Auton Robot (2015) 38:261–282
DOI 10.1007/s10514-014-9409-9

Model-predictive asset guarding by team of autonomous surface vehicles in environment with civilian boats

Eric Raboin · Petr Švec · Dana S. Nau · Satyandra K. Gupta

Received: 4 May 2013 / Accepted: 10 July 2014 / Published online: 6 August 2014

Abstract In this paper, we present a contract-based, decentralized planning approach for a team of autonomous unmanned surface vehicles (USV) to patrol and guard an asset in an environment with hostile boats and civilian traffic. The USVs in the team have to cooperatively deal with the uncertainty about which boats pose an actual threat and distribute themselves around the asset to optimize their guarding opportunities. The developed planner incorporates a contract-based algorithm for allocating tasks to the USVs through forward simulating the mission and assigning estimated utilities to candidate task allocation plans. The task allocation process uses a form of marginal cost-based contracting that allows decentralized, cooperative task negotiation among neighboring agents. The task allocation plans are realized through a corresponding set of low-level behaviors. In this paper, we demonstrate the planner using two mission scenarios. However, the planner is general enough to be used for a variety of scenarios with mission-specific tasks and behaviors. We provide detailed analysis of simulation results and discuss the impact of communication interruptions, unreliable sensor data, and simulation inaccuracies on the performance of the planner.

Keywords Decentralized planning · Task allocation · Unmanned vehicles · Unmanned surface vehicles

Electronic supplementary material The online version of this article (doi:10.1007/s10514-014-9409-9) contains supplementary material, which is available to authorized users.

E. Raboin
Department of Computer Science, University of Maryland, College Park, MD 20742, USA
e-mail: eraboin@umd.edu

P. Švec
Simulation Based System Design Laboratory, Maryland Robotics Center, Department of Mechanical Engineering, University of Maryland, College Park, MD 20742, USA
e-mail: petrsvec@umd.edu

D. S. Nau
Department of Computer Science and Institute for Systems Research, University of Maryland, College Park, MD 20742, USA
e-mail: nau@umd.edu

S. K. Gupta (✉)
Simulation Based System Design Laboratory, Maryland Robotics Center, Department of Mechanical Engineering and Institute for Systems Research, University of Maryland, College Park, MD 20742, USA
e-mail: skgupta@umd.edu

1 Introduction

Teams of cooperative, highly-maneuverable unmanned surface vehicles (USVs) Corfield and Young (2006) can be utilized for guarding of designated regions or assets (e.g. oil tankers, commercial cargo ships, etc.) in naval missions. The use of autonomous robotic systems brings several advantages which include reducing the risk of human fatalities and significantly decreasing the cost of missions, while preserving the expected level of security. This, however, imposes multiple challenging requirements on the decision-making capability of these vehicles.

The guarding of an asset by a team of USVs requires cooperative patrolling of the surrounding area, approaching and observing passing boats, recognizing the hostile boats, and taking other appropriate means to maximize the effectiveness of the mission (see Fig. 1). Intelligent, balanced decisions about which tasks to perform must be made by the vehicles to prevent adversaries from reaching the asset undetected. This presents a non-trivial challenge for the USVs, since the identity of the hostile boats may not be known at the time they enter the visibility range of the USVs. In addition, the

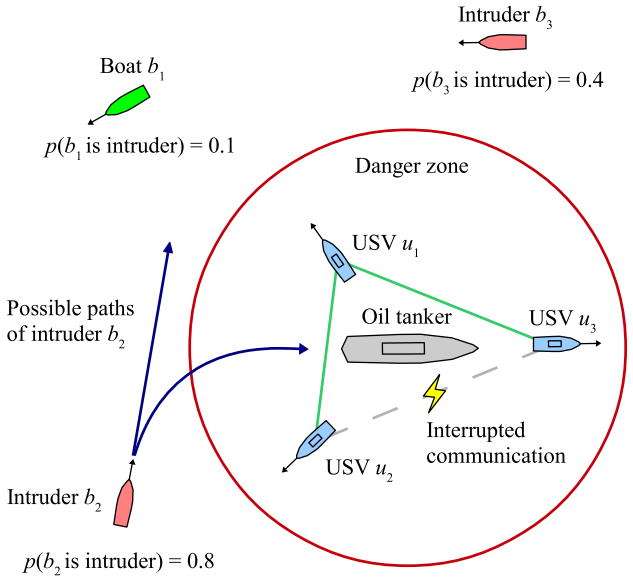


Fig. 1 A team of unmanned surface vehicles (USVs) is guarding an oil tanker against hostile boats in a region with civilian traffic. During the operation, each boat is assigned a probability of being hostile based on observations made by the USVs

possibility of intermittent communication interruptions, noisy sensor data, and the differential constraints of the vehicles all impose additional complications. The vehicles also have to consider the time dependencies of the problem, since selecting current tasks requires knowledge of what future tasks are possible in order to maximize the expected performance of the entire team. Finally, the task allocation must be done efficiently despite the very large state-action space, and the developed approach should be general enough to be usable for a range of scenarios and missions.

This paper presents a decentralized, contract-based planning approach for a team of USVs to patrol and guard a given region containing a valuable asset against hostile boats in an environment with civilian traffic. The work is mostly concerned with high-level task allocation and behavior optimization to allow safe, high-performance, autonomous operation. The developed planner computes an approximate solution to an instance of the MT-MR-TA (i.e., multi-task robots, multi-robot tasks, time-extended allocation) variant of the task allocation problem (Gerkey and Matarić 2004).

The developed planning approach uses a decentralized algorithm where each USV is responsible for managing its own task assignment and negotiating exchanges with the other vehicles. The individual USVs use two-side share and offer contracts to incrementally agree on the set of observe, guard, and delay tasks. These contracts allow the vehicles to establish a specific communication and task allocation protocol. Model-predictive simulations are leveraged during the evaluation of candidate task allocations, i.e., by looking ahead to estimate the utility of the task allocation based on the future states of the boats in the scene.

The tasks are realized by corresponding parameterized behaviors, which are optimized for a specific mission defined by the number of available USVs, estimated number of intruders, spatial distribution of the incoming boats with respect to the target, expected distribution of civilian traffic, etc. These behaviors implement a local, reactive obstacle avoidance that respects the differential constraints of the vehicles. The weighting of each tasks and parametrization of the behaviors are optimized to account for their individual contributions to the overall guarding strategy.

The developed planner is capable of generating a task allocation plan efficiently using an online computation and is scalable to a large number of vehicles. We demonstrate its performance in simulation using a team of autonomous unmanned surface vehicles (USVs) guarding a stationary asset. We compare it to a baseline approach, which does not perform online task re-allocation, and a heuristic strategy that does not use predictive simulation to aid in the task assignment. We demonstrate that the use of model-predictive simulation leads to significantly higher performance and robustness than the pure use of task-tailored heuristic rules. We also show that through careful Monte-Carlo sampling over the distribution of possible worlds, the model-predictive simulation produces better results, even if the number of samples is kept relatively small.

We believe this paper presents a novel application of contract-based task allocation with model-predictive simulation to the domain of planning for unmanned surface vehicles. We provide a detailed analysis of the developed approach and discuss lessons learned when designing the algorithm so that this information may be used by robotic practitioners attempting solve related problems. These contributions include: (1) an analysis of the trade-off between computational effort and plan quality when varying parameters of the algorithm; (2) an analysis of the scalability/computational complexity of the approach for different numbers of USVs or boats; (3) an analysis of the effect of the errors in model-predictive simulation or opponent model on the planner; and (4) an analysis of the impact of sensor noise or communication uncertainty on the planner.

Our results demonstrate that simulating only a few seconds into the future is enough to see performance gains of greater than 50 % when evaluating task exchanges. Additionally, we show that even using a small sample size, Monte-Carlo sampling is beneficial for dealing with sensor uncertainty or uncertainty about the opponent model. Finally, the running time analysis reveals that the algorithm has a fast execution time and low-order polynomial time complexity in relation to the number of USVs, suggesting that the algorithm has the computational efficiency needed for online planning.

The outline of the paper is as follows: in Sect. 2 we review existing approaches to multi-agent guarding, patrolling and task assignment problems. In Sect. 3, we provide a formal

definition of our multi-agent planning problem. In Sect. 4 we provide a detailed description of our approach. In Sect. 5 we describe an experimental setup and discuss our results. We provide a brief summary of future work in Sect. 6.

2 Related work

The outlined problem can be decomposed into multiple components, e.g., accelerated simulation (Thakur and Gupta 2011; Thakur et al. 2012), trajectory planning for collision-free guidance Švec et al. (2011), Švec et al. (2012), Thakur et al. (2012), Švec et al. (2013), Bertaska et al. (2013), Švec et al. (2013), learning of interception behaviors (Švec and Gupta 2012), and multi-agent task allocation and planning.

In this paper, our focus is mostly on task allocation and planning. Hence, we provide an overview of representative approaches in this domain for intentionally cooperative systems of robotic agents Parker (2008). In particular, we focus on distributed and hybrid teams of agents Dias et al. (2006).

Task planning involves the manual or automated decomposition of mission objectives into individual subtasks, which may be arranged in hierarchical trees, or clustered and assigned as roles to individual robots. The subtasks then need to be allocated to the robots based on a number of factors according to the multi-robot task allocation (MRTA) taxonomy in Gerkey and Matarić (2004). The particular factors include the number of tasks that can be performed by a single robot (i.e., ST as single-task robots, and MT as multi-task robots), the number of robots that may be required to fulfil a task (i.e., SR as single-robot tasks, and MR as multi-robot tasks), whether the current assignment of tasks is optimized for future task assignments or not (i.e., IA as instantaneous task allocation and TA as time-extended allocation), and the level of task interdependencies. The variant mostly related to our work is MT-MR-TA, which is known in general to be $\mathcal{NP}$-hard, making efficient computation of the optimal task allocation infeasible.

The core techniques for solving MRTA problems can be categorized into behavior-based and negotiation-based approaches (Parker 2008; Mosteo and Montano 2010) depending on whether the robots solely rely on the states of other robots and their capabilities, or explicitly communicate to decide on the tasks.

The behavior-based approaches do not favor explicit communication among agents to allocate the tasks. Rather, the task allocation is decided based on the known states and skills of other agents in a purely distributed manner. A short survey of the techniques that belong to this category can be found in Parker (2008).

Our task allocation algorithm, an extension of our previous work Raboin et al. (2013), belongs to the category of negotiation-based approaches where agents must individually communicate to decide on the task assignment. This negotiation-based category includes contract and market-based techniques for allocating tasks between agents, as opposed to more centralized approaches Simmons et al. (2000). A survey on the current state-of-the-art market-based techniques for multirobot task allocation is given in Dias et al. (2006), Parker (2008), Shoham and Leyton-Brown (2010).

Most of the currently existing market-based approaches for cooperative task assignment are based on the Contract Net Protocol (CNP) Smith (1980), one of the pioneering negotiation (auction) protocols for implementation of task allocation algorithms in a distributed setting. According to this protocol, the robots explicitly communicate and negotiate tasks using a specific strategy. This leads to a gradual improvement in the assignment of tasks to robots that have the best capabilities to perform them. The market-based task allocation approaches differ on the type of the negotiation protocol. The protocol defines the way in which the agents offer or request tasks given their capabilities, how many of these tasks can be involved in a single contract (e.g., cluster contracts if more than one task is dealt with), how many agents are involved in a single negotiation (e.g., so-called multi-agent contracts if more than two agents are considered), or whether the agents offer tasks in exchange for another task, e.g., in the form of swap contracts in exchange-like auctions (Sandholm 1998; Shoham and Leyton-Brown 2010).

The representative market-based techniques include MURDOCH (Gerkey and Matarić 2002) and TraderBots (Dias 2004), solving the ST-SR-IA and ST-SR-TA variants, respectively.

A distributed, market-based approach MURDOCH for hierarchical task allocation was introduced in Gerkey and Matarić (2002). The approach is based on the CNP protocol and publish/subscribe communication model. It can handle robot failures by reassigning the tasks to the most suitable robots in a greedy fashion, and can consider newly created tasks in the allocation process.

The market-based approach TraderBots (Dias 2004) was developed for distributed coordination of self-interested agents. The approach is known for its capability to create centralized sub-groups within the distributed team to improve the global task allocation efficiency. The approach is able to deal with disruptions in communication through exchange task style of auctioning. Zlot and Stentz present an extension of the TraderBots approach for complex task allocation in Zlot and Stentz (2006). The approach explicitly considers the task structure and its properties to produce more efficient allocations. This includes complex decisions on subtasks sharing among the robots. The subtasks are hierarchically arranged into a task tree, which allows them to be negotiated at different levels.

The Hoplites approach (Kalra et al. 2005) was developed for solving a complex coordination of a distributed group of robots with highly coupled tasks. The approach is market-based is one of the first approaches able to solve the ST-MR-IA type of problems. It provides planned coordination in addition to the tight coordination capability. It combines two different coordination strategies, i.e., the passive, purely local strategy according to which the robots have to quickly decide on the tasks, and the active, market-based strategy which allows them to agree on the tasks in more complex scenarios.

A task allocation approach for computing a combination of strongly and weakly cooperative solutions for a group of heterogeneous robots operating in the context of a single task allocation application was introduced in Tang and Parker (2007). More specifically, the ASyMTRe-D algorithm for the synthesis of coalitions within the group was combined with a market-based approach for the allocation of weakly cooperative tasks. According to the authors, the approach is thus highly flexible and amenable to a large variety of robotic applications.

A game-theoretic approach is introduced in Fang et al. (2013) for solving the "Mobile Resources protecting Moving Targets" (MRMT) problem, where multiple defenders must guard a set of moving targets against multiple attackers. The authors represent the problem as a continuous time Stackelberg game and use linear programming to find a Strong Stackelberg equilibrium. This approach has the advantage of offering a minimum performance guarantee against any possible opponent, something that our work lacks. However, compared to our work, the interaction between defenders and attackers in their model is highly simplistic. They do not consider the movement of the attackers, so the notion of blocking or intercepting an attacker is only dealt with abstractly. The attacker may choose an arbitrary time to attack one of the targets, and the probability of the attack succeeding is determined by whether or not the target is currently within the "protection radius" of one or more defenders. In contrast, we explicitly model the movement of USVs and intruders when blocking or intercepting, and include that information directly in our strategy evaluation. We also address the problem of neutral, non-hostile boats that must be distinguished from intruders through observation, something which their work does not directly address.

Similar to the work above, the work in Bošanský et al. (2011) computes a Stackelberg equilibrium for a defensive game with multiple moving targets and multiple attackers. Unlike the work above, this work does consider the movement of the attackers, but the problem space is very heavily discretized. Their approach depends on solving an $\mathcal{NP}$-hard non-linear program, so the largest problem they evaluated used a 5x4 grid to represent the environment. Because their model is fairly coarse, this work does not address differential constraints or realistic blocking behavior. This approach was utilized in the marine domain to protect merchant ships against pirate boats (Jakob et al. 2012). Additional work on Stackelberg games in this domain includes the incorporation of Quantal Response models for adversary behavior (Shieh et al. 2012) and accounting for the constrained mobility and limited endurance of defender agents (Vanek et al. 2012).

In the USV domain, a decentralized, behavior-based STAGS approach for a multi-USV system to protect sensitive areas against intruders was developed in Zhang and Meng (2010). The deployment of the vehicles is controlled by a heuristic algorithm that uses dynamically created gaps between the vehicles and the asset. The parameters of the approach are optimized to improve its performance by minimizing the average response time and missing rate.

Purely rule-based approaches include Simetti et al. (2010) as a part of the Swarm Management Unit (SMU) used for controlling a team of USVs to carry out surveillance and guarding an asset by intercepting detected intruders. The approach selects which USVs should intercept a detected intruder based on domain-specific heuristics. The positions of the USVs are optimized according to two criteria, i.e., preventing the intruders from getting too close to the asset and minimizing the interception time. The aim is to find a balance between the desired coverage of the area around the asset and the level of security.

Related research also includes techniques for patrolling a polygonal area using a group of agents. A survey of the current state-of-the-art patrolling algorithms is provided in Portugal and Rocha (2011). The representative approaches are evaluated in detail in Portugal and Rocha (2011) in terms of the average idleness of a patrolling graph and scalability to the number of agents metrics. In our approach, the patrolling strategy is computed indirectly through the market-based exchange of guard tasks commanding the vehicles to computed waypoints or predefined patrolling locations.

In contrast to the previously outlined approaches, our work makes explicit consideration of sensing uncertainty when differentiating intruder boats from other non-hostile boats (i.e., by requiring observation of passing boats to identify threats), and accounts for differential constraints of the USVs and their complex interaction with the intruders when allocating tasks (i.e., when executing intercepting and blocking strategies). We show that by carefully integrating the model-predictive simulation with the underlying task allocation, features such as differential constraints and sensing uncertainty can be directly considered during task allocation and still run efficiently. We provide a detailed analysis of the developed approach and describe lessons learned for realizing high-fidelity task allocation with unmanned surface vehicles.

3 Problem formulation

We define a multi-agent planning problem where a team of USVs must defend a stationary target from an attack by a set of hostile intruder boats interspersed among other non-hostile boats. The USV team does not know *a priori* which boats are hostile intruders, but can estimate the probability that a boat is an intruder through observation. Once an intruder is identified, an alert is triggered, and the objective of the USV team becomes to delay the hostile boats from reaching the target for as long as possible. This is done by actively blocking the path of the intruders, forcing the intruders to slow down or change direction.

In addition to uncertainty about which boats are intruders, the USV team must deal with noisy sensor data, which creates uncertainty over the position of passing boats, and random communication interruptions, which create periods of time where USVs cannot exchange information directly.

The formal definition of the problem includes:

(i.) a team of USVs $U = \{u_1, u_2, \ldots u_m\}$ responsible for defending the target from intruder boats,
(ii.) a set of passing boats $B = \{b_1, b_2, \ldots b_n\}$ including a subset of one or more hostile intruders $I \subseteq B$,
(iii.) the location l_{target} of the stationary target,
(iv.) a vehicle state space $X \subseteq \mathbb{R}^3 \times \mathbb{S}$, where each state $\mathbf{x} = \{x, y, \psi, v\}$ defines the coordinates x, y, heading angle ψ, and surge speed v of a single boat or USV,
(v.) a global state space $S \subseteq \mathbb{R} \times X^{(m+n)}$, where each global state s defines the time $t \in \mathbb{R}$, and the vehicle state $\mathbf{x}_{u_i} \in X$ and $\mathbf{x}_{b_j} \in X$ for every USV u_i and boat b_j in the scene,
(vi.) a control action space $Q(\mathbf{x}) \in \mathbb{R}^2$ for each $\mathbf{x} \in X$, where each control action $q = \{\Delta v, \Delta\psi\}$ defines a change in surge speed Δv and heading angle $\Delta\psi$,
(vii.) a pair of probabilistic opponent models, $P_{B,b_i}(q|s)$ and $P_{I,b_i}(q|s)$, for passing boats and intruders respectively, which define the probability of boat b_i performing control action q given global state s,
(viii.) a non-finite set of observations O, where each observation $o_{b_j} = \{\widetilde{\mathbf{x}}_{b_j}, f_{b_j}\}$ provides a noisy estimate $\widetilde{\mathbf{x}}_{b_j}$ of the vehicle state $\mathbf{x}_{b_j}$ and observed features f_{b_j} of the boat b_j (e.g., color, size, etc.),
(ix.) an observation function $\Omega_i(o_{b_j}|s)$ which returns the probability that the USV u_i will perform observation o_{b_j} in global state s, adding o_{b_j} to USV u_i's observation history $\mathcal{O}_{u_i} \in O$,
(x.) a communication reliability function $C_{i,j}(s)$ which returns the probability that the USV u_i can communicate with the USV u_j in global state s,
(xi.) an intruder classification function $P(b_i \in I|\mathcal{O}_{u_j})$ which returns the probability that boat b_i is an intruder given observation history $\mathcal{O}_{u_j}$,
(xii.) a response team probability threshold p_{alert}, defining the probability $P(b_i \in I|\mathcal{O}_{u_j})$ above which an alert will be triggered.

The objective of the USV team is to find a set of policies $\Pi_U = \{\pi_{u_1}, \pi_{u_2}, \ldots, \pi_{u_m}\}$ that maximize the expected delay time, $E[t_{delay}]$, defined as the time difference between when the alert is first triggered and when an intruder first arrives at the target, or $t_{delay} = t_{arrival} - t_{alert}$. Thus, the optimal set of policies Π_U^* is defined as,

$$\Pi_U^* = \arg\max_{\Pi_U} E[t_{delay}|\Pi_U]. \tag{1}$$

Exactly computing Π_U^* is likely to be intractable, so we are interested only in finding a set of policies that can be computed efficiently, even if they are sub-optimal.

The motivation behind maximizing $E[t_{delay}]$ is to provide a hypothetical *response team* as much time as possible to deal with the intruders. Although the response team is not modeled explicitly as part of the problem, we assert that the longer intruders are delayed from reaching the target, the more time the response team will have to repel an attack, make emergency preparations, or perform other task which are beneficial the overall mission. Additionally, by having the alert triggered by the response team, we isolate the USVs from the responsibility of classifying intruders or weighing the cost of false alarms.

For the motion of the boats and USVs, we use a simple steering model where control action $q = \{\Delta v, \Delta\psi\}$ determines the change in surge speed v and orientation ψ, while $\Delta x = v\cos(\psi)$ and $\Delta y = v\sin(\psi)$ determine the change in coordinates (x, y). To estimate the movement of the USV after a time step Δt, we use the formula

$$\mathbf{x}' = \mathbf{x} + \Delta\mathbf{x}(q)\Delta t \tag{2}$$

where $\Delta\mathbf{x}(q) = \{\Delta x, \Delta y, \Delta\psi, \Delta v\}$ represents the change in vehicle state $\mathbf{x}$ given control action q.

The set of control actions $Q(\mathbf{x}) = A(\mathbf{x}) \times \Theta(\mathbf{x})$ is subject to physical constraints, where

$$A(\mathbf{x}) = \{\Delta v : a_{min}(\mathbf{x}) \leq \Delta v \leq a_{max}(\mathbf{x})\} \tag{3}$$

$$\Theta(\mathbf{x}) = \{\Delta\psi : \theta_{min}(\mathbf{x}) \leq \Delta\psi \leq \theta_{max}(\mathbf{x})\} \tag{4}$$

define the minimum and maximum change in surge speed Δv and turning angle $\Delta\psi$ given vehicle state $\mathbf{x}$. We assume that the maximum turning radius decreases as the surge speed v increases. Each boat has a maximum surge speed, v_{max}, which it cannot accelerate past. We also assume that boats cannot travel in reverse, so $a_{min}(\mathbf{x})$ is zero when the surge speed v is zero.

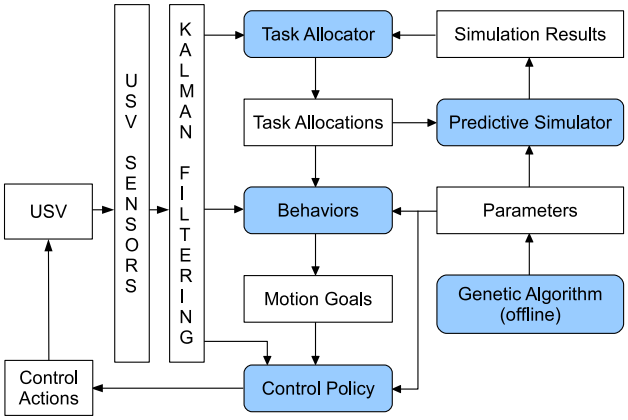


Fig. 2 Planning and control architecture for one USV

To accurately reflect the limitation of USVs' knowledge, policy π_{u_i} must depend only on information that is accessible to the USV u_i at the time the policy is executed. The information known to USV u_i at time t, includes:

(a.) the state $\mathbf{x}_{u_i}$ of USV u_i up to time t
(b.) USV u_i's observation history $\mathcal{O}_{u_i}$ up to time t
(c.) information received from other USVs before time t

When in communication, USVs are permitted to exchange arbitrary information, including their current state and observation histories. USVs are also assumed to have access to opponent models $P_{B,b_j}(q|s)$ and $P_{I,b_j}(q|s)$, classifier $P(b_j \in I|\mathcal{O}_{u_i})$, target location l_{target} and response team threshold p_{alert} when computing their policies.

The parameterization of opponent models $P_{B,b_i}(q|s)$ and $P_{I,b_i}(q|s)$, observation function $\Omega_i(o_{b_j}|s)$, communication reliability function $C_{i,j}(s)$ and intruder classification function $P(b_i \in I|\mathcal{O}_{u_j})$ are described in the experimental setup in Sect. 5. The noisy state estimates $\widetilde{\mathbf{x}}_{b_j}$ produced via observation are assumed to be the result of adding Gaussian noise to the individual components of the state vector $\mathbf{x}$, whose variances are also provided in Sect. 5.5 to accompany the relevant set of experiments.

4 Approach

We present a solution technique for the problem described in Sect. 3. Our approach, illustrated in Fig. 2, consists of three major components: (1) a decentralized task allocation process that determines the high-level task assignment of each USV; (2) a set of parameterized behaviors that map each USV's task assignment into a unique motion goal; and (3) a control action policy that selects a control action for each USV to reach its motion goal while performing local obstacle avoidance. Each of these processes operate concurrently and are performed online, updating as new sensor information becomes available.

Our approach utilizes a set of domain-specific tasks that are specified by the system developer. For the purposes of the problem defined in this paper, we define the set of high-level tasks, $H \subseteq H_o \cup H_g \cup H_d$, as the union of three distinct task types:

(i.) a set of *observe* tasks, H_o, where USVs are responsible for gathering information about passing boats
(ii.) a set of *guard* tasks, H_g, where USVs must position themselves in vulnerable areas around the target
(iii.) a set of *delay* tasks, H_d, where USVs must intercept and then block a hostile boat

Each task $h_j \in H_o \cup H_d$ specifies a single boat to observe or delay, while each $h_j \in H_g$ specifies a single location to guard. The task assignment $H_{u_i} \subseteq H$ for USV u_i may contain any combination of these tasks.

The joint task allocation $\mathcal{A} = \{H_{u_1}, H_{u_2}, \ldots H_{u_m}\}$ defines the current task assignment for all USVs and is computed online and updated during each planning step via a decentralized task allocation process. Our algorithm uses both heuristic and model-predictive simulation to determine how the task allocation should be updated. This process is described in Sect. 4.3.

Since communication between USVs can be interrupted, each USV u_i maintains a local task allocation $\mathcal{A}_{u_i}$ representing u_i's belief about the joint task allocation $\mathcal{A}$ based on the most recent information made available to u_i. The local task allocation $\mathcal{A}_{u_i}$, together with the observation history $\mathcal{O}_{u_i}$, form the input to USV u_i's motion goal selection $M_{u_i}(\mathcal{O}_{u_i}, \mathcal{A}_{u_i})$ and control action selection $\pi_{u_i}(\mathcal{O}_{u_i}, \mathcal{A}_{u_i})$ policies. These are defined in Sects. 4.1 and 4.2.

Both motion goal selection and control action selection policies are based on hand-coded heuristics which are tuned offline using a genetic algorithm. The set of tuning parameters $\Gamma = \{\gamma_{guard}, \gamma_{intr}, \gamma_{dist}, \gamma_{lead}, \gamma_{block}, \gamma_{max}, \gamma_{initial_r}, \gamma_{occupied}, \gamma_{slow_r}, \gamma_{goal_r}, \gamma_{fan_r}, \gamma_{fan_\theta}\}$ determine the low-level behavior of the USVs when selecting motion goals or performing obstacle avoidance. The tuning process used for these parameters is described in Sect. 4.7.

The sequence of state estimates $\{\widetilde{\mathbf{x}}_{b_j,1}, \widetilde{\mathbf{x}}_{b_j,2}, \ldots \widetilde{\mathbf{x}}_{b_j,n}\}$ for boat b_j contained in USV u_i's observation history $\mathcal{O}_{u_i}$ are processed using a simple Kalman filter, where $l_{b_j}(\mathcal{O}_{u_i})$ represents the estimate of boat b_j's location given $\mathcal{O}_{u_i}$ and $K_{b_j}(\mathcal{O}_{u_i})$ represents the corresponding covariance matrix. We will refer to these as simply l_{b_j} and K_{b_j} throughout the paper.

4.1 Behaviors

The behaviors for USV u_i are a set of hand-coded heuristics that map u_i's task assignment H_{u_i} to a unique motion goal $M_{u_i}(\mathcal{O}_{u_i}, \mathcal{A}_{u_i})$. The rules for computing this motion goal are defined below.

Each task $h_j \in H_{u_i}$ assigned to USV u_i is given a task-specific motion goal, $M_{h_j}(\mathcal{O}_{u_i}, \mathcal{A}_{u_i})$, defined as

$$M_{h_j}(\mathcal{O}_{u_i}, \mathcal{A}_{u_i}) = \begin{cases} \text{boat location, } l_{b_j}, & \text{if } h_j \in H_o, \\ \text{guard location, } l_{g_j}, & \text{if } h_j \in H_g, \\ \text{intercept point, } l_{int}, & \text{if } h_j \in H_d. \end{cases} \quad (5)$$

For observe or guard tasks, this motion goal corresponds to the estimated boat location l_{b_j} or pre-defined guard location l_{g_j} associated with task h_j. For delay tasks, the motion goal is an intercept point l_{int} positioned along a line between some intruder b_j and the target.

Since USV u_i can be assigned multiple tasks, u_i's desired motion goal may differ from the motion goal of any individual task. The desired motion goal for u_i is

$$M_{u_i}(\mathcal{O}_{u_i}, \mathcal{A}_{u_i}) = \begin{cases} M_{h_j}(\mathcal{O}_{u_i}, \mathcal{A}_{u_i}), & \text{if } \exists h_j \in H_{u_i} \cap H_d, \\ M_w(\mathcal{O}_{u_i}, \mathcal{A}_{u_i}), & \text{otherwise,} \end{cases} \quad (6)$$

which returns $M_{h_j}(\mathcal{O}_{u_i}, \mathcal{A}_{u_i}) = l_{int}$ if H_{u_i} contains some delay task $h_j \in H_d$. Otherwise, it returns a weighted motion goal (see Fig. 3) based on the USV u_i's currently assigned guard and observe tasks,

$$M_w(\mathcal{O}_{u_i}, \mathcal{A}_{u_i}) = \frac{\sum_{h_j \in H_{u_i}} w_{h_j}(\mathcal{O}_{u_i}) M_{h_j}(\mathcal{O}_{u_i}, \mathcal{A}_{u_i})}{\sum_{h_j \in H_{u_i}} w_{h_j}(\mathcal{O}_{u_i})}, \quad (7)$$

where $w_{h_j}(\mathcal{O}_{u_i})$ is the weight of task h_j, equal to γ_{guard} if h_j is a guard task, and equal to $w_{b_j}(\mathcal{O}_{u_i})$ if h_j is an observe task for boat b_j,

$$w_{b_j}(\mathcal{O}_{u_i}) = \gamma_{intr} P(b_j \in I | \mathcal{O}_{u_i}) \left(1 + \frac{\gamma_{dist}}{|l_{target} - l_{b_j}|}\right). \quad (8)$$

The parameters γ_{guard}, γ_{intr}, and γ_{dist} are tuned by the genetic algorithm using the method described in Sect. 4.7.

We simplify the calculation of intercept point l_{int} by assuming both USVs and intruders can travel in any direction at maximum velocity, ignoring differential constraints and acceleration. If the intruder follows a linear path directly to the target, then l_{int} is the closest intercept point for the USV along that path.

More formally, to find l_{int} for a single USV u_i assigned to a single intruder b_j, the intercept point calculation finds the nearest point in the set of possible intercepts $L_{int}(u_i, b_j)$, defined as

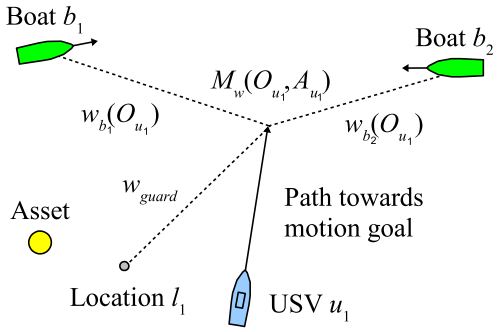


Fig. 3 USV u_1 approaches the weighted motion goal $M_w(\mathcal{O}_{u_i}, \mathcal{A}_{u_i})$ corresponding to two observe tasks for boats b_1 and b_2 and a guard task for location l_1

$$L_{int}(u_i, b_j) = \{l : L_{path}(u_i, b_j) \cap L_{target}(b_j)\} \quad (9)$$

where $L_{path}(u_i, b_j)$ is the set of points for which some linear path for u_i intercepts some linear path for b_j at their respective maximum velocities,

$$L_{path}(u_i, b_j) = \left\{ l : \frac{|l - l_{u_j}|}{v_{max,u_i}} = \frac{|l - l_{b_j}|}{v_{max,b_j}} \right\}, \quad (10)$$

and L_{target} is the set of points l that lie on the path between b_j and the target,

$$L_{target}(b_j) = \{l : \exists_{s \geq 0}[l_{b_j} + s(l - l_{b_j}) = l_{target}]\}. \quad (11)$$

We define $L_{int}(H_{u_i})$ as the union of $L_{int}(u_i, b_j)$ for all boats b_j with a corresponding delay task $h_j \in H_{u_i} \cap H_d$. The point $l_{int}(H_{u_i})$ is the intercept point $l \in L_{int}(H_{u_i})$ that minimizes its distance to u_i's current location.

$$l_{int}(H_{u_i}) = \begin{cases} \arg\min_{l \in L_{int}} |l - l_{u_i}|, & \text{if } L_{int}(H_{u_i}) \neq \emptyset, \\ l_{target}, & \text{otherwise.} \end{cases} \quad (12)$$

If $L_{int}(H_{u_i})$ is empty, no intercept is reachable, so USV u_i will head to the location of the target instead.

When multiple USVs are assigned to delay b_j, the calculation of $L_{int}(H_{u_i})$ incorporates a speed reduction for each additional USV that intercepts the intruder. An example of this calculation for multiple USVs is shown in Fig. 4. The calculation estimates the speed of the intruder as $v_{b_j}(k) = v_{max,b_j} * (\gamma_{block})^k$ after it has been intercepted by k USVs. The calculation adjusts the intercept points for all subsequent USVs accordingly. Since $v_{b_j}(k)$ underestimates the real travel time of the boats and USVs, we define a lead-time parameter, γ_{lead}, which the calculation adds to USV u_i's

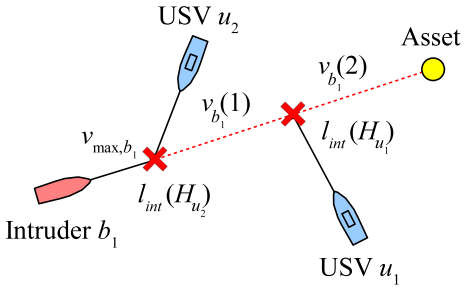


Fig. 4 Heuristic model of USVs u_1 and u_2 intercepting intruder b_1 in a simplified version of the problem. The intercept points serve as a motion goal for the delay behavior of the USVs

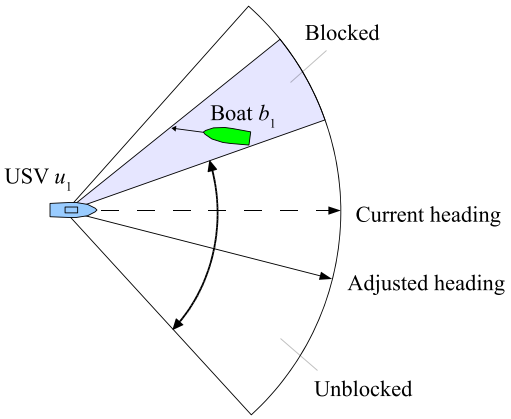


Fig. 5 USV u_1 adjusts its heading to steer away from the region blocked by boat b_1 based on the depicted obstacle avoidance fan

starting time when computing the intercept. The parameters γ_{lead} and γ_{block} are tuned by the genetic algorithm using the method described in Sect. 4.7.

4.2 Control action selection

Given USV u_i's motion goal $M_{u_i}(\mathcal{O}_{u_i}, \mathcal{A}_{u_i})$ and vehicle state $\mathbf{x}_{u_i}$, an appropriate control action $q \in Q(\mathbf{x}_{u_i})$ must be selected to direct u_i towards its goal while avoiding collisions with other boats or static obstacles. Let ψ_{u_i} be u_i's current heading angle and let $\phi_{M_{u_i}}$ be the desired heading angle in the direction of M_{u_i}. The steering angle to achieve this new heading is determined by,

$$\Delta\psi_{M_{u_i}}(\mathbf{x}_{u_i}) = \arg\min_{\Delta\psi\in\Theta(\mathbf{x}_{u_i})} |d(\phi_{M_{u_i}}, \psi_{u_i} + \Delta\psi)|. \quad (13)$$

where $d(\phi_j, \phi_k)$ is the difference between any two angles ϕ_j and ϕ_k. Similarly, the change in surge speed is determined by,

$$\Delta v_{M_{u_i}}(\mathbf{x}_{u_i}) = \arg\min_{\Delta v\in A(\mathbf{x}_{u_i})} |\eta_{M_{u_i}} - (v_{u_i} + \Delta v)| \quad (14)$$

where $\eta_{M_{u_i}}$ is the desired surge speed of USV u_i as it approaches M_{u_i}. The resulting control action is simply,

$$q_{M_{u_i}}(\mathbf{x}_{u_i}) = \{\Delta v_{M_{u_i}}(\mathbf{x}_{u_i}), \Delta\psi_{M_{u_i}}(\mathbf{x}_{u_i})\}. \quad (15)$$

However, this control action may lead to a collision with obstacles such as other boats or rocks. To reduce the chance of collision, the desired heading $\phi_{M_{u_i}}$ and velocity $v_{M_{u_i}}$ are adjusted using reactive obstacle avoidance.

As depicted in Fig. 5, each USV has an obstacle avoidance fan with radius γ_{fan_r} and angular span γ_{fan_θ} to identify which obstacles pose a risk of collision. Headings within the obstacle avoidance fan are considered blocked if they are occupied by an obstacle or will become occupied by an obstacle within some time t_{lead} based on the obstacles' current velocities. Obstacles are assumed to have a non-zero radius.

Let $Z = \{z_1, z_2, \ldots, z_n\}$ be a set of unblocked sectors inside the obstacle avoidance fan, where each $z_j = [\phi_{j,a}, \phi_{j,b}]$ is a range of angles that are not blocked and where the ordering constraint $\phi_{j,b} \leq \phi_{j+1,a}$ holds for all $j < n$. Let $\phi_{j,mid}$ be a midpoint between $\phi_{j,a}$ and $\phi_{j,b}$. Heading ϕ is considered safe if it is within the obstacle avoidance fan, and $\phi \in [\phi_{1,a}, \phi_{1,mid}]$ or $\phi \in [\phi_{n,mid}, \phi_{n,b}]$, which is trivially true if $n = 1$.

If $\phi^*_{M_{u_i}}$ is the most direct heading to the motion goal, the adjusted heading after reactive obstacle avoidance is,

$$\phi_{M_{u_i}} = \begin{cases} \phi^*_{M_{u_i}}, & \text{if } \phi^*_{M_{u_i}} \text{ is safe,} \\ \phi_{k,mid} \text{ s.t. } z_k = z_{max}, & \text{otherwise,} \end{cases} \quad (16)$$

where z_{max} is the widest unblocked sector,

$$z_{max} = \arg\max_{z_j\in Z} |d(\phi_{j,b}, \phi_{j,a})| \quad (17)$$

The surge speed of the USV is not affected by obstacle avoidance unless $Z = \emptyset$, at which point the USV will slow to a stop. Thus, the desired surge speed is,

$$\eta_{M_{u_i}} = \begin{cases} 0, & \text{if } Z = \emptyset, \\ v_{max,u_i} \frac{|M_{u_i} - l_{u_i}|}{\gamma_{slow_r}}, & \text{if } |M_{u_i} - l_{u_i}| < \gamma_{slow_r}, \\ v_{max,u_i}, & \text{otherwise,} \end{cases} \quad (18)$$

where l_{u_i} is u_i's current location, and γ_{slow_r} determines at what distance the USV should start to slow down.

We define non-zero acceptance radius γ_{goal_r} such that USV u_i is considered at its destination if it is within the distance γ_{goal_r} of its motion goal M_{u_i}. The resulting policy for USV u_i is simply,

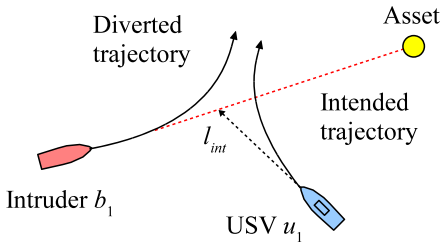


Fig. 6 USV u_1 intercepts intruder b_1, diverting it from its intended path to the asset

$$\pi_{u_i}(\mathcal{O}_{u_i}, \mathcal{A}_{u_i}) = \begin{cases} q_{M_{u_i}}(\mathbf{x}_{u_i}), & \text{if } |M_{u_i} - l_{u_i}| > \gamma_{goal_r}, \\ q_{stop}(\mathbf{x}_{u_i}), & \text{otherwise,} \end{cases} \tag{19}$$

where $q_{stop}(\mathbf{x}_{u_i}) = \{a_{min}(\mathbf{x}_{u_i}), 0\}$ is a control action that quickly halts the movement of the USV.

The control action selection for passing boats is identical to USVs, only a different motion goal M_{b_j} and different set of parameters γ_{fan_r}, γ_{fan_θ} and γ_{slow_r} are selected. For our experiments detailed in Sect. 5, the parameters for the passing boats including intruders are predefined, while the parameters for the USVs are learned using a genetic algorithm, described in Sect. 4.7.

If one or more USVs move within the obstacle avoidance fan of an another boat, the boat will be forced to adjust its trajectory to avoid a collision. This is illustrated in Fig. 6, where a USV intercepts an intruder, diverting its trajectory away from the target.

4.3 Task allocation

The joint task allocation $\mathcal{A}$ is periodically updated via a decentralized task re-allocation process that is performed concurrently by each of the USVs. This process behaves like a local hill-climbing algorithm, where each USV u_i evaluates variations of the task allocation $\mathcal{A}_{u_i}$ that differ by exchanging one or two tasks. This process is less computationally demanding than evaluating all possible task allocations. It also makes it easier to decentralize the re-allocation process, since each exchange alters the task assignment of at most two USVs. However, a disadvantage of this approach is that it will only find locally optimal solutions, a limitation common to many hill-climbing algorithms.

Before the task re-allocation process can be performed, an initial task allocation $\mathcal{A}_{u_i,0}$ must be assigned to the USV team. We assume that the initial allocation will be assigned by the system developer. In our case, we assign each USV one or more guard tasks $h_j \in H_g$, distributed uniformly at radius $\gamma_{initial_r}$ around the target. For each new boat b_j that enter the scene, the nearest USV assigns itself an observation task $h_j \in H_o$. If any boats are identified as intruders, meaning

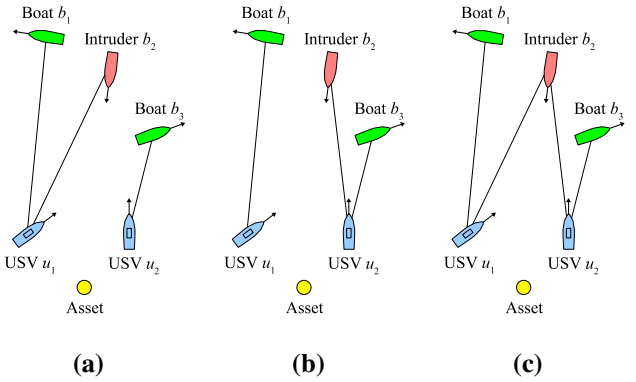


Fig. 7 Candidate task allocations **a** the current task allocation $\mathcal{A}$ without modification, **b** modification of $\mathcal{A}$ with a delay task offered to USV u_2 by USV u_1, **c** modification of $\mathcal{A}$ with a delay task shared to USV u_2

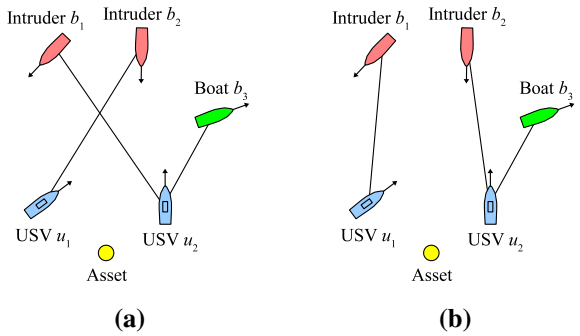


Fig. 8 A conditional "swap" exchange, composed of two distinct offer exchanges **a** the current task allocation $\mathcal{A}$ where USV u_2 by USV u_1 are assigned delay tasks for separate intruders, **b** modification of $\mathcal{A}$ where USV u_2 by USV u_1 have swapped delay tasks

$P(b_i \in I|\mathcal{O}_{u_j}) > p_{alert}$, the observation task for that boat becomes an equivalent delay task, $h_j \in H_d$.

At regular time interval t_{alloc}, each USV u_i performs a task re-allocation step, in which u_i computes a revised allocation $\mathcal{A}'_{u_i}$, defined as

$$\mathcal{A}'_{u_i} = \arg \max_{\mathcal{A}_{u_i,j} \in C} \widetilde{E}[t_{delay}|\mathcal{O}_{u_i}, \mathcal{A}_{u_i,j}] \tag{20}$$

where C is a set of candidate task allocations produced by Algorithm 1, and $\widetilde{E}[t_{delay}|\mathcal{O}_{u_i}, \mathcal{A}_{u_i,j}]$ is the estimated utility of candidate $\mathcal{A}_{u_i,j}$ given observation history $\mathcal{O}_{u_i}$ as computed by the model-predictive simulation described in Sect. 4.5. Each candidate $\mathcal{A}_{u_i,j} \in C$ differs from $\mathcal{A}_{u_i}$ by sharing or offering one or more tasks from H_{u_i} to another USV, as depicted in Figs. 7 and 8.

The method GENERATECANDIDATES($\mathcal{O}_{u_i}, \mathcal{A}_{u_i}, u_i$) in Algorithm 1 produces a set of candidate task allocations C by iterating through every task $h_j \in H_{u_i}$ in USV u_i's task assignment and setting up potential exchanges with every USV $u_k \in U$ on the team.

Algorithm 1 GENERATECANDIDATES($\mathcal{O}_{u_i}, \mathcal{A}_{u_i}, u_i$): Generate a set of candidate task allocations.

```
C ← {A_{u_i}}
for each h_j ∈ H_{u_i} ∩ H_d do
    for each u_k ∈ U do
        C ← C ∪ SHARETASK(A_{u_i}, h_j, u_k)
for each h_j ∈ H_{u_i} ∩ (H_g ∪ H_o) do
    for each u_k ∈ U do
        C ← C ∪ OFFERTASK(A_{u_i}, h_j, u_i, u_k)
return C
```

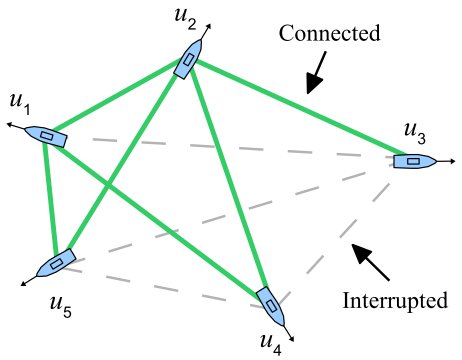


Fig. 9 An example of a communication network

The method SHARETASK($\mathcal{A}_{u_i}, h_j, u_k$) returns a new task allocation $\mathcal{A}'_{u_i}$ that differs from $\mathcal{A}_{u_i}$ by appending task h_j to USV u_k's task assignment H_{u_k}. The result is a task allocation where both u_i and u_k share task h_j.

The method OFFERTASK($\mathcal{A}_{u_i}, h_j, u_i, u_k$) is very similar to SHARETASK, only it removes task h_j from USV u_i's task assignment H_{u_i} before adding it to USV u_k's task assignment H_{u_k}. The method can also be modified to include conditional "swap" exchanges, composed of two distinct offer exchanges, depicted in Fig. 8. This is applicable to situations where USV u_i is already assigned a delay task h_1 and then USV u_j offers u_i an additional delay task h_2. USV u_i cannot intercept and block two different boats at the same time, so u_i will "swap" task h_1 for task h_2. This is equivalent to u_j offering to h_2 to u_i and u_i offering to h_1 to u_j, but both exchanges are represented by a single candidate task allocation, allowing the combined exchanges to be evaluated by the predictive simulation.

4.4 Communication protocol

Communication between USVs is modeled as a network of pairwise connections, as shown in Fig. 9. At each time step t_k, each USV attempts to synchronize its information with the other USVs. If USV u_j is able to communicate with USV u_i, then u_j's most recent observation history $\mathcal{O}_{u_j}$ and task assignment H_{u_j} of USV u_j are merged into USV u_i's observation history $\mathcal{O}_{u_i}$ and global task allocation $\mathcal{A}_{u_i}$. If USV u_i cannot directly communicate with USV u_j, it may still learn of u_j's observations and task assignment through a third USV that can communicate with both agents, but it will take an additional time step for this information to propagate.

For USV u_i to either offer or share a task with another USV u_j, the two USVs must explicitly agree to the exchange. If no communication is possible between u_i and u_j, then exchanges between these agents cannot be performed until communication is re-established. Since communication interruptions occur at random, the USV evaluates all candidate exchanges even if communication is not possible, then determines whether communication has been restored once the evaluation is finished. If the best-performing candidate allocation cannot be applied due to interrupted communication, then the USV selects the next best candidate.

To avoid conflicts that might result from concurrency, the protocol requires that USV u_i retain the exclusive right to assign tasks from H_{u_i} to another USV. For offer and share exchanges, this rule is trivial to enforce, since USV u_i is the agent that initiates the exchange. However, for "swap" exchanges, USV u_i must ask another USV u_j to offer some task h_2 in exchange for the task h_1. Since exchanges are evaluated concurrently, task h_2 may no longer be in u_j's task assignment at the time the request is made. In this situation, USV u_i must forego the swap and perform a different exchange from its list of candidates.

4.5 Predictive simulation

During task re-allocation process, USV u_i uses model-predictive simulation to evaluate the set of candidate task allocations C generated by the method in Algorithm 1. The predictive simulation estimates the expected delay time $E[t_{delay}|\mathcal{O}_{u_i}, \mathcal{A}_i]$ for each candidate task allocation $\mathcal{A}_i \in C$, given USV u_i's observation history $\mathcal{O}_{u_i}$. The simulation uses the probabilistic opponent models defined in Sect. 3 to estimate the future control actions of passing boats and intruders, and uses the control action selection policies for the USVs defined in Sect. 4.2 to estimate the control actions of other USVs.

USV u_i estimates $\mathcal{A}_i$'s performance for the set of possible worlds W consistent with USV u_i's observation history $\mathcal{O}_{u_i}$. Since USV u_i is uncertain which boats are intruders and about the location of each boat, each possible world $w_j \in W$ consists of a set of possible intruder boats $I_j \subseteq B$ and an approximate of the global state $s_j \in S$. The set W is non-finite, so the algorithm uses Monte-Carlo sampling to select n_{sample} possible worlds to estimate the expected utility, $E[t_{delay}|\mathcal{O}_{u_i}, \mathcal{A}_i]$, for each candidate task allocation $\mathcal{A}_i \in C$.

For simplicity, we assume the probability distributions over I and s are statistically independent, meaning

$$P(w_j|\mathcal{O}_{u_i}) = P(I_j = I|\mathcal{O}_{u_i})P(s_j = s|\mathcal{O}_{u_i}).$$

where I is the true set of intruders and s is the true global state. Each global state s_j is sampled using the state estimate l_{b_j} and covariance matrix K_{b_j} produced by the Kalman filter described in Sect. 4. The algorithm samples the set of intruders I_i with probability,

$$P_{I_i} = \left(\prod_{b_j \in I_i} P(b_j \in I | \mathcal{O}_{u_i}) \right) \left(\prod_{b_j \in B \setminus I_i} 1 - P(b_j \in I | \mathcal{O}_{u_i}) \right)$$

where P_{I_i} is an approximation of $P(I_i = I | \mathcal{O}_{u_i})$ computed by assuming that the appearance of each intruder is statistically independent. If $n_{sample} = 1$, the possible world w_j with the highest probability $P(w_j | \mathcal{O}_{u_i})$ is sampled, rather than a random sample.

The task re-allocation process is not simulated during the predictive simulation. This is to prevent the predictive simulation from recursively calling itself, which would lead to an exponential increase in the computational workload as the simulation searches deeper. Each trial is also given a maximum lookahead time, $t_{lookahead}$, after which the utility is estimated using the intercept point heuristic described in Sect. 4.6.

4.6 Heuristic evaluation function

After the maximum lookahead time $t_{lookahead}$ has expired, the predictive simulation uses a heuristic to estimate the value of task allocation $\mathcal{A}_{u_i}$ given global state s_j. The heuristic uses the calculation for the intercept point l_{int}, defined in Sect. 4.1, to estimate the arrival time of an intruder given some subset of USVs assigned to delay it. This estimate is not as accurate as performing a full predictive simulation, but it can be performed very quickly, which is useful for evaluating the state of the world after the maximum lookahead time has passed. When multiple intruders are present, we use the minimum time estimate across all intruders to approximate the expected utility $E[t_{delay} | \mathcal{O}_{u_i}, \mathcal{A}_i]$ for the USV team.

During the task re-allocation process, USVs can use heuristics to evaluate task exchanges directly without using predictive simulation at all. For example, delay tasks can be quickly evaluated by performing a predictive simulation with $t_{lookahead}$ set to zero. For guard or observe tasks, the estimated arrival time from the intercept point calculation does not provide useful information, but alternative heuristics can be used as well.

To evaluate guard or observe task exchanges for USV u_i's, we developed a heuristic that prioritizes the exchange of task $h_j \in H_{u_i}$ whose motion goal M_{h_j} is furthest from USV u_i's current motion goal M_{u_i}. This task is offered to the USV u_k whose current motion goal is closest to task h_j's motion goal. If u_k is assigned a delay task, then the heuristic considers the distance between u_k's current motion goal and the motion goal for the new task as multiplied by $\gamma_{occupied}$, to discourage giving too many tasks to USVs that are already responsible for delaying an intruder.

We evaluate the performance of this heuristic candidate selection strategy as an alternative to the predictive simulation in the experimental results in Sect. 5.

4.7 Optimization of behaviors

We use a genetic algorithm (GA) (Holland 1992) to optimize the 12 underlying parameters γ_{guard}, γ_{intr}, γ_{dist}, γ_{lead}, γ_{block}, γ_{max}, $\gamma_{initial_r}$, $\gamma_{occupied}$, γ_{slow_r}, γ_{goal_r}, γ_{fan_r} and γ_{fan_θ}, of the behavior and control action selection policies to further improve the expected utility of the USV policy. The optimization of these parameters allows the USVs to make balanced decisions between guarding a certain location, observing incoming boats, and intercepting and delaying the movement of identified intruders.

For the results presented in Sect. 5, the genetic algorithm was run for 150 generations using a population size of 100 chromosomes, where each chromosome represented a complete set of parameters. The parameters of the initial population were assigned at random, while subsequent populations were bred based on the fitness values of the previous generation. Each chromosome's fitness was measured using the median *delay time* from 250 random simulation runs. We utilized roulette wheel selection to determine the breeding population, and applied genetic operators with a crossover rate of 0.35 and mutation rate of 0.08. Each chromosome in the subsequent population might have some or all of its parameters modified by these operators.

4.8 Complexity analysis

Given the set of tasks H and the set of USVs U, the number possible task allocations is $O(|2^{H^U}|)$, which is too large to explore exhaustively. The number candidate task allocations selected by Algorithm 1 is a more modest $O(|H \times U|)$. This means the number of candidates evaluated increases linearly as the number of USVs or tasks increases. Since we are using predictive simulation to evaluate each candidate, increasing the number of USVs should also increase the time it takes to perform a simulation. As a result, the time complexity of a single task re-allocation step should be $O(|H \times U| \cdot |U|)$, which grows quadratically as the number of USVs increases. This is confirmed by the experimental results in Sect. 5.4.

The algorithm in this paper considers only single-task exchanges (or two-task exchanges, in the special case of swaps). If we consider arbitrary sequences of exchanges of length k, then there are $O(|H \times U|^k)$ such sequences. The number of possible sequences grows exponentially as k increases, meaning it will be prohibitively time consuming to evaluate all such sequences for large values of k. However,

the problem can be made slightly more efficient by applying some selective pruning.

In our implementation in Sect. 5.1, we narrow the set of candidate task allocations by pruning certain types of exchanges. First, we eliminate share exchanges for guard and observe tasks, so only one instance of these tasks will exist at a time. Second, we limit the number of USVs that can simultaneously delay a single intruder, determined by the parameter γ_{max}. Both of these changes reduce the total number of tasks in $\mathcal{A}$ by eliminating redundant assignments while still preserving at least one copy of each task. Additionally, we only consider "swap" exchanges in the case described in Sect. 4.3 so that the number of candidates evaluated remains $O(|H \times U|)$.

5 Results

We have evaluated our planning approach by performing experiments in two simulated scenarios, depicted in Figs. 10 and 11. The motion model used for the experiments is the same motion model used by the predictive simulator, which was defined by Eq. 2 in Sect. 3. Details about the parametrization of the simulator are provided in the experimental setup below, followed by results and discussion for a number of different experiments. We discuss limitations of the experiments in Sect. 5.7.

5.1 Experimental setup

In scenario 1, shown in Fig. 10, the target is positioned within a circular region without any static obstacles. In scenario 2, shown in Fig. 11, the target is positioned near static terrain, restricting the direction of incoming boats. In scenario 1 there are a total of 5 USVs and 3 intruders, while in scenario 2 there are 3 USVs and 2 intruders. In both scenarios, passing boats will continuously enter and leave the operating space, with a maximum of 8 passing boats appearing in the scene at any given time.

At the beginning of each trial, the positions of passing boats are initialized at random locations around the target. During each trial run, new boats appear at random locations along the boundary of the operating space, which is defined as a ring in scenario 1, (with an inner and outer radius of 80 and 100 m), or as two rectangles on the left and right sides of the target in scenario 2 (with a distance of 80 m from the target and a width of 20 m). The rate at which new boats appear is balanced with the rate at which other boats leave the operating space, restricted to the maximum of 8 passing boats.

Each boat's initial trajectory is a path tangent to a randomly sized circle (or semi-circle in scenario 2) surrounding the target with minimum and maximum radius of 30 and 60 m. Unlike non-hostile boats, intruders will change their trajectory and turn towards the target when they pass within 60 m If an intruder passes within 5 m of a USV, it will assume it is being blocked and start approaching the target immediately. Whether a boat will be an intruder or not is determined by the amount of time elapsed during the simulation. Only non-intruder boats will appear during the first 30 s of the simulation, immediately followed by 2 or 3 intruders depending on the scenario. As a result, a group of intruders will always appear at or around the same time in the simulation.

Both intruders and non-intruder boats use the same reactive obstacle avoidance strategy described in Sect. 4.2. For intruders, the parameters γ_{fan_r} and γ_{fan_θ} are set to 10 m and 120 ° respectively, while for non-intruders they are set to 15 m and 180 °. The intruder is given a more aggressive set of parameters allowing it to approach other boats more closely before adjusting its trajectory. For USVs, these parameters are optimized using the genetic algorithm described

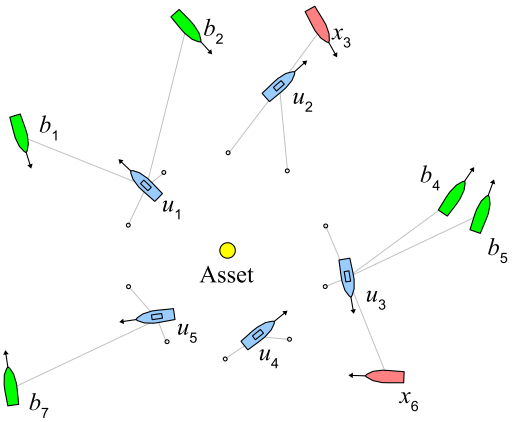


Fig. 10 Scenario 1, five USVs defend a target in an open ocean with several passing boats. Boats x_3 and x_6, identified as intruders, are pursued by USVs u_2 and u_3. The tasks assigned to each USV are depicted as lines

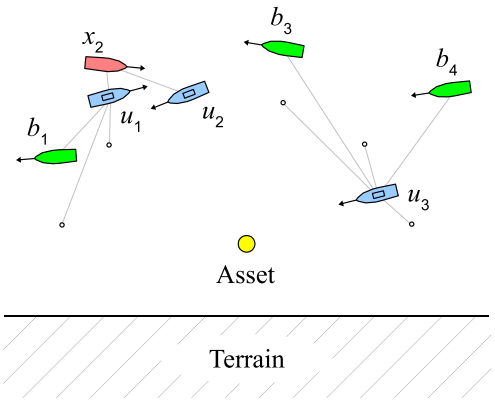


Fig. 11 Scenario 2, three USVs defend a target that is protected by terrain to the south. USVs u_1 and u_2 are actively blocking intruder x_2, reducing the speed at which it approaches the target

in Sect. 4.7. The maximum surge speed for all USVs and other boats is fixed at 10 m/s.

To make the interactions between USVs and intruders more challenging, intruders will perform evasive actions to avoid being blocked. If the intruder is blocked by another boat and diverted away from the target for some time t, where $t > t_{flip}$, the intruder will turn away from the blocking boat and reverse its direction of movement. For each evasive turn, the value of t_{flip} is selected at random, uniformly between 1 and 3 seconds, to introduce non-deterministic behavior into the intruder strategy. We show that intruders which perform evasive turns are more difficult to defend against in Sect. 5.2. However, this model is not guaranteed to be a best-response to the USV team's strategy, and therefore cannot be used to determine the worst-case performance against any theoretical opponent.

The observation classification function provides a simulated probability that each boat is an intruder based on the quality of the observations made by the USV team. The observation quality $\alpha_{b_j} \in [0, 1]$ for boat b_j is initially set to 0 and increases monotonically while USV u_i is within 50 m of boat b_j. If d represents the distance between u_i and b_j, then α_{b_j} increases at a rate of $\delta_{learn}(1 - d/50)$ per second, with the default learning rate $\delta_{learn} = 0.5$. This means it takes at most 5 seconds to obtain an observation quality of $\alpha_{b_j} = 1$ when observing boat b_j from a distance of 30 meters.

The function $P(b_j \in I|\mathcal{O}_{u_i})$ returns a prior probability of 0.05 when $\alpha_{b_j} = 0$, indicating that no observations have occurred. The choice of 0.05 is arbitrary, but is meant to represent a small non-zero chance that each boat could be an intruder. As α_{b_j} increases, the probability $P(b_j \in I|\mathcal{O}_{u_i})$ converges linearly to 1 or 0 depending on whether or not b_i is actually an intruder. Gaussian noise with standard deviation $0.1(1 - \alpha_{b_j})$ is added to the probability function so that the change is non-monotonic. For all simulations, the value p_{alert} for determining whether a boat should be classified as a threat was set to 0.6.

During the predictive simulation, the lookahead time $t_{lookahead}$ is set to 5 s, and the Monte-Carlo sample size $n_{samples}$ is set to 5 for all experiments unless otherwise indicated. These parameters are useful for ensuring the predictive simulation can be executed in real time while still obtaining good performance.

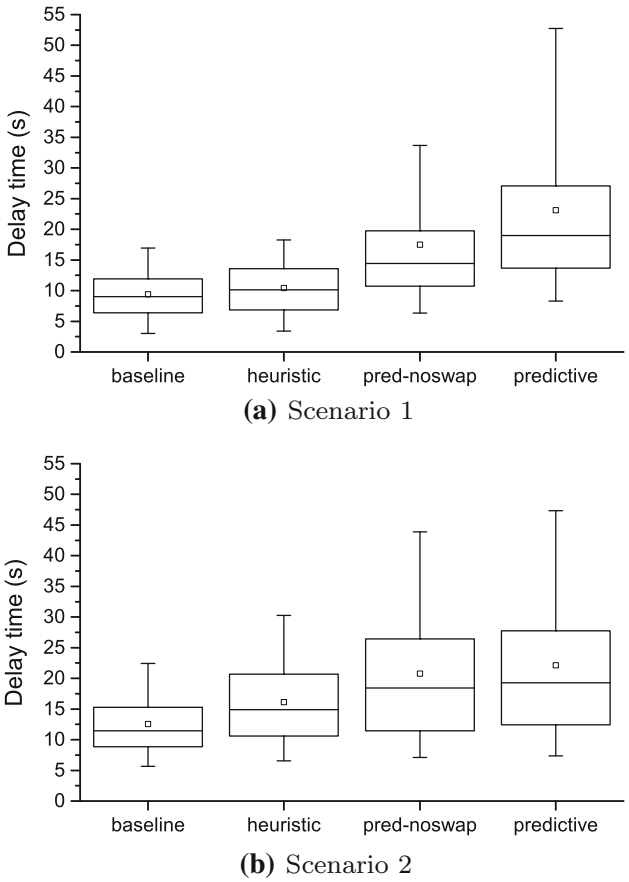


Fig. 12 Average *delay time* across 1,000 randomly seeded trials for USV teams using *baseline*, *heuristic*, *pred-noswap* or *predictive* strategies

(1) the *predictive* strategy uses the complete task allocation strategy described in Sect. 4,
(2) the *heuristic* strategy does not use predictive simulation, but performs task allocation based on the heuristic approach described in Sect. 4.6,
(3) the *baseline* strategy does not perform task exchanges at all, instead each USV waits at its guard location until an intruder is identified, then delay tasks are assigned using the heuristic in Sect. 4.6,
(4) the *pred-noswap* strategy is a variant of the predictive strategy that does not use the conditional "swap" exchange discussed in Sect. 4.3.

5.2 Strategy comparison

To evaluate the performance of the approach defined in Sect. 4, we performed experiments on several strategy variants. All the strategies evaluated use the same motion goal and control action selection policies defined in Sects. 4.1 and 4.2, but they differ in how the task allocation process is performed:

Figures 12 a) and b) show the average *delay time* across 1,000 randomly generated trial for each of the four different strategies. The box plots show the median, upper and lower quartile of the data set, while the whiskers mark the 5th and 95th percentiles. The mean value is marked with a small square in each figure.

As expected, the *predictive* and *pred-noswap* strategies performed best, followed by the *heuristic* strategy, while the

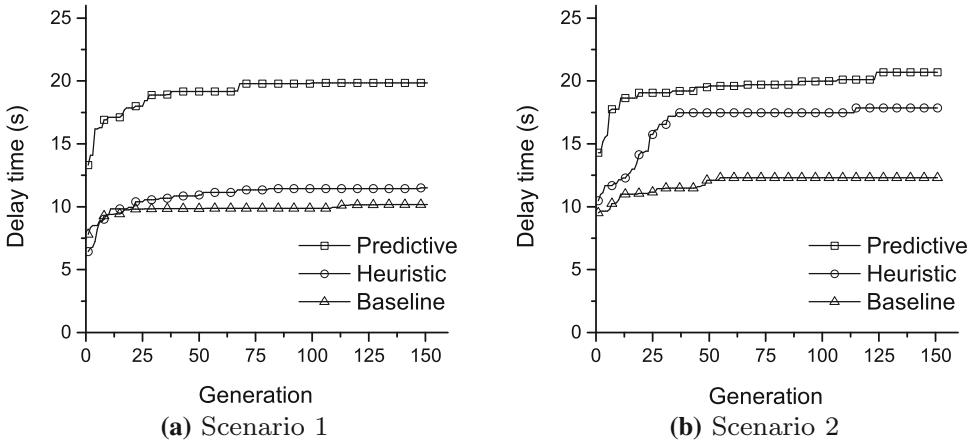


Fig. 13 Median *delay time* for each generation of the genetic algorithm when optimizing strategies for scenarios 1 and 2. Results shown for the best-performing chromosome in the population

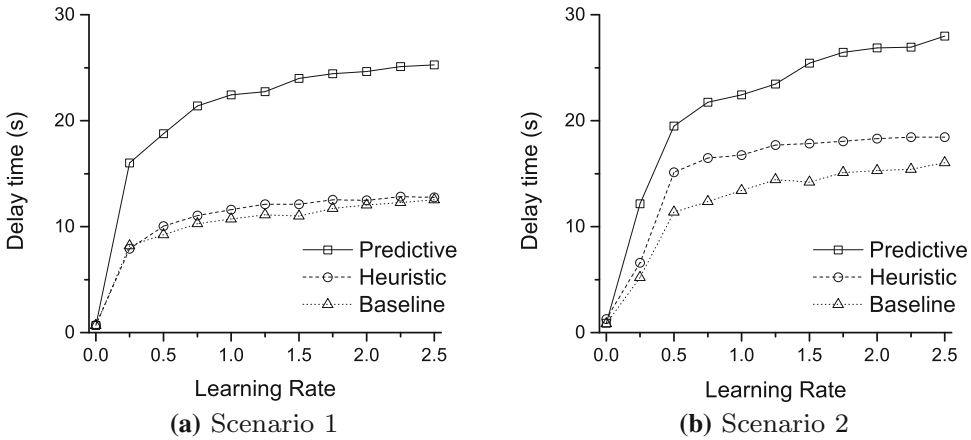


Fig. 14 Median *delay time* across 1000 randomly seeded trials for *baseline*, *heuristic* and *predictive* strategies as the learning rate δ_{learn} increases

baseline strategy performed worst. For scenario 1, the predictive strategy increased the median *delay time* by 110 % compared to the baseline strategy and by 87 % compared to the heuristic strategy. For scenario 2, the increase was 68 % compared to the baseline strategy and 29 % compared to the heuristic strategy. The predictive strategy also performed better when the "swap" exchange was included, increasing *delay time* by 31 % in scenario 1 and 4.5 % in scenario 2.

The difference in *delay time* between the heuristic and predictive strategies is less for scenario 2 when compared to scenario 1, possibly due to the smaller number of choices during the task allocation step, decreasing the likelihood of the heuristic selecting a bad candidate.

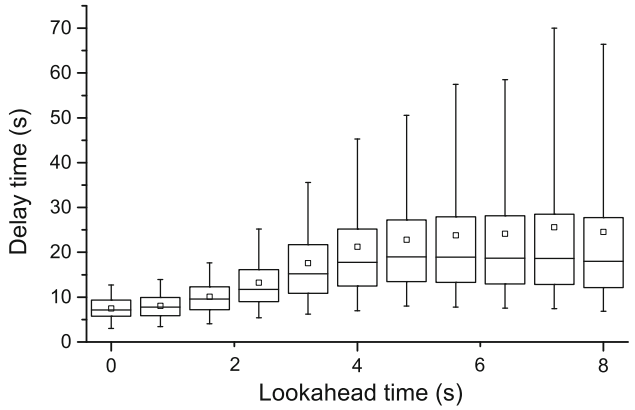
To optimize the parameter set Γ for each of the three main strategy types, the genetic algorithm described in Sect. 4.7 was performed six separate times, once for every combination of strategy and scenario. An exception was made for the *pred-noswap* strategy, which was given the same parameter set as the *predictive* strategy. The change in performance across 150 generations is shown in Fig. 13a and 13b for scenarios 1 and 2, respectively. Most of the gains occurred within the first 50 generations of the algorithm.

Figures 14 a) and b) show the average *delay time* as the learning rate δ_{learn} is varied. The utility of the USV team decreases as the learning rate decreases, and increases as the learning rate increases. This is true for all of the strategies evaluated. However, the *predictive* strategy remains the preferred strategy for all $\delta_{learn} > 0$. The special case where $\delta_{learn} = 0$ means that the USV team is unable to identify the intruders through observation, resulting in a delay time of zero.

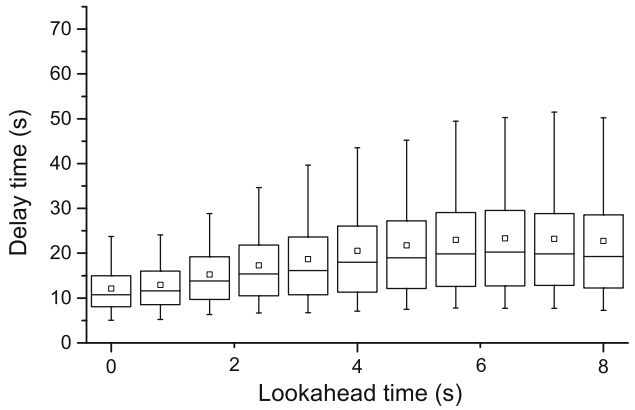
Table 1 Median *delay time* for *baseline*, *heuristic*, *predictive* strategies in scenario 1 against intruders that perform evasive turns and those that don't

	$Evasive$	$Non-evasive$
Baseline	9.0s	21.2
Heuristic	10.1s	67.6
Predictive	19.0s	158.3

Table 1 shows the average *delay time* when each of the strategies is performed against a set of evasive or non-evasive intruders in scenario 1. As expected, non-evasive intruders are delayed from reaching the target for longer than evasive intruders. In practice, this occurred because the non-evasive intruders would become locked in a continuous blocking pattern with a single USV, looping around the target for an extended period of time. The evasive intruder was able to



(a) Scenario 1



(b) Scenario 2

Fig. 15 Average *delay time* across 1,000 randomly seeded trials for the *predictive* strategy as the lookahead time, $t_{lookahead}$, increases

break this pattern by intermittently flipping direction, reaching the target more quickly. We use the evasive intruder model for all the remaining experiments in this paper.

5.3 Running-time tradeoff

Figures 15 and 16 show the change in USV team utility as the duration and sample size of the predictive simulation increases. Generally, performing more and better simulations result in higher utility for the USV team, but takes longer to execute.

Figures 15 a) and b) show the benefit of increasing the lookahead time, $t_{lookahead}$, for the *predictive* strategy in scenario 1 and 2. Longer values $t_{lookahead}$ correspond with more time spent evaluating each predictive simulation. The predictive simulation will run for a maximum duration of $t_{lookahead}$, after which the static evaluation function (defined in Sect. 4.6) is performed to quickly estimate the value of the remainder of the simulation. When $t_{lookahead} = 0$, no predictive simulation is run at all, and the static evaluation function is performed immediately.

Increasing $t_{lookahead}$ from 0 s to 8 s offers a 151 % increase in the median utility for scenario 1, and a 79 % increase in the median utility for scenario 2. The benefit of increasing $t_{lookahead}$ starts to diminish at around five seconds, possibly due to the gradual accumulation of errors in the simulation.

Figures 16 a) and b) show the benefit of increasing the sample size, n_{sample}, for the *predictive* strategy in scenario 1 and 2. The value of n_{sample} corresponds with the number of possible worlds evaluated via Monte-Carlo sampling. Increasing n_{sample} from 1 to 8 offers a 25 % increase in the median utility for scenario 1, and a 14 % increase in the median utility for scenario 2.

Increasing $t_{lookahead}$ or n_{sample} individually should result a linear increase in running time for the predictive simulation. However, if the goal is to maximize USV utility, the tradeoff between the quality of the evaluation and the running time of the task re-allocation step should be considered. Both $t_{lookahead}$ and n_{sample} suffer from diminishing returns as their value increases; each additional second added to $t_{lookahead}$ or sample added to n_{sample} is less valuable than the previous. This suggests that the ideal selection of values for $t_{lookahead}$ and n_{sample} will vary depending on the computational power available.

One motivation for minimizing the running time of the predictive simulation is to reduce the time between reallocation steps, t_{alloc}. As shown in Fig. 17, reducing t_{alloc} has a positive effect on utility for both the heuristic and predictive strategies. However, Fig. 17 also shows that reducing t_{alloc} alone is not sufficient to maximize utility, since the heuristic strategy is out-performed by the predictive strategy even when USVs are allowed to exchange tasks at very high frequency. This suggests that, at least for some range of values t_{alloc}, time is better spent carefully evaluating which tasks to exchange instead of exchanging tasks quickly based on an inexpensive heuristic.

5.4 Scalability

Figures 18 and 19 show the effect that increasing the number of USVs or passing boats simultaneously appearing in the scene has on the computational workload of *predictive* strategy. Running time was measured using an Intel Core 2 Q6600 Quad processor with a 2.4 GHz clock speed with the algorithm running in a single thread. Results are shown for scenario 1, but similar results should be expected for other scenarios.

Figure 18 shows that increasing the number of USVs causes a roughly linear increase in the number of candidate task allocations evaluated and a polynomial increase in the running time. As explained in Sect. 4.8, the number of candidate task allocations evaluated by predictive simulation is at most $O(|U \times H|)$, where U is the set of USVs and H is the set of tasks. The number of USVs is directly proportional to

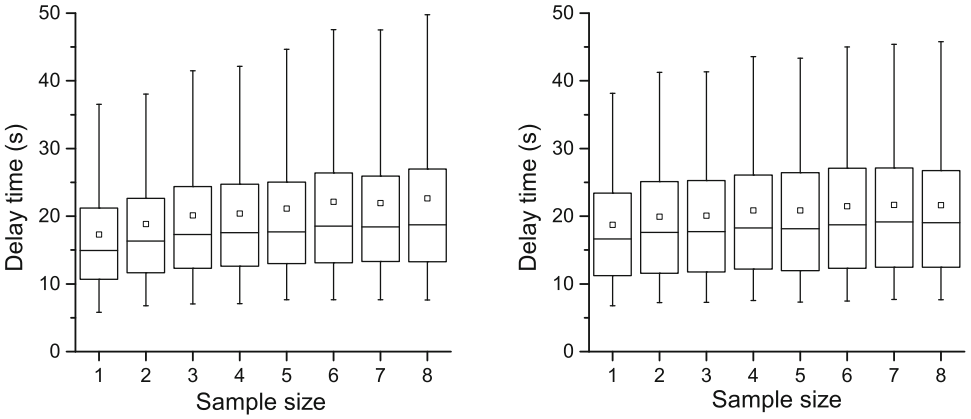


Fig. 16 Average *delay time* across 1,000 randomly seeded trials for the *predictive* strategy as the number of samples n_{sample} increases

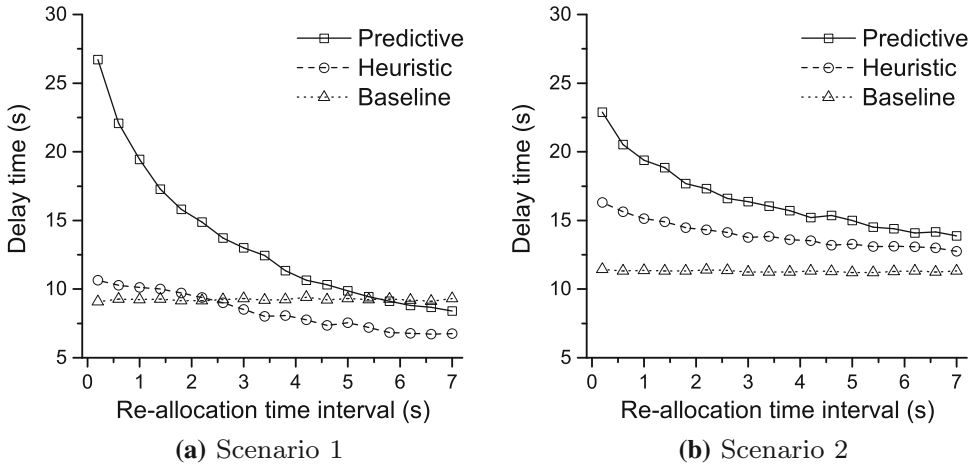


Fig. 17 Median *delay time* across 1,000 randomly seeded trials for the *baseline*, *heuristic* and *predictive* strategies as the re-allocation time interval t_{alloc} increases

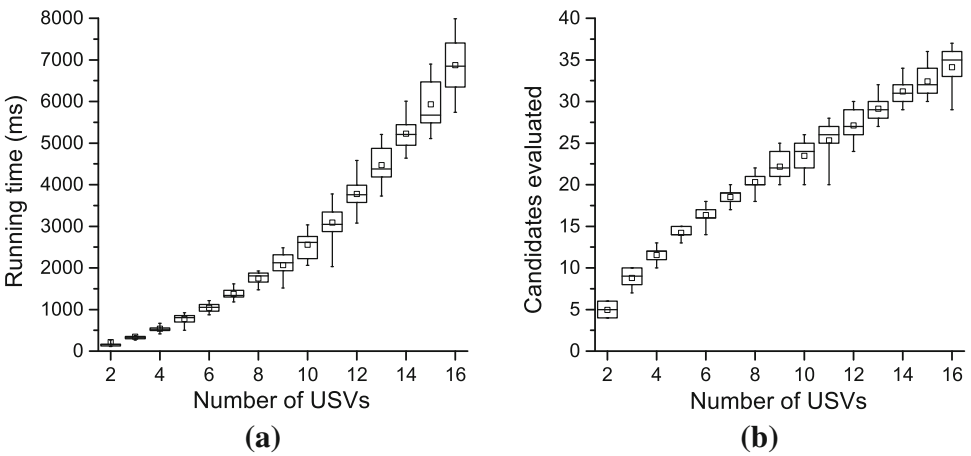


Fig. 18 Average running time and number candidate task allocations evaluated across 1,000 randomly seeded trials, as the number of USVs varies, for the *predictive* strategy in scenario 1

the number of task exchanges that are considered. Similarly, increasing the number of USVs increases the running time of each predictive simulation. As a result, doubling the number of USVs from 4 to 8 increases running time of the task re-allocation step by 3.4 times, from 525 ms to 1805 ms.

Figure 19 shows the effect of increasing the number of passing boats in scenario 1. Since each passing boat must be assigned an observe or delay task, increasing the number of boats should result in a linear increase in the number of tasks, similar to increasing the number of USVs. However, the effect is much less pronounced than in Fig. 18, because a single USV does not have to evaluate every new task that is added. Thus, the cost of adding a USV is greater than the cost of adding a new boat or task to evaluate.

The median running time for a single USV to compute a task re-allocation step when there are 5 USVs and 8 passing

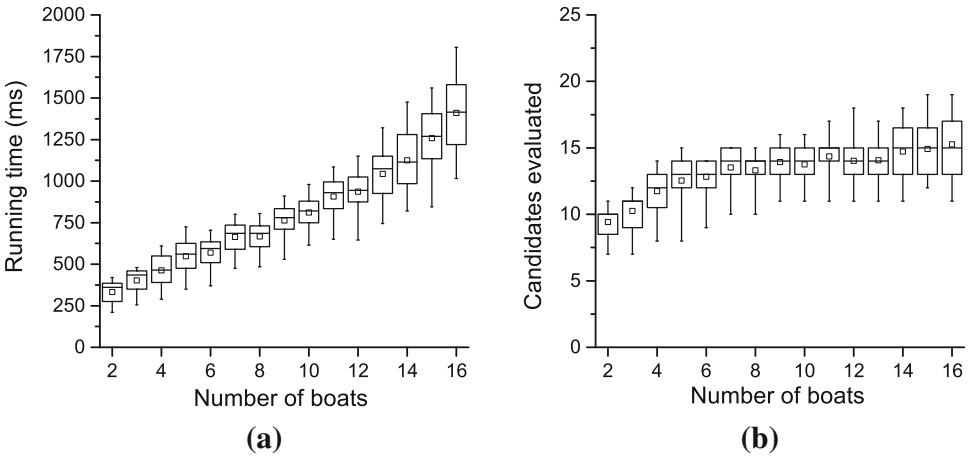


Fig. 19 Average running time and number candidate task allocations evaluated across 1,000 randomly seeded trials, as the number of passing boats varies, for the *predictive* strategy in scenario 1

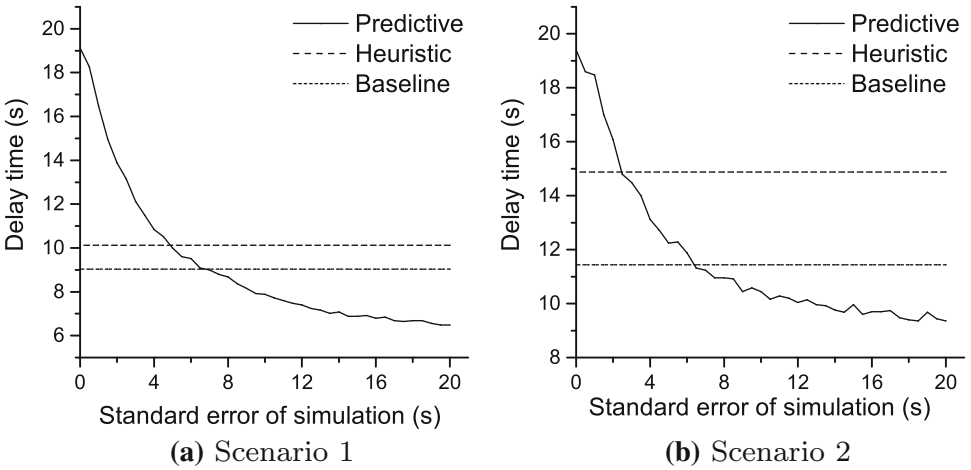


Fig. 20 Median *delay time* across 1,000 randomly seeded trials for USV teams using the *predictive* strategy when normally distributed *error* is added to the result of the predictive simulation

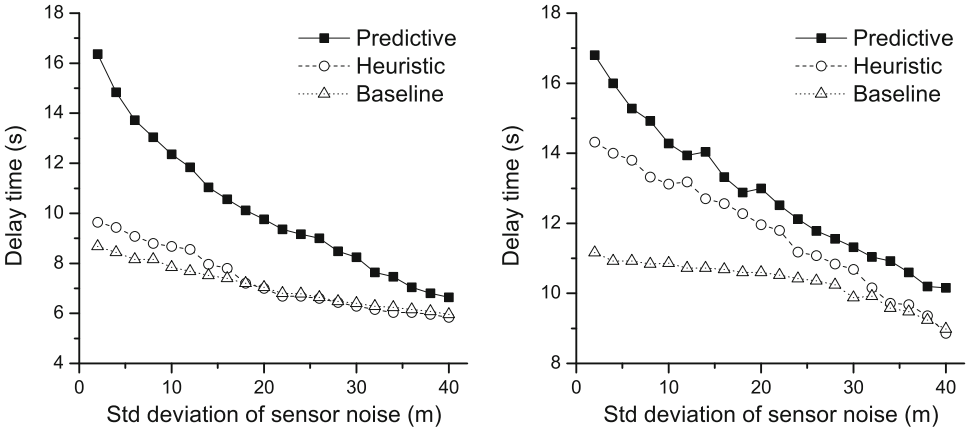
boats (the default parameters for scenario 1) was 805 ms, which is within the 1 second allocated in the experimental setup. This result, combined with the low-order polynomial complexity of the algorithm, suggests that the approach is efficient and that online computation is feasible on equivalent hardware.

5.5 Robustness

The reliability of the predictive strategy depends on how accurately the predictive simulation is able to estimate the expected value of candidate task allocations. Since the simulator used for our experiments and and the predictive simulator used by USVs both use the same motion model, it is relatively easy for the USVs to estimate the expected value. However, we cannot expect the same level of fidelity in the real world. To determine what effect inaccuracies in the predictive simulation have on performance, we tried several ways to make the simulation less reliable: (1) adding normally distributed error to the utility value returned by the predictive simulation, (2) adding normally distributed error to the USV sensor measurements and (3) using the incorrect opponent model in the predictive simulation.

The results of the first experiment are shown in Fig. 20. There is a significant drop in performance as the utility value returned by the predictive simulation becomes noisier. For scenario 1, the performance of the predictive strategy drops below that of the purely heuristic strategy when standard error exceeds 4.5 s. For scenario 2, this occurs when the standard error exceeds 3.0 s.

For the results in Fig. 21, we added normally distributed error to the USVs' sensor measurements of the locations (x and y coordinates) of the passing boats. The labels on the figure display the standard deviation of this error at an observation range of 80 m or greater. For ranges between 80 m and 0 m, the amount of error was decreased linearly based on distance, so that the measurements had no error if taken at 0 m range. As mentioned in Sect. 4, each USV uses a Kalman filter to produce estimates of the boats' locations based on this sensor data. The results show a gradual decrease in utility as the sensor measurements become noisier, however the predictive strategy remains the best-performing strategy in spite of the noise.



(a) Scenario 1 **(b)** Scenario 2

Fig. 21 Median *delay time* across 1,000 randomly seeded trials for USV teams using the *predictive* strategy when normally distributed *error* is added to USV sensor measurements

To evaluate how sensitive the predictive simulation is to the accuracy opponent model, we created three variations on the opponent model by modifying the intruders' level of aggression:

(1.) the "normal" model is the standard intruder model described in Sect. 5.1,
(2.) the "timid" model avoids collisions more actively and performs evasive turns less frequently, increasing γ_{fan_r} and γ_{fan_θ} by 10 % and increasing t_{flip} by 100 % compared to the normal model,
(3.) the "aggressive" model avoids collisions less actively and performs evasive turns more frequently, decreasing γ_{fan_r} and γ_{fan_θ} by 10 % and decreasing t_{flip} by 50 % compared to the normal model.

For each opponent model, we produced a complementary predictive strategy, named Pred(N), Pred(T) and Pred(A), where the predictive simulation used the normal, timid, or aggressive model of the intruder respectively. We also produced a fourth strategy, Pred(R), where one of the three intruder models is selected at random using Monte-Carlo sampling at the start of each predictive simulation. For each of the strategies, the GA parameters Γ were tuned for the normal intruder model.

The results in Table 2 show the median *delay time* when using each of the different task re-allocation strategies against each of the three intruder models. As expected, the USV team performed best when the correct opponent model was used. Additionally, the Pred(R) strategy was the second-best performing strategy in several cases, and performed better than the heuristic strategy in all of the cases.

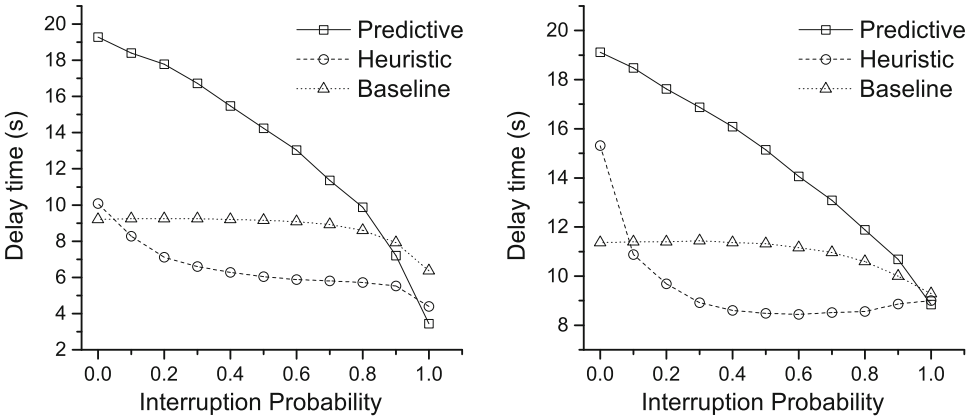
One key result of this experiment is that using the correct model of the intruder can have a significant impact on the performance of the algorithm. For example, using the Pred(A) strategy against the timid intruder causes the predictive strategy to perform worse than the heuristic strategy.

Table 2 Median *delay time* across 1,000 randomly seeded trials for USV teams using the *predictive*, *heuristic*, and *baseline* strategies against intruders with timid, normal or aggressive behaviors

	Normal (s)	Timid (s)	Aggressive (s)
Scenario 1			
Pred(N)	19.2	17.5	13.8
Pred(T)	18.5	36.9	12.8
Pred(A)	11.4	10.2	15.5
Pred(R)	18.1	28.4	13.8
Heuristic	10.0	10.6	9.5
Baseline	9.2 s	9.0	9.0
Scenario 2			
Pred(N)	19.3	16.9	18.2
Pred(T)	18.7	22.8	15.6
Pred(A)	16.6	14.4	20.8
Pred(R)	19.0	18.2	19.0
Heuristic	15.0	15.4	16.1
Baseline	11.4	11.0	13.9

This is because the Pred(A) strategy assumes the intruder is aggressive, when it is actually timid, and makes worse predictions than the other predictive strategies. In contrast, Pred(T) was the best-performing strategy for this situation, while Pred(R) was the second-best.

These results suggest that when the intruder model is not exactly known, decent performance can be obtained by using Monte-Carlo sampling over the set of possible intruder models. Additionally, using a model that closely approximates the correct model, (e.g. using the normal model to approximate the timid intruder, or the aggressive model to approximate the normal intruder), results in better utility than using a model that poorly approximates the correct model (e.g. using the aggressive model to approximate the timid intruder). This suggests that the predictive simulator still provides useful



(a) Scenario 1 **(b)** Scenario 2

Fig. 22 Median *delay time* across 1,000 randomly seeded trials for USV teams using the *predictive*, *heuristic*, and *baseline* strategies when communication between USVs is interrupted

information even if the model used does not exactly match the behavior of the opponent.

5.6 Interrupted communication

To determine the effect of communication interruptions on USV team performance, we performed simulation runs where the communication link between each pair of USVs had a random chance of being interrupted. Fig. 22 shows the average utility for each of the three strategies when the chance of communication being interrupted is varied. The interruption probability determines the likelihood of an interruption event occurring between a pair of USVs during a 1 s time interval. If an interruption event occurs, the two USVs cannot exchange information for the whole 1 s interval. Each interruption event is modeled as statistically independent, so repeated interruptions can block communication between two USVs indefinitely.

As the probability of interruption increases, the performance of the USV team is negatively impacted. This is true for all three strategies. Going from no interruption, to interruptions occurring with 0.5 probability every second, the mean *delay time* of the predictive strategy drops from 19.2 s to 14.2 s in scenario 1, and from 19.1 s to 15.2 s in scenario 2, a difference of 26 and 20 % respectively. The impact on the heuristic strategy is more significant, dropping from from 10.1 s to 6.0 s in scenario 1, and from 15.3 s to 8.5 s in scenario 2, a difference of 40 and 44 % respectively.

For both scenarios, the difference in performance between the heuristic and predictive strategies decreases as the probability of interruption increases. Since task exchanges are not possible without communication between agents, high interruption renders the additional predictive simulation less effective. The predictive strategy still out-performs both the heuristic baseline strategies in all but the most extreme levels of interruption. The baseline strategy also performs better than the heuristic strategy when the interruption probability exceeds 0.1. This may be due to the fact that the baseline strategy uses very few task exchanges to begin with and is therefore better optimized for situations with low communication.

5.7 Limitations

One limitation of our experimental design is that we do not directly incorporate concurrency into the simulation. The implementation is single-threaded, meaning that the task reallocation step for each USV is performed sequentially. As a result, some of the concurrency issues that may be experienced in a real world scenario are not directly evaluated by our experiments.

Due to concurrency it is possible for the system to cycle between two or more locally sub-optimal task allocations. This can occur when two different USVs perform separate exchanges without being mutually aware of the other's decision. The two exchanges together may have a lower expected utility than the original task allocation, causing the exchanges to be reversed in the subsequent iteration. In this paper, we did not develop mechanisms specifically to deal with these types of cycles, however, this and other concurrency issues are something that should be evaluated more closely in future work.

Another limitation of our experimental design is that simulator used during experiments uses the same motion model for boat physics as the predictive simulator used by the USVs to evaluate candidate task allocations. In the real world, it may be unrealistic to expect the same level of fidelity from a predictive simulation. Meanwhile, the results in Sect. 5.5 suggest that it is important for the predictive simulator to provide good estimates of utility. To address this issue in future work, we intend to perform experiments using a higher fidelity simulation environment (Thakur and Gupta 2011; Thakur et al. 2012).

6 Conclusions and future work

We have developed a decentralized, contract-based planning approach for protecting an asset by a team of USVs operating in an environment with civilian traffic. The developed planner is able to deal with uncertainty about which boats are actual intruders and accounts for complex interactions between USVs and intruders when allocating tasks. The planner combines high-level task allocation with low-level user defined behaviors by using model-predictive simulations to evaluate plan performance. The planner is capable of computing the task allocation efficiently, is scalable to large teams of USVs, and can be optimized for a specific mission.

We have evaluated the performance of the planner in two different simulation scenarios. In both scenarios, the developed model-predictive planner had a significant performance advantage compared to the baseline and heuristic strategies. We also evaluated the scalability of the planner for large numbers of USVs and passing boats, explored the trade-off between plan quality and execution time, and evaluated the planner's robustness in dealing with noisy sensor data, inaccurate opponent models and high levels of communication interruption.

In our results, we show that by carefully tuning and integrating the model-predictive simulation into the task allocation process, features such as differential constraints and sensing uncertainty can be directly considered during task allocation and still run efficiently. We also demonstrate that the use of model-predictive simulation leads to significantly higher performance and robustness than the pure use of task-tailored heuristic rules. We show that through careful Monte-Carlo sampling over the distribution of possible worlds, the model-predictive simulation produces better results despite the fact that the task-tailored heuristics are computationally more efficient. These results suggest that time is better spent carefully evaluating which tasks to exchange instead of exchanging tasks quickly based on an inexpensive heuristic.

In future work, we would like to explore ways to improve the planner under high communication uncertainty. Since explicit task exchanges between agents are not possible when communication is interrupted, it may be beneficial to blend contract-based task exchanges with purely local task assignment as communication between agents becomes less reliable. Another idea worth investigating is whether the algorithm should always perform the best exchange with the boats currently in communication, or wait a bit longer for openings in communication to perform task exchanges with higher expected value. More work should also be done to evaluate the effect of concurrency on task exchanges in an experimental setting.

Significant gains in performance might also come from improved sampling of candidate task allocations during the task re-allocation process. The current algorithm behaves like a local search, evaluating a small number of variations based on the current task allocation. Stochastic sampling beyond this local set of candidates could reduce the likelihood of becoming stuck in a local minimum, increasing the utility of the USV team.

Additional study can be carried out using more sophisticated heuristic approaches. In particular, a heuristic approach can be developed that can reason about longer sequences of task exchanges than the model-predictive strategy in the same amount of time. It would be worth investigating whether careful and thus slower reasoning about fewer and/or shorter sequences of task exchanges is more beneficial than fast reasoning about longer and/or a larger number of plans of task exchanges.

Finally, we would like to extend this approach to more complex scenarios, such as defending a moving target, accounting for coordinated behavior from the intruders, incorporating static obstacles with complex shapes into the environment, or by applying the algorithm in the ground and aerial vehicles domains. To do this, the blocking and guarding behaviors may need to be modified and new tasks may need to be created to account for changes in the target's position, incorporate path planning to navigate around obstacles, or model the motion constraints of different vehicles. Adding more sophisticated behaviors and corresponding tasks for the current scenarios, such as cooperative blocking for two or more USVs, may also improve performance without any changes to the high level task allocation process.

Acknowledgments This research has been supported by ONR N000 14-10-1-0585, N00014-12-1-0494, N00014-12-1-0430, N00014-13-1-0597, and ARO W911NF-12-1-0471 and W911NF-11-1-0344 grants. The opinions expressed in this paper are those of the authors and do not necessarily reflect opinions of the sponsors.

References

Bertaska, I. R., Alvarez, J., Sinisterra, A. J., von Ellenrieder, K., Dhanak, M., Shah, B. C., et al. (2013). Experimental evaluation of approach behavior for autonomous surface vehicles. In *6th Annual Dynamic Systems and Control Conference (DSCC '13)* Stanford University, Palo Alto. October 21–23.

Bošanský, B., Lisý, V., Jakob, M., & Pěchouček, M. (2011). Computing time-dependent policies for patrolling games with mobile targets. In *10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'11)*.

Corfield, S.J., & Young, J.M., (2006). Unmanned surface vehicles-game changing technology for naval operations. In *Advances in unmanned arine vehicles* (pp. 311–328). London: Institution of Engineering and Technology.

Dias, M.B. (2004). *Traderbots: A new paradigm for robust and efficient multirobot coordination in dynamic environments*. PhD thesis, Carnegie Mellon University.

Dias, M. B., Zlot, R., Kalra, N., & Stentz, A. (2006). Market-based multirobot coordination: A survey and analysis. *Proceedings of the IEEE, 94*(7), 1257–1270.

Fang, F., Jiang, A.X., & Tambe, M. (2013). Designing optimal patrol strategy for protecting moving targets with multiple mobile resources. In *International Workshop on Optimisation in Multi-Agent Systems (OPTMAS)*.

Gerkey, B. P., & Matarić, M. J. (2002). Sold!: Auction methods for multirobot coordination. *IEEE Transactions on Robotics and Automation*, *18*(5), 758–768.

Gerkey, B. P., & Matarić, M. J. (2004). A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, *23*(9), 939–954.

Holland, J. H. (1992). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Cambridge, MA, USA: MIT Press.

Jakob, M., Vaněk, O., Hrstka, O., & Pěchouček, M. (2012). Agents vs. pirates: multi-agent simulation and optimization to fight maritime piracy. In *11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'12)* (pp. 37–44). *International Foundation for Autonomous Agents and Multiagent Systems*.

Kalra, N., Ferguson, D., & Stentz, A. (2005). Hoplites: A market-based framework for planned tight coordination in multirobot teams. In *IEEE International Conference on Robotics and Automation (ICRA'05)* (pp. 1170–1177). IEEE.

Mosteo, A. R., & Montano, L. (2010). *A survey of multi-robot task allocation*. Instituto de Investigación en Ingeniería de Aragón.

Parker, L. E. (2008). Multiple mobile robot systems. In *Springer handbook of robotics* (pp. 921–941). Berlin: Springer.

Portugal, D., & Rocha, R. (2011). A survey on multi-robot patrolling algorithms. In *Technological Innovation for Sustainability* (pp. 139–146).

Portugal, D., & Rocha, R.P. (2011). On the performance and scalability of multi-robot patrolling algorithms. In *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)* (pp. 50–55). IEEE.

Raboin, E., Švec, P., Nau, D. S., & Gupta, S. K. (2013). Model-predictive target defense by team of unmanned surface vehicles operating in uncertain environments. In *IEEE International Conference on Robotics and Automation (ICRA'13)*.

Sandholm, T. (1998). Contract types for satisficing task allocation. In Proceedings of the AAAI spring symposium: Satisficing models (pp. 23–25).

Shieh, E.A., An, B., Yang, R., Tambe, M., Baldwin, C., DiRenzo, J., Maule, B., & Meyer, G. (2012). Protect: An application of computational game theory for the security of the ports of the united states. In *AAAI Conference on Artificial Intelligence*.

Shoham, Y., & Leyton-Brown, K. (2010). *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge: Cambridge University Press.

Simetti, E., Turetta, A., Casalino, G., Storti, E., & Cresta, M. (2010). Protecting assets within a civilian harbour through the use of a team of USVs: Interception of possible menaces. OCEANS.

Simmons, R., Apfelbaum, D., Fox, D., Goldman, R.P., Haigh, K.Z., Musliner, D.J., Pelican, M., & Thrun, S. (2000). Coordinated deployment of multiple, heterogeneous robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)* (volume 3, pp. 2254–2260). IEEE.

Smith, R. G. (1980). The contract net protocol: high-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, *100*(12), 1104–1113.

Švec, P., & Gupta, S. K. (2012). Automated synthesis of action selection policies for unmanned vehicles operating in adverse environments. *Autonomous Robots*, *32*(2), 149–164.

Švec, P., Schwartz, M., Thakur, A., & Gupta, S. K. (2011). Trajectory planning with look-ahead for unmanned sea surface vehicles to handle environmental disturbances. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'11)*.

Švec, P., Shah, B. C., Bertaska, I. R., Alvarez, J., Sinisterra, A. J., von Ellenrieder, K., et al. (2013). Dynamics-aware target following for an autonomous surface vehicle operating under COLREGs in civilian traffic. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'13)* Tokyo. November 3–8.

Švec, P., Thakur, A., & Gupta. S. K. (2012). USV trajectory planning for time varying motion goal in an environment with obstacles. In *ASME 2012 International Design Engineering Technical Conferences (IDETC) & Computers and Information in Engineering Conference (CIE)*.

Švec, P., Thakur, A., Shah, B. C., & Gupta, S. K. (2013). Target following with motion prediction for unmanned surface vehicle operating in cluttered environments. Autonomous Robots, 2013. Retrived for publication. doi:10.1007/s10514-013-9370-z.

Tang, F., & Parker, L. E. (2007). A complete methodology for generating multi-robot task solutions using asymtre-d and market-based task allocation. In *IEEE International Conference on Robotics and Automation (ICRA'07)* (pp. 3351–3358).

Thakur, A., & Gupta, S. K. (2011). Real-time dynamics simulation of unmanned sea surface vehicle for virtual environments. *Journal of Computing and Information Science in Engineering*, *11*(3), 031005.

Thakur, A., Švec, P., & Gupta, S. K. (2012). Gpu based generation of state transition models using simulations for unmanned surface vehicle trajectory planning. In *Robotics and Autonomous Systems*.

Vanek, O., Bosansky, B., Jakob, M., Lisy, V., & Pechoucek. M. (2012). Extending security games to defenders with constrained mobility. In *Proceedings of AAAI Spring Symposium GTSSH*.

Zhang, Y., & Meng, Y. (2010). A decentralized multi-robot system for intruder detection in security defense. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'10)* (pp. 5563–5568). IEEE.

Zlot, R., & Stentz, A. (2006). Market-based multirobot coordination for complex tasks. *The International Journal of Robotics Research*, *25*(1), 73–101.

Eric Raboin is a Ph.D. student at the University of Maryland, College Park. He received his Master of Science (M.S.) degree in Computer Science from the University of Maryland, College Park in 2012, and his Bachelor of Science (B.S.) degree in Computer Science from the University of Massachusetts Amherst in 2007. His research interests include multi-agent planning, game theory, machine learning and robotics.

Petr Švec is an Assistant Research Scientist at the University of Maryland, College Park. His research interests include robotics, motion planning and control theory, artificial intelligence for task planning, and machine learning. Dr. Švec received a master's degree in Computer Science (this was an integrated bachelor's and master's degree awarded after three years of study of Mechanical Engineering and two years of study of Computer Science) from the Faculty of Mechanical Engineering at the Brno University of Technology in 2003, followed by Ph.D. in 2007 with a focus on robot motion planning from the same institute. He also received a bachelor's degree and a second master's degree in Computer Science from the Masaryk University in 2006. In addition, he spent one year as a Graduate Research Assistant at the Department of Computer Science at the University of Bristol in the UK.

Dana S. Nau is a Professor at the University of Maryland, in the Department of Computer Science and the Institute for Systems Research. He does Artificial Intelligence research, especially in the areas of automated planning and game theory. He is both an ACM Fellow and an AAAI Fellow. Some of his best-known accomplishments include the discovery of pathological game trees, in which looking farther ahead produces worse decision-making; applications of AI in automated manufacturing; the strategic planning algorithm that Bridge Baron used to win the 1997 world championship of computer bridge; the SHOP, SHOP2, and Pyhop automated-planning systems; and the graduate-level textbook Automated Planning: Theory and Practice.

Satyandra K. Gupta is a Professor in the Department of Mechanical Engineering and the Institute for Systems Research at the University of Maryland, College Park. He is the director of the Advanced Manufacturing Laboratory. He was the founding director of the Maryland Robotics Center. Prior to joining the University of Maryland, he was a Research Scientist in the Robotics Institute at Carnegie Mellon University. Dr. Gupta's interest is broadly in the area of automation. He is specifically interested in automation problems arising in Engineering Design, Manufacturing, and Robotics. He is a fellow of the American Society of Mechanical Engineers (ASME). Dr. Gupta has received several honors and awards for his research contributions. Representative examples include: a Young Investigator Award from the Office of Naval Research in 2000, a Robert W. Galvin Outstanding Young Manufacturing Engineer Award from the Society of Manufacturing Engineers in 2001, a CAREER Award from the National Science Foundation in 2001, a Presidential Early Career Award for Scientists and Engineers (PECASE) in 2001, Invention of the Year Award in Physical Science category at the University of Maryland in 2007, Kos Ishii-Toshiba Award from ASME Design for Manufacturing and the Life Cycle Committee in 2011, and Excellence in Research Award from ASME Computers and Information in Engineering Division in 2013. He has also received six best paper awards at conferences.