure 3. This creates a problem for Action 2, where the RP limit is 299 $minutes$, and 2 vehicles are sent to perform search task. This brings too many targets for the next time interval and the state is out of bounds. Hence a limit of 200 $minutes$ was set only for Actions 0, 1 and 2 (although only Action 2 was usually affected by it), and the excess of ID was handled as credit for the next time period. This is a limitation due to the ID dimension of the state space.

The battery state space dimension also requires an exception and this is our second action limitation. To guarantee that all actions lead to the termination state, $s[:, -1]$, all actions adopt the behaviour of Action 0, once the sum of the current battery state with the expected RP duration exceeds the termination state index. For example, if we are in battery state $s(:, 580)$, then the maximum RP duration can be $599 - 580 = 19$ $minutes$, and all vehicles are set to do an ID task.

The imposed action limitations are consistent and no loss is introduced to the model. Everything that is trimmed at the current action is transferred to the next, and penalty is applied, following the reward model. If the agent constantly chooses an action that carries over workload to the next time period, at the end of the mission this results in a penalty and hence this is an undesirable behaviour. Further details on such tradeoffs are explained in the reward subsection.

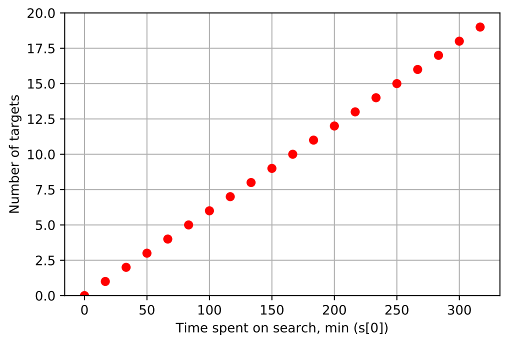

***Fig. 3.*** *Expected average number of targets as a function of time spent on search task.*

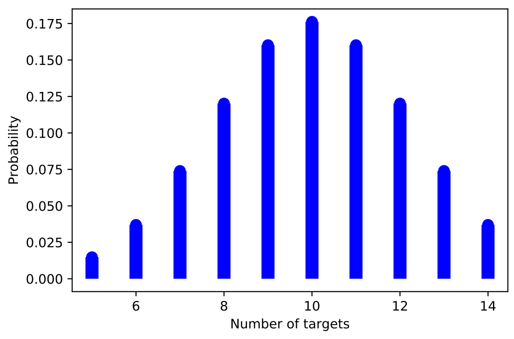

***Fig. 4.*** *Sample from the transition probability model using binomial distribution.*

## *3.2. World Model Approximation*

The world model is the agent's representation of the environment's dynamics. Equation 1 shows how any state and action define the probability of the possible future state $s'$.

$$P^a_{ss'} = Pr\{s_{t+1} = s'|s_t = s, a_t = a\} \tag{1}$$

We assume this transition dynamics is available for our scenario. The next state, $s'$, is defined by ID and battery time at the next RP. To be able to predict it based on current action and state, we have developed a function to link search time, number of detections and shortest path to revisit them.

We assume the search time is proportional to the number of detections that will be found. The time is transferred to search area by multiplying it to the vehicle's speed and sensor swath. We define an expectation of average number of targets (10) per unit area (1 $km^2$), as well as uniform distribution of the targets in $x$,$y$ plane. Then the area defined by the search time is multiplied by the number of targets per unit area. This model is unrealistic but since modelling of the mine field is not the purpose of the project, and details of such distributions are rarely disclosed in open literature, such model is considered acceptable. The MDP and the planning algorithm will still be valid, if an updated environment model is applied at a later stage.

We assume a stochastic state transition function which has 10 different possible future states, $s'$. We represent it by a binomial distribution defined by $mean = n * p = (2 * \mu) * 0.5$. $\mu$ is given by the expectation of targets per search area. Figure 3 shows the linear relationship chosen for time spent on search ($x$ axis) and the expected average number of target detections ($y$ axis). A sample derived from the transition probability function can be seen on Figure 4, where the $x$ axis shows expected number of targets and the $y$ axis gives the expected probabilities. A total of 7 such transition dynamics are produced for each state $s$, depending on the action, $a$, for which the function is being calculated.

From the described model, a prediction of the number of targets for the next RP interval is obtained. Then, this needs to be translated into future state space, $s'$. To do this, uniformly distributed points in a 2D plane need to be visited in a shortest path - this is the ID task, from where the time is extracted to define $s'$. This problem is an instance of the travelling salesman problem (TSP). It is an NP hard problem, however some approximations are available. The approximation chosen for this model is the formula presented by Beardwood in [23]:

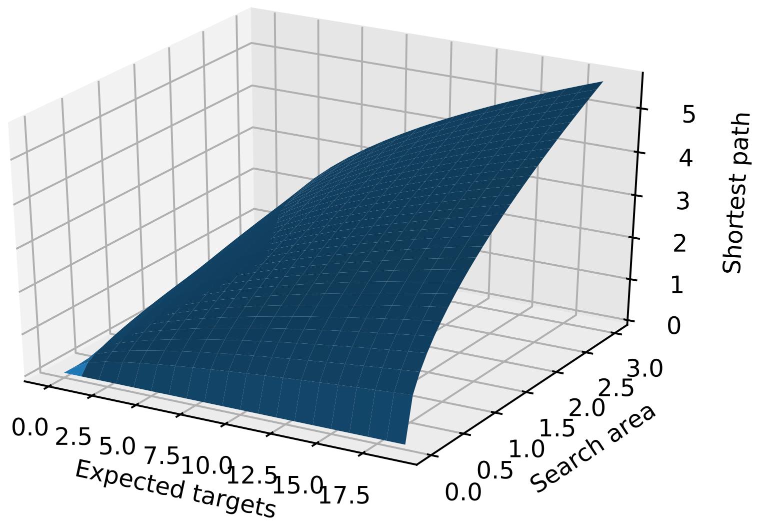

***Fig. 5.*** *Average expected shortest path in $km$ (axis Z) as a function of number of targets (axis X), uniformly distributed within a search area in $km^2$ (axis Y). Calculations follow Equation 2.*

$$A = K * \sqrt{n * area} \tag{2}$$

where A is average path length, K is an empirical coefficient ($K = 0.765$), n is the number of targets, given from the predictive model presented before, and $area$ is the search area for $s'$. The last step is to transfer the average path length from distance to time by dividing it by the speed of the vehicles. Constant speed of $2\ m/s$ is assumed. The shortest path in $km$ ($z$ axis) as a function of expected number of targets ($x$ axis) and search area in $km^2$ ($y$ axis) is presented in Figure 5, following Equation 2. This distance is then divided by the speed of the vehicle to get the time for $s'$.

One limitation of this approach is that each of the 10 results for $s'$ per action $a$ need to be multiplied by a normal distribution in order to account for the unlimited options the approximation from Beardwood [23] gives. This aspect is ignored at this stage. What is lost in this omission is precision in the expectation of shortest path time. However, since the model dynamics is not based on a realistic model, this precision is already lost. In case a real target distribution is applied, then accurate boundaries of the shortest path expectation should be studied further and this will add value to the solution.

### 3.3. Reward Function Model

The reward is the second element that models the environment dynamics, together with the transition probability function. It represents the agent's preference. We introduce the MCM mission goal to the agents by rewarding them for the behaviour we want to encourage and penalising them for deviations or unwanted mission results. The equation that defines the expected reward an agent will receive in the next time step, $r_{t+1}$, given the current state $s$, the choice of action $a_t$ and the

| Name | Unit | Function |
|---|---|---|
| Area | 1 min time on search | y = x |
| RP penalty | 1 min deviation from RP | $y = \frac{x}{RP_{max}} 100$ |
| Credit | 1 min ID time carried over | if $x > 0, y = 0.01x^2$, if $x < 0, y = 0.1x^2$ |

**Table 2** Reward function

next state $s'$ it is expected to reach, is:

$$R^a_{ss'} = E\{r_{t+1}|s_t = s, a_t = a, s_{t+1} = s'\} \tag{3}$$

Optimal actions are selected by calculating the expected reward for the whole duration of the mission, rather than a myopic single state transition. The discounting factor $\gamma$ is introduced in the MDP model to account for the uncertainty that comes from calculating expected reward [14]. $\gamma$ can take values in the interval $[0, 1]$. When $\gamma = 0$, the future reward is not taken into account and the agents take myopic decisions. When $\gamma = 1$, all future rewards have the same weights as the most recent ones. Since our wold model is stochastic, we select $\gamma = 0.9$, which means that we discount the future rewards, but it still adds weight to the current decision. Equation 4 shows the expected reward for the whole mission duration.

$$R_t = \sum_{k=0}^{T} \gamma^k r_{t+k+1} \tag{4}$$

$R_t$ is the expected reward at time $t$, $k$ is the number of future time interval, $k \in [0, T]$. For out MDP model, $k$ is each RP. $r_{t+k+1}$ is the reward for each future time interval $k$ we evaluate, starting from the current moment $t$.

The reward function has to account for the desired tradeoffs dictated from the MCM application and the RP method. We have come up with an equation to represent the reward, based on the desire to bring certain behaviour into the agents. This model can be further improved when concrete specifications are available. For our simulations, the used elements and their weights are in Equation 5 and Table 2:

$$R = area - penalty - credit \tag{5}$$

where R is the reward, which increases when we gain more search area ($area$ in the equation), $penalty$ is for deviating from the preferred RP interval defined in the action section, $RP_{max}$, and $credit$ is the id work that was left unfinished from previous search area.

Figures 6 and 7 show a sensitivity analysis of the reward model. Both the area and RP penalty depend on the RP interval, which is varied on the $x$ axis of Figure 6. The credit penalty depends on the reminder of time credit from one RP to the next. On Figure 7, the credit penalty is tenfold for negative credit as this means vehicles are being idle. This happens when agents are given a longer time window for ID task, while there is lower amount of ID workload. When the credit is positive, this means the agents are accumulating ID time that was not finished at the current RP.

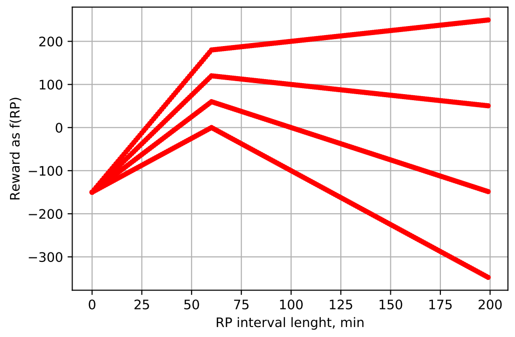

***Fig. 6.*** *Reward as a function of RP duration - area gain time and penalty for exceeding the expected RP.*

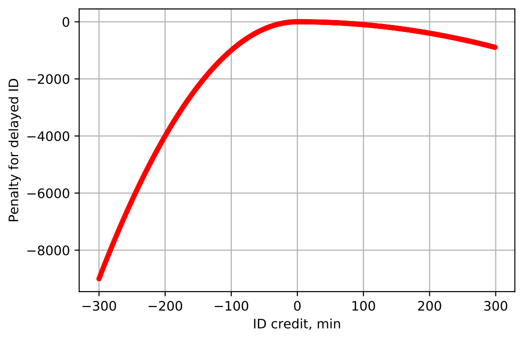

***Fig. 7.*** *Reward as a function of ID state and action - penalty for carrying over workload between RPs (ID credit $> 0$) or being idle (ID credit $< 0$).*

## 4. Value Iteration Results and Discussion

We solve the MDP using value iteration algorithm, shown in Algorithm 1. It is a backpropagation algorithm starting from the goal state. The value function, $V(s)$, evaluates each state so that the agent can estimate how good it is to be in a particular state. Value Iteration uses the Bellman Optimality Equation as an update rule:

$$V^*(s) = \max_a \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma V^*(s')] \tag{6}$$

$V^*(s)$ is the optimal value function for all states. It evaluates all the possible actions, $a$, at each state $s$, and estimates the expected reward for all future iterations, $P^a_{ss'}[R^a_{ss'} + \gamma V^*(s')]$. The maximum sum over the actions is selected and the value assigned to the new V(s). This update rule is repeated, until Algorithm 1 converges. The policy, $\pi$, is the optimal action the agent can select from its current state $s$. The equation to extract the policy is the same as the value function, but the assignment uses the argument, rather than the value. This classical dynamic programming (DP) algorithm has a proof of optimality, which can be found in [24].

**Algorithm 1** Value Iteration

```
1. Initialise $V(s) = 0$, for all $s \in S^+$
2. Repeat until converged:
   (a) For each $s \in S$:
         i. $v \leftarrow V(s)$
        ii. $V(s) \leftarrow \max_a \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma V(s')]$
3. Output: policy $\pi$
4. $\pi(s) = arg\max_a \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma V(s')]$
```

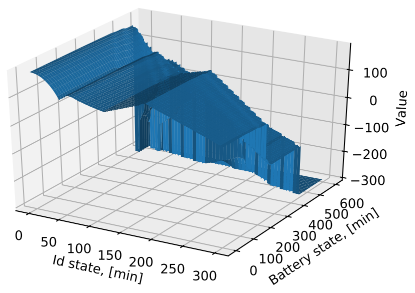

*Fig. 8.* Value function solution.

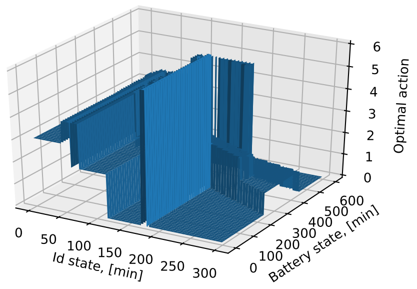

*Fig. 9.* Optimal policy solution of the multi-AUV MCM problem.

## 4.1. Results

Figures 8 and 9 give the value function and optimal policy of the MDP. The $x$ and $y$ axis for both show the state space, $s[0]$ and $s[1]$, respectively. The $z$ axis on Figure 8 is the expected reward, and on Figure 9 is the optimal action the agents can take, given their state.

## 4.2. Results Discussion

In order to see the MDP solution in more detail, Figure 10 gives 2D slices of the 3D policy graph. Each 2D graph corresponds to one of the 7 available actions. For each of them, the $x$ axis is the state representing the remaining battery time and the $y$ axis is the state representing the ID workload. The white colour means an action number was selected for this state space. Table 1 is useful to consult for the following action selection analysis.

In the top left corner is Action 0 graph, where all vehicles do an ID task and the RP is adapted to the time it will require to cover all detections. Following our mission goal, that all detections are revisited by the end of the mission, Action 0 is selected at the end of the mission, where the $x$ axis is approaching 600. The action is also selected earlier in the mission when we have more workload to handle, or when the $y$ axis increases and reaches high values. Another region where Action 0 is represented is in the beginning of the mission ($x = 0 \ to \ 300$), where again the $y$ axis shows large number of detections to ID. This section was not preferred in the first iteration of the algorithm, or the myopic case, but later, when the workload was not handled, the agents deal with detections early on in the mission.

Action 1 is when the RP duration is tied to the ID time, but we have 1 vehicle doing search and 2 ID. The agents gain reward by covering new search area, but are still far from the end of the mission to spend all resource on ID.

Action 3 sends 3 vehicles to search and 0 to ID, RP duration is kept at 60 minutes, the predefined optimal value in the specifications of the mission. This action dominates for the majority of the mission when the ID is low. At the end, an action with ID resource allocation is chosen, such as Action 0, 1 or 2, so that the RP interval is adapted to the remaining battery state.

Action 4 sends 2 vehicles to search and 1 to ID, when the RP duration is kept at 60. It is selected mainly at and around $y = 60$. This follows the reward function penalising deviation from the preference to keep RP=60, so this action is considered optimal by the agents.

Action 6 keeps the RP duration at 60 and all vehicles to ID. This is when the agents put all the

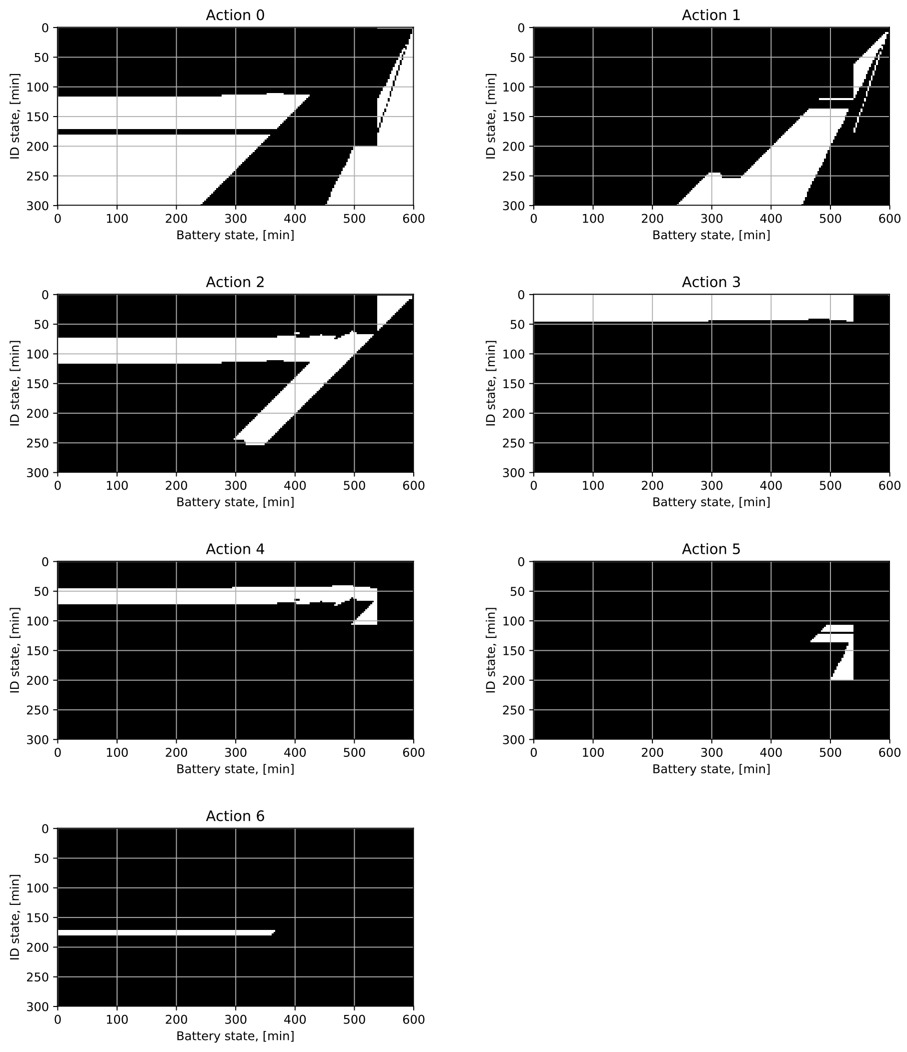

***Fig. 10.*** *2D slices for each action from 3D policy solution. Axis x and y are the battery and ID state spaces.*

system resource into clearing the accumulated ID credit. The action is selected at $y = 180$, which also follows the reduced penalty of not exceeding the RP.

## 4.3. Complexity and Convergence

MDP is considered unrealistic for most real world problem as its state and action space often are continuous or infinite. A lot of research is done in designing approximate techniques to deal with this computation load, as shown in Section 2. Dynamic programming provides an optimal solution at the cost of doing a full sweep through all the states. For our problem, we manage to use the DP algorithm, value iteration, by representing the problem with very small state and action space. The complexity of Value iteration is quadratic in regards of states and linear in regards of actions (per iteration):

$$O(|A||S|^2) \tag{7}$$

Our state space is $s[0] * s[1] = 300 * 600 = 180000$ and the action space is 7. This MDP representation allows a DP solution. The computation time for iteration on an Intel Core i7 Processor takes 4 minutes. However, the computations were done on a single core and the code can be further optimised to reduce the time for a real time application.

In order to evaluate the solution's performance we analysed the convergence criterion. To reach optimal policy, the value iteration algorithm does not need to compute the final value function. Therefore, a small value is selected for a threshold and V(s) of each iteration is compared to the previous one. The maximum value is compared to the threshold, and once this difference is satisfactory, the algorithm terminates. In Figure 11 we have analysed the variation of value function, on axis $x$ and change of policy, axis $y$ for our MDP model. Each subsequent iteration for $V(s)$ and $argmaxV(s)$ is subtracted from the previous one. The dots on the graph represent these differences. The first iteration after the myopic case ($k = 0$) has a difference of the value function calculation of around 250, which corresponds to nearly $35\%$ change in the policy between the two iterations. The graph shows a reduction for the first four iterations and after that, almost no change is observed. We can assume the convergence threshold is reached. The computation time of 4 iterations of the algorithm is around 16 minutes. This is not suitable for real time mission, however optimisation of the code will allow reduction of this time.

## 4.4. Fixed parameters analysis

This subsection discuses some of the fixed parameter choices used in the simulations. It provides the authors' expectation how changing them would affect the results of the simulation.

1. State: The ID parameter can take a value in the interval [0:300). Increasing the number of ID states will increase the state space and hence the complexity. This might be the case when too many targets are detected (for example false alarms). One mitigation technique could be to revisit the area in search mode but change the aspect angle. Introducing this option would require changes to the action model. The Battery parameter has values in the interval [0:600). If a longer endurance needs to be modelled, the state space will increase, as well as the complexity. One solution can be to change the resolution of the calculation from 1 min (current simulation) to 2 to 5 minutes. Altering this parameter might bring reduced computation but the accuracy will decrease.

2. Action: The number of vehicles in the simulation is set to 3 and is directly related to the action

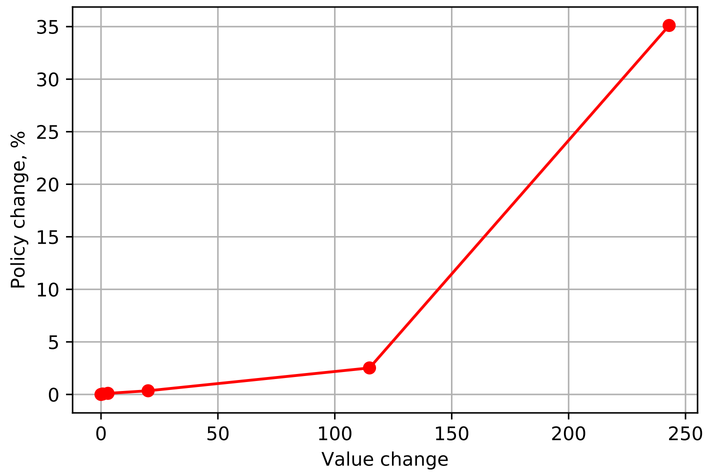

***Fig. 11.*** *Value Iteration termination - convergence condition based on small change in the value function. Axis $x$ is the value function maximum change between iterations and axis $y$ is the percentage of action change between iterations.*

space. The change in vehicle number will require a new action model. Additional implication include things that are ignored in the current simulation but with increased number of vehicles will not be trivial: division of the ID task between vehicles and loss to reach the RP. Another set parameter with direct impact on the action space is the RP interval. It is set to 60 in this simulation. It is a tradeoff between monitoring time and resource loss to travel to the meeting point [9]. The RP loss depends on the mission and vehicle specifications and can be analysed and altered. The RP duration is related to the MDP horizon. Reducing the RP interval will increase the MDP horizon and this will also increase the computation load.

3. Transition Probability: The mine distribution is assumed uniform with a number of 10 per square km. Using a different statistical model for the mine distribution will change the model used to calculate the predicted outcome (s'). If the distribution model is not representative, the prediction error will grow, making the selected action suboptimal. If the quantity of mines is changed, this will have a direct impact on the ID state space. The choice of TSP calculation, using the Beardwood formula for this simulation, also affects the estimation of s'.To improve the model, a better understanding of the upper and lower bounds are important.

4. Reward: The reward function balances searched area, deviation from the desired RP interval and the credit of ID workload left from previous RP intervals. The simulation values are set in Table 2. Changing this balance will change the final goal and the weights of the tradeoffs. They depend on the mission objectives and the user's preference for the agent's goal.

## 5. Conclusion

In this paper, we have developed a decision-making method that balances the tradeoffs of multi-AUV MCM mission and the RP scheduling. As a result, the agents can select an optimal action to maximise the desired output of the mission while complying with hard and soft constraints. In a of mine hunting scenario, the goal we have set is to maximise a search area until the battery of the vehicles is exhausted. One constraint is that the detections found during the search phase need to be revisited. In order to exchange status and contact data, the vehicles meet multiple times throughout the mission, at a Rendezvous Point. The time and place for this RP also needs to be decided autonomously. This resource management tradeoff of covering more area while revisiting contacts and scheduling meeting points is what our MDP is solving. It provides an optimal action policy for the agents, that takes into account estimates for all possible future states, up until the end of the mission.

The main contribution of this work is that our model is capable of computing a solution for the MDP in manageable time, with potential to reach real time execution. We achieved this by discrediting the action and state spaces without losing vital mission information. The other critical element is that utilising the RP method made the decision making intervals very few throughout the mission and we managed to define our MDP with a finite horizon until we reach mission completion.

This work can be expanded by using an actual model for mine field distribution or making the agents learn a model by turning the problem from planning to RL. Another aspect for improvement is using a different solution method, rather than Value Iteration. Novel approximate solutions are available and can significantly reduce the computation time to reach real-time execution. Finally, validation of the model, or parts of it, in a real experiment, will bring insights how suitable the input functions are and what can be updated to make the MDP applicable for a real MCM mission.

## 6. Acknowledgment

This work has been funded by FP7-PEOPLE-2013-INT Robocademy project (ref 608096) by the European Commission, as well as Atlas Elektronik UK and EPSRC Doctoral Training Centre no: EP/G037264/1.

## 7. References

[1] Kalwa, J., Pascoal, A, Ridao, P., *et al*: ’The European R&D-Project MORPH: Marine robotic systems of self-organizing, logically linked physical nodes.’ IFAC Proceedings Volumes 45.5, pp. 349-354, 2012.

[2] Petillo, S., Schmidt, H., Balasuriya A.: ’Constructing a distributed AUV network for underwater plume-tracking operations.’ International Journal of Distributed Sensor Networks, 2011.

[3] Caiti, A., Calabro, V., Munafò, A., Dini, G., Lo Duca, A.: ’Mobile underwater sensor networks for protection and security: field experience at the UAN11 experiment.’ Journal of Field Robotics 30.2, pp. 237-253, 2013

[4] Munafò, A., Simetti, E., Turetta, A., Caiti, A. Casalino, G.: ’Autonomous underwater vehicle teams for adaptive ocean sampling: a data-driven approach.’ Ocean Dynamics 61.11, pp. 1981-1994, 2011.

[5] Hanson, F., Radic, S.: ’High bandwidth underwater optical communication.’ Applied optics 47.2, pp. 277-283, 2008.

[6] Stojanovic, M.: 'Recent advances in high-speed underwater acoustic communications.' IEEE Journal of Oceanic engineering 21.2, pp. 125-136, 1996.

[7] Potter, J., Alves, J., Green, D., Zappa, G., Nissen, I., McCoy, K.: 'The JANUS underwater communications standard.' IEEE Underwater Communications and Networking (UComms), 2014.

[8] Yordanova, V., Griffiths, H.: 'Synchronous rendezvous technique for multi-vehicle mine countermeasure operations.' OCEANS'15 MTS/IEEE Washington, 2015.

[9] Yordanova, V., Griffiths, H.: 'Rendezvous Point Technique for Multivehicle Mine Countermeasure Operations in Communication-Constrained Environments.' Marine Technology Society Journal, 50(2), pp. 5-16, 2016.

[10] Navy, U. S. 'The navy unmanned undersea vehicle (UUV) master plan.' US Navy, 2004.

[11] Silver, D., Huang, A., Maddison, C., *et al*: 'Mastering the game of Go with deep neural networks and tree search.', Nature, 2016, pp. 484-489.

[12] Mnih, V., Kavukcuoglu, K., Silver, D., *et al*: 'Playing atari with deep reinforcement learning.' arXiv preprint arXiv:1312.5602, 2013.

[13] White, D.J.: 'A survey of applications of Markov decision processes.' Journal of the Operational Research Society 44.11 (1993): pp. 1073-1096.

[14] Sutton, R., and Barto, A.: 'Reinforcement learning: An introduction' (Vol. 1. No. 1. Cambridge: MIT press, 1998)

[15] Kearns, M., Mansour Y., Ng, A.: 'A sparse sampling algorithm for near-optimal planning in large Markov decision processes.' Machine learning 49.2 (2002): 193-208.

[16] Panait, L., Luke, S.: 'Cooperative multi-agent learning: The state of the art.' Autonomous agents and multi-agent systems 11.3 (2005): 387-434.

[17] Crites, R., Barto, A.: 'Improving elevator performance using reinforcement learning.' Advances in neural information processing systems 8 (1996).

[18] Dean, T., Kaelbling, L., Kirman, J., Nicholson, A.: 'Planning With Deadlines in Stochastic Domains.' In AAAI, 1993, pp. 574-579.

[19] Barto, A., Bradtke, S., Singh, S.: 'Learning to act using real-time dynamic programming.' Artificial intelligence 72.1-2, 1995, pp. 81-138.

[20] Boutilier, C., Dean, T., Hanks, S.: 'Decision-theoretic planning: Structural assumptions and computational leverage.' Journal of Artificial Intelligence Research 11.1, 1999, p. 94

[21] Shani, G., Pineau, J., Kaplow, R.: 'A survey of point-based POMDP solvers.' Autonomous Agents and Multi-Agent Systems, 2013, pp. 1-51.

[22] Lin, J., Morse, A., Anderson, B.: 'The Multi-Agent Rendezvous Problem. Part 1: The Synchronous Case.' SIAM Journal on Control and Optimization 46.6, 2007, pp. 2096-2119.

[23] Beardwood, J., Halton, J., Hammersley, J.: 'The shortest path through many points', Mathematical Proceedings of the Cambridge Philosophical Society, 1959, **55**, pp. 299-327.

[24] Bellman, R.: 'Dynamic programming and Lagrange multipliers.' Proceedings of the National Academy of Sciences 42.10 (1956): pp. 767-769.