Anti-Submarine Warfare Planning Using Public Belief States and Self-Play

Christoffer Limér*, Joel Brynielsson*†, Mika Cohen*†, Felix Rydell*
*FOI Swedish Defence Research Agency, SE-164 90 Stockholm, Sweden
†KTH Royal Institute of Technology, SE-100 44 Stockholm, Sweden
Email: christoffer.limer@foi.se, joel@kth.se, mikac@kth.se, felix.rydell@foi.se

***Abstract*—We consider the problem of how to move active sonars unpredictably in pursuit of a stealthy underwater vehicle. The search problem is formalized as an imperfect-information game played on a discretized nautical chart with fine-grained hydroacoustics. The game is solved approximately using public belief states and self-play following a game-theoretically sound approach. The solution method is shown empirically to approximate the Nash equilibrium in a restricted scenario small enough to be solvable with tabular methods from algorithmic game theory.**

***Index Terms*—Anti-submarine warfare; mobile sensor planning; game theory; public belief state; self-play.**

I. Introduction

It has long been assumed that unpredictability in anti-submarine warfare (ASW) requires psychological intuition and cunning [1]. Today, however, computer algorithms can outperform human players in challenging bluffing games. Most notably, bots are now unbeatable in poker, including the most demanding forms of poker [2]. This invites a compelling question: Could poker-playing algorithms also play a game of hide-and-seek with a submarine?

In this paper, we apply game-theoretic self-play techniques from poker to the problem of how to search unpredictably for a small, stealthy underwater vehicle with search vehicles equipped with active sonars. To the best of our knowledge, this represents the first publicly reported application of superhuman poker AI outside poker and poker-like games, to a real-world decision problem.

Existing (game-theoretic) approaches for generating unpredictable search paths (cf. [3], [4], [5], [6], [7]) apply only to passive (silent) sonars. However, in modern ASW, active sonars (that emit a powerful sound, typically revealing the position of the sonar) play an essential role, as they are the only reliable means of detecting small, stealthy underwater threats in real-time.[1]

We formalize the planning problem as a fog-of-war strategy game between an intruder (stealthy underwater vehicle) and a defender (group of search units with active sonars) moving in a discretized chart, with momentary detection probabilities that may depend on any aspect of the current game state, such as distance to target, speed of target, bearing of target, etc. (Fig. 1). We solve the game, that is, generate approximately unexploitable stochastic strategies, following an approach from superhuman poker AI employing public belief states and self-play [8], [9], [10], [11]. More specifically, we transform the imperfect-information ASW game into a perfect-information game [12] that can be solved using efficient forms of self-play reinforcement learning. We provide evidence that the proposed solution method is sound, showing empirically that the algorithm indeed closely approximates the game-theoretic solution in a restricted scenario that is small enough to be solvable by tabular method from algorithmic game theory, e.g. counterfactual regret minimization [13].

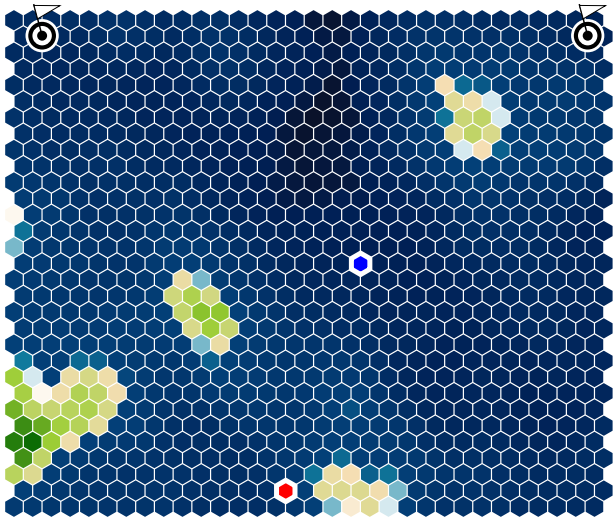


Fig. 1. Running example. A surface vessel (marked in red) pauses in mid-water, an indication it might be launching an unmanned underwater vehicle. There are two critical infrastructure assets (marked as flags) that need to be protected from underwater threats. A patrol vessel with a dipping sonar is dispatched to the area, arriving after 1 hour to the position (marked in blue). How should the patrol vessel search?

The organization of this paper is as follows. Section II reviews related work, providing some background on patrolling and game-theoretic approaches to planning thereof. Section III presents the ASW mobile sensor planning problem considered in this paper, and Section IV formalizes the planning problem as a strategy game with fog-of-war. Section V reviews a theoretical framework for solving poker-like games, and Section VI applies and specializes the theoretical framework to the ASW strategy game introduced. Section VII presents a prototype im-

[1]https://www.navylookout.com/protecting-maritime-infrastructure-from-attack-new-technologies-and-tactics/.

1946-0759/25/$31.00 ©2025 IEEE
DOI 10.1109/ICMLA66185.2025.00185

plementation of the proposed solution method, demonstrating the prototype in a classical ASW scenario. Section VIII reports the experimental results, verifying convergence toward game-theoretic equilibria. Finally, Section IX concludes the paper by summarizing contributions and discussing implications for future applications.

II. Background

Patrolling is a common task in defense and security: submarines patrol the seas, patrol vessels patrol marine and coastal areas, foot patrols patrol designated land areas, cyber soldiers patrol cyberspace, etc. The task of patrolling is to "operate with small patrols over large areas to early detect and counteract security-threatening activities."[2] To maximize the deterrent effect, patrols must avoid being predictable in a way that can be exploited by adversaries that observe and study the patrolling before launching an intrusion mission.

For more than a decade, unpredictable randomized patrols have been planned within various arenas using decision support systems based on algorithmic game theory. The systems generate optimally randomized patrols, i.e., patrols that cannot be exploited even by an intruder that knows the very randomization method being used to generate the patrols. The PROTECT decision support system, which has been operational within the US Coast Guard for more than a decade, is a celebrated example [14], listed in 2021 as one of the most successful applications of AI within defense and security [15]. A newer example is PAWS, which is used to plan foot patrols in nature reserves in, among other places, Uganda, and is now on its way to being deployed in over 400 locations [16]. Similar game-theoretic decision support systems have also been used successfully to plan unpredictable patrols in other domains, including airports, air traffic, public transport, and IT networks [17].

There have also been some attempts at transferring the above algorithmic game theory to ASW with passive (silent) sonars (cf. Hew and Yiap [18]). However, in modern ASW, active sonars (that emit sound) play an essential role, as only active sonars are able to detect modern stealthy underwater threats in real-time.[3] In general, the sound emitted by an active sonar is strong enough to reveal the position of the sonar even to very distant submarines, typically long before the active sonar itself detects the submarine. In effect, this means that in the game of hide-and-seek between a submarine and a search group, the submarine continuously receives new information throughout the game as to the whereabouts of search units, and can adjust its movement accordingly. Unfortunately, the algorithmic game theory used so successfully in PROTECT, PAWS, etc. cannot handle this kind of real-time information [19], making it somewhat challenging to apply these same methods to modern ASW.

[2]https://www.forsvarsmakten.se/sv/organisation/livgardet/insatsforband/13e-sakerhetsbataljonen.

[3]https://en.wikipedia.org/wiki/Diver_detection_sonar.

Decision support for ASW in operational use today, such as the Acoustic Mission Planner[4] and the Operational Route Planner,[5] generate search routes through one-sided optimization algorithms [20], [21], i.e., by optimizing search routes against a manually crafted reactive (behavior) model for the enemy submarine. Unfortunately, a one-sided optimization may yield search patterns that can be exploited by adversaries that observe and study how the search is being conducted over time.

In this paper, we transfer recent game-theoretically sound reinforcement learning, originally developed for superhuman poker-bots [8], [9], [10], to ASW—specifically, to route planning for mobile active sonars. Our goal is to open up the possibility of unpredictable decision support systems, similar to PROTECT and PAWS, in the arena of ASW.

III. Planning Problem

In this paper, we consider the problem of how to search for a stealthy underwater vehicle (UV) using mobile active sonars carried on surface vehicles (SVs). As an example, the search party could be a group of unmanned, remotely controlled SVs patrolling a body of water with scattered critical infrastructure in order to detect any small UV operating in the area.[6] Some of the SVs in the search party might pull a towed sonar, while other SVs might carry a dipping sonar, moving swiftly from one dipping point to the next (but with the dipping sonar ineffective while moving). The following micro-scenario, a variant of the classical Flaming datum problem [22], will serve as a running example.

Flaming datum scenario. *A surface vessel from a scientific expedition fleet with a somewhat shady track record is being tracked (by satellite, radar, etc.). The vessel suddenly stops in mid-water for no apparent reason, an indication it might be launching an unmanned UV, a capability the vessel is known to have. In order to prevent any surreptitiously launched UV to reach either of the two critical infrastructure assets located in the area, a patrol vessel, in the form of an unmanned SV with a dipping sonar, is dispatched to the site (see Fig. 1). How should the patrol vessel search the area?*

In the Flaming datum scenario above, the patrol vessel needs to be unpredictable, or else an adversary will soon learn (exploitable aspects of) how the vessel patrols. Indeed, provoking patrols in order to study them is a standard scouting procedure also in naval grey zone conflicts.

The assumption that the adversary knows the search pattern employed is often reasonable in practice. Before launch, the adversary can be assumed to study the search units over time (by means of satellite, passive sonar, radar, etc.) until the adversary has learned about (statistical) regularities in the search patterns employed, and then adapting the behavior of the UV accordingly.

[4]https://www.wagner.com/acoustic-mission-planner-amp-for-the-mh-60r/.

[5]https://www.wagner.com/operational-route-planner-orp/.

[6]https://www.navylookout.com/protecting-maritime-infrastructure-from-attack-new-technologies-and-tactics/.

The assumption that the search party only employs active sonars (that emit sound) is, perhaps, less generally applicable. However, current intruder detection systems use only active sonars as passive sonars have a limited ability to detect small signature UV in real-time.[6]

IV. Game Formulation

We formulate the ASW planning problem from Section III as a fog-of-war strategy game played on a discretized nautical chart between an intruding stealthy UV and a defending search team with mobile active sonars.

A. Simplifying Assumptions

The modeling is based on the assumption that the intruding UV knows the current position of any active sonar in the water throughout its mission. This is a reasonable simplification when planning search with active sonars, especially for worst-case analysis, since active sonars emit a powerful sound that may be heard far away. The model assumes, moreover, that the intruding UV knows whether a sonar dip has been successful or not. Again, this is a reasonable simplifying assumption when planning the use of active sonars. Finally, the modeling assumes that sonars return either a hit or a miss, while in practice sonars output probabilities, and an algorithm or operator interprets the output as a hit if the probability reaches a certain threshold. In a typical, low-intensity conflict, however, the defending organization can focus search resources as soon as there is a weak indication of an intruder.

B. Formalization

We formalize the planning problem as a two-player, zero-sum, and turn-based game of imperfect information: player 1 controls an UV and player 2 controls one or multiple synchronized SVs, each vehicle carrying either a towed sonar or a dipping sonar. For a given planning problem, the nautical chart yields the state space of the game, vehicle capabilities yield the action space, and the mission objective yields the end states and a utility function as follows.

State space. The game is played on a hexagonal board M represented in axial coordinates (q, r) for some $q \in \{0, \dots, Q-1\}, r \in \{0, \dots, R-1\}$, where Q, R are positive integers, as illustrated in Fig. 2. (The playing board may also include a vertical dimension (depth), omitted here for ease of presentation). The axial coordinate system limits the board to have the shape of a parallelogram, but hexagons can be masked out to create a search area with a desired shape. (The axial coordinate system makes it easier to vectorize the computations, allowing the simulation to run smoother on accelerated hardware, e.g., GPUs/TPUs). Each position in the grid represents either water (with acoustic attributes such as reverberation level) or land (Fig. 1).

Initial states. When the game starts, the UV is located at a position (unknown to player 2) drawn from a prior distribution over the hexagonal grid. Similarly, the SVs are given initial positions from a prior distribution.

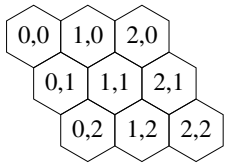


Fig. 2. Illustration of the axial hexagonal coordinate system in a 3 × 3 grid. Each hexagon is denoted by its axial coordinate (q, r). Here, $Q = R = 3$.

Action space. Player 1 moves the UV and player 2 moves the SVs. Each type of vehicle has its own timer for when it makes a move, reflecting differences in speed. The UV moves to an adjacent hexagon after every $t_{\text{UV}} > 0$ minutes (or rests in the same hexagon). Similarly, an SV with a towed sonar moves to an adjacent hexagon every $t_{\text{SV}}^{\text{tow}} > 0$ minutes. An SV with a dipping sonar can move to any hexagon within a given radius (and drop its sonar and complete a search) every $t_{\text{SV}}^{\text{dip}} > 0$ minutes.

Observations. Player 1 (UV) observes its own position as well as the position of each search vessel (as motivated above), while player 2 (the search team) observes only the positions of search vessels.

End states. The game terminates when the search units detect the UV or if a scenario-specific condition is met, e.g., the UV reaches a target or when we have reached a time limit T. The detection probability may depend on the distance to the UV, the angle of incidence, whether the UV is at rest or not, the bottom topography, depth, etc., according to sonar equations whose parameters vary across the hexagonal grid. The sonar surplus is converted to an instantaneous detection probability according to standard naval operational analysis [23]. In the example in Section VII, the detection probability depends only on the distance to the UV (Fig. 3).

Utility. The reward function (utility) captures the intended mission goal and as such may vary with the scenario under consideration. In the examples of this paper, the reward for player 1 is 1 if the UV reaches any of its goal destinations, -1 if the UV is detected, and 0 otherwise.

V. Theory

The method proposed below for solving the ASW strategy game introduced in Section IV follows a recent theoretical framework for game-theoretically sound deep reinforcement learning in poker-like games [12]. With this framework, a poker-like game is transformed into a game of perfect information, but in a more complex and continuous space. The same kind of elimination of hidden information has been applied in connection with several historical milestones in poker-AI, but without general mathematical theory [8], [9], [10].

Sokota et al. [12] explain that games with imperfect information are harder due to two main causes: (1) the backward dependence problem, and (2) the non-correspondence problem. The *backward dependence problem* is that the expected outcome at a given point in time during a game depends on what policies were applied in the past at earlier decision points

during the game, thereby ruling out solution methods based on backward induction. Nayyar, Mahajan, and Teneketzis [24] resolved this by having players publicly announce their policies during play, which transforms the imperfect-information game into a game of perfect information, more specifically, to a *Public Belief Alternating Markov Game* (PuB-AMG), referred to hereafter as the *public belief game*.

However, the Nash equilibrium [25], [26] solution to the public belief game is not necessarily equivalent to Nash equilibrium in the original game—this is the *non-correspondence problem*. For two-player zero-sum games, [12, Prop. 5.13] shows that, by adding a certain regularization, the solution can be made arbitrarily close to a Nash equilibrium of the original game. To be precise, they use the following MiniMaxKL objective:

$$\mathfrak{R}\colon (h,a,\pi) \mapsto \begin{cases} \mathcal{R}(h,a) + \alpha_{\mathbf{KL}}\mathbf{KL}(\pi,\rho(h)), & \iota = 1, \\ \mathcal{R}(h,a) - \alpha_{\mathbf{KL}}\mathbf{KL}(\pi,\rho(h)), & \iota = 2. \end{cases} \tag{1}$$

Here, $\mathcal{R}(h,a)$ is the reward for choosing action a, given the history h of all public and private actions and outcomes, ι is the player index, and π is the policy used to choose/sample the action. $\mathbf{KL}$ is the Kullback–Leibler divergence between the current policy π and some regularization policy ρ with at least $\epsilon > 0$ probability on each legal action. $\alpha_{\mathbf{KL}}$ is the regularization factor used to control the magnitude of the regularization on the objective. The choice of the regularization policy ρ is described in Section VI-B.

VI. Solution Method

Applying the framework from Section V, we compute approximately game-theoretic optimal strategies for the ASW game introduced in Section IV by converting the game into a fully observable public belief game and training agents via self-play on public states with regularized objectives (Eq. 1).

A. Transformation Into a Perfect-Information Game

The state space, initial state, action space, transitions, and utility function for the public-belief game are produced as follows.

State space. A (public belief) state at time t in the public-belief game, denoted β_t, consists of a public belief distribution μ_t over the hexagonal grid M (representing uncertainty about the location of the UV) together with public information about player 2's units.

Initial states. The public-belief game starts with a prior public belief distribution μ_0 inherited from the underlying ASW game.

Actions. Players act in a public belief game by announcing a policy (from the underlying game). In particular, player 1 announces a mapping Π_1 from board positions (q,r) (representing hidden information) to distributions over actions. Since player 2 has no private information in the underlying ASW game, announcing a policy for player 2 reduces to simply announcing a particular action (in the underlying game). Henceforth, we will refer to an action in the public game (i.e., an announced policy Π_1 of player 1 or an observable action a_2 of player 2) as a *public action*, written a^{pub}.

Transitions. When player 1 announces a policy, the public belief distribution is updated analytically to match the transitions. When player 2 takes an action, the public belief distribution is updated to reflect the sonar-sensor's observation, and the public information is updated with the public movement of vehicles.

Utility. The reward (utility) in the public-belief game is given by the expected reward in the original game. For a policy Π_1 announced by player 1 at a public belief state β, the reward is computed as follows:

$$\begin{aligned} \widetilde{\mathcal{R}}(\beta,\Pi_1) &= \mathbb{E}_{(q,r)\sim\beta}\,\mathbb{E}_{a_1\sim\Pi_1(\cdot\mid(q,r))}[\mathcal{R}((q,r),a_1)] \\ &= \sum_{(q,r)\in M} \mu^{\beta}(q,r) \sum_{a_1\in\Pi_1(q,r)} \Pi_1(a_1 \mid q,r)\,\mathcal{R}((q,r),a), \end{aligned} \tag{2}$$

where μ^{β} is the public belief distribution in the public belief state β, and $\mathcal{R}((q,r),a)$ is the reward received from action a when the UV is at coordinates (q,r) (which, in turn, is equivalent to the reward $\mathcal{R}(h,a)$ from the underlying ASW game, where h is an execution history in which the UV is at position (q,r)). For an observable action a_2 of player 2, the reward is computed more simply as follows:

$$\begin{aligned} \widetilde{\mathcal{R}}(\beta,a_2) &= \mathbb{E}_{(q,r)\sim\beta}\left[\mathcal{R}((q,r),a_2)\right] \\ &= \sum_{(q,r)\in M} \mu^{\beta}(q,r)\,\mathcal{R}((q,r),a_2). \end{aligned} \tag{3}$$

In summary, the public reward $\widetilde{\mathcal{R}}(\beta,a^{\text{pub}})$ is $\widetilde{\mathcal{R}}(\beta,\Pi_1)$ if $a^{\text{pub}} = \Pi_1$ and $\widetilde{\mathcal{R}}(\beta,a_2)$ if $a^{\text{pub}} = a_2$.

B. Self-Play Learning Framework for the Transformed Game

We define a shared network $\mathcal{M}_\theta$ that maps a tensor observation $\mathcal{O}(\beta_t)$ of the public state to (i) a policy for player 2, $\pi_2(\cdot \mid \mathcal{O}(\beta_t))$, (ii) a decision rule for player 1, $\Pi_1(\cdot \mid \mathcal{O}(\beta_t))$, and (iii) a value *map* $\overline{V}_\theta(\beta_t)[q,r]$ estimating the expected future return for the UV at (q,r). The public estimated value is $V_\theta(\beta_t) = \sum_{q,r} \mu_t(q,r)\,\overline{V}_\theta(\beta_t)[q,r]$.

We implement self-play by first sampling N independent public trajectories $\{T^{(i)}_{\text{pub}}\}_{i=1}^{N}$. Player 2's information only depends on the public belief states, and not on any sampled trajectory of the UV. Then for each public trajectory $T^{(i)}_{\text{pub}}$ we sample n private UV trajectories $\{T^{(i,j)}_{\text{UV}}\}_{j=1}^{n}$ by drawing from the announced decision rule $\Pi_1(\cdot \mid \beta^{(i)}_t)$ that coincide with the public actions in $T^{(i)}_{\text{pub}}$. Each public trajectory updates μ_t (and hence β_t) using the public actions and the sonar-sensor model; each UV sub-trajectory samples a concrete hidden path and yields corresponding rewards. We then update: (i) player 2's policy from public action sequences $\{a_t^{(i),\,pub}\}$, $\{a^{(i)}_{2,t}\}$ with public rewards $\{\widetilde{\mathcal{R}}(\beta^{(i)}_t, a^{(i)}_{2,t})\}$, and (ii) player 1's decision rule Π_1 from sampled UV actions $\{a^{(i,j)}_{1,t}\}$ with underlying rewards $\{\mathcal{R}(h^{(i,j)}_t, a^{(i,j)}_{1,t})\}$.

We employ a per decision **KL** regularizer (described in Section V) using a slowly updated a regularization policy

$\pi_{\text{reg}} = \pi_{\text{reg}}(\mathcal{O}(\beta_t))$. We use a searcher and target model pair $(\pi_{\text{search}}, \pi_{\text{tgt}})$ with parameters $(\theta_{\text{search}}, \theta_{\text{tgt}})$ to stabilize training, and set the regularization policy π_{reg} to be prior checkpoints of the target model, similar to [27]. We let the searcher generate trajectories and train its parameters θ_{search} using PPO based updates with regularized objective and policy. The target model updates its parameters θ_{tgt} every training step linearly/proportionally towards the searcher model θ_{search}

$$\theta_{\text{tgt}} = (1 - \alpha_{\text{reg}}) * \theta_{\text{tgt}} + \alpha_{\text{reg}} * \theta_{\text{search}}, \tag{4}$$

where α_{reg} is a small step-size. After every Δm updates, we snapshot the target policy $\pi_{\text{tgt},m}$, where m is a new checkpoint index, and define a linearly interpolated *regularization policy* between the old and new target checkpoint:

$$\pi_{\text{reg}} \;=\; (1-\lambda) * \pi_{\text{tgt},\,m-1} \;+\; \lambda * \pi_{\text{tgt},\,m}, \tag{5}$$

for the interpolation scalar $\lambda \in [0,1]$ that increases within the Δm update window. Let $\widetilde{G}_t^{(i)} = \sum_{s=t}^{T} \widetilde{\mathcal{R}}(\beta_s^{(i)}, a_s^{(i),pub})$ and $G_t^{(i,j)} = \sum_{s=t}^{T} \mathcal{R}(\beta_s^{(i,j),}, a_s^{(i,j),pub})$ be the public and private sampled value of a state. Denote by G_t one of $\widetilde{G}_t^{(i)}$ and $G_t^{(i,j)}$. Then the *locally* regularized sampled value (used for advantage computation in the PPO algorithm) is

$$G_t^{\text{reg}} := G_t \;+\; \alpha_{\mathbf{KL}}\, \mathbf{KL}\big(\pi_{\text{search}}(\cdot \mid \beta_t), \pi_{\text{reg}}(\cdot \mid \beta_t)\big). \tag{6}$$

We use this local form rather than summing the **KL** at every future step in order to avoid double-counting the same regularizer along a trajectory and to match per-decision reward transformations. In practice, we also add the regularization term of Eq. 6 to the loss function (Eq. 8) in order to directly regularize the policy.

We train θ_{search} by optimizing each decision independently using PPO. In detail, the policy ratio is:

$$\rho_t(\theta) = \pi_\theta(a_t \mid s_t) / \pi_{\theta^k}(a_t \mid s_t), \tag{7}$$

where π_{θ^k} is the policy used to sample a_t in the trajectory trajectory. We define the estimated advantage as $\widehat{A}_t := G_t^{\text{reg}} - V_t$, where V_t denotes either the value map $\overline{V}_\theta(\beta_t)[q,r]$ or $V_\theta(\beta_t)$ depending on G_t. We update the policies by maximizing the clipped likelihood ratio

$$L^{\text{CLIP}}(\theta) = \min\Big(\rho_t(\theta)\,\widehat{A}_t,\; \text{clip}\big(\rho_t(\theta),\, 1-\epsilon,\, 1+\epsilon\big)\,\widehat{A}_t\Big), \tag{8}$$

and we update the value function (value map $\overline{V}_\theta(\beta)[q,r]^{(i,j)}$) toward G_t^{reg} with Huber-loss.

VII. Implementation

The solution method proposed in Section VI has been implemented in a simple prototype decision support that generates unpredictable search paths and estimates the enemy position based on incoming sonar reports. The prototype is written entirely in JAX [28], including the ASW strategy game from Section IV, which provides a high-throughput simulation environment tailored for parallelized self-play on GPU hardware.

The prototype is illustrated below with the Flaming datum scenario from Section III. Variants of the Flaming datum problem have previously been analyzed with game-theory [3], [4], [5], [6], [7], but under the assumption that the submarine cannot hear (and react to) the position and movement of sonars during the search (even when sonars are active). Moreover, sonars are abstracted as 'cookie-cutters', i.e., a submarine is detected if (and only if) it is within a given, fixed range of a sonar.

A. Setup and Training

The nautical chart, vehicle capabilities, and mission objective are set as follows for the Flaming datum scenario.

Chart. The grid in axial coordinates is 47 × 31 hexagons, where each hexagon is 800 meters wide. Sides are masked to create a square grid of about 25 × 25 kilometers with ocean and land hexagons (Fig. 1).

Capabilities. The UV travels 7 kilometers per hour, i.e., t_{UV} is roughly 7 minutes. The SV can move to a new dipping point up to 4 kilometers away from its current position every 15 minutes, i.e., $t_{\text{SV}}^{\text{dip}}$ is 15 minutes and the SVs movement radius is 5 hexagons (4 kilometers). Detection probabilities depend only on the distance to the UV (Fig. 3). The capabilities, in particular the movement of the SV and the detection probabilities, are chosen with a view of making the scenario an intuitively comprehensible test case for validating generated search strategies by informal inspection.

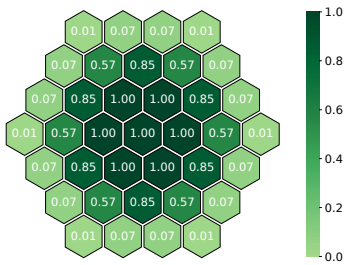
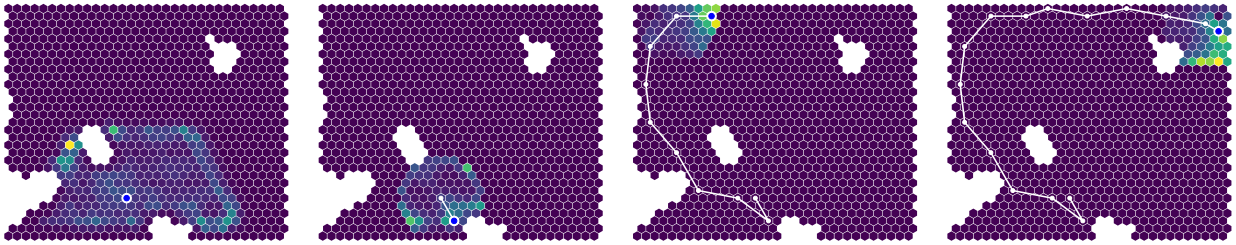


Fig. 3. Probabilities of detecting a UV around the sonar in the Flaming datum scenario. In this scenario, the probabilities only depend on the distance from the sonar to the UV. However, the implementation allows for detection probabilities that depend on any aspect of the current state (such as impact angle, speed, etc.), as described in Section IV.

Mission objective. To reflect the mission objective of the search party (preventing a suspected UV of reaching either of the two infrastructure assets), the game terminates with a win for the UV (reward +1) if the UV reaches either of the two flags, terminates with a loss for the UV (reward -1) if the UV is detected, and terminates in a draw (reward 0) otherwise. The time limit T is set to 8 hours (which entails 57 sequential decision points for the UV and 21 for the SV in each run of the game).

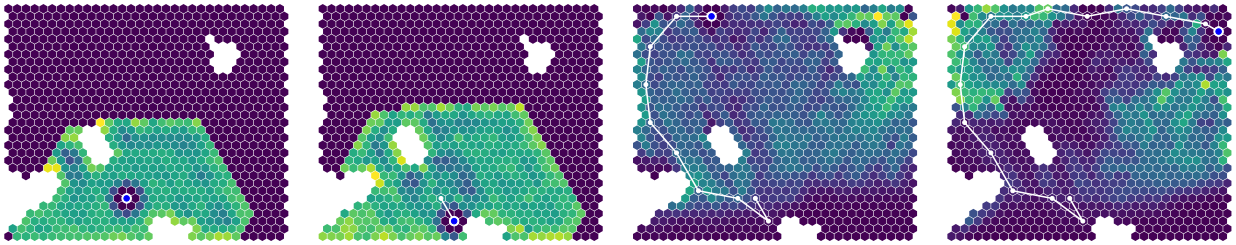
We train with a regularized objective on the public game, storing checkpoints of θ_{search} and θ_{tgt} along the way.[7] Our train-

[7]To speed up convergence, we add a small term to the reward for the UV proportional to the negative belief of the UV being in its current position ($\propto -\mu_t[q,r]^{(i,j)}$). For stability, μ_t is normalized, which yields the transformed reward: $\mathcal{R}_t^{(i,j)} = \mathcal{R}_t^{(i,j)} - c \times \big(\mu_t[q,r]^{(i,j)} - \mathbb{E}_{\mu_t}[\mu_t]\big) / \text{Std}_{\mu_t}(\mu_t)$, where $c = 0.008$.



(a) First dipping point (b) Second dipping point (c) Tenth dipping point (d) Sixteenth dipping point

Fig. 4. Successive distributions for sampling some early dipping points during a possible run of the Flaming datum scenario in Fig. 1. (a)–(d) Distribution for sampling the first, second, tenth, and sixteenth dipping points respectively. The distributions are visualized using the viridis color map, scaled to the minimum and maximum heatmap values (brighter green/yellow indicates higher probability to dip there). The dipping points sampled during this particular run are marked in blue with white outline. The skerries (rocky islands) can be seen implicitly as masked out hexagons.



(a) First dipping point (b) Second dipping point (c) Tenth dipping point (d) Sixteenth dipping point

Fig. 5. Probability distribution of the submarine's location based on successive (negative) sonar reports during a possible run of the Flaming datum scenario. (a)–(d) Distribution after completing the first, second, tenth, and sixteenth dipping respectively. These distributions are visualized with the viridis color map and scaled to the respective minimum and maximum probabilities.

ing parameters are $\alpha_{\text{tgt}} = 0.001$, $\alpha_{\mathbf{KL}} = 0.04$, $\Delta m = 1024$, $N = 64$, and $n = 2048$. We train for 21 hours (320 000 training steps) on an RTX 4090 graphics card and observe converging game metrics after 8 hours (Fig. 6).

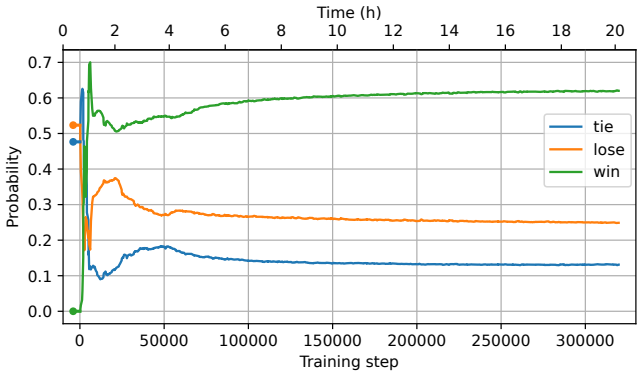


Fig. 6. Probability of winning, losing and playing a draw over training steps in the Flaming datum scenario. *win*: Probability of winning for player 1, i.e., probability that the UV reaches an asset. *lose*: Probability of losing for player 1, i.e., probability that the UV is detected. *tie*: Probability of a tie, i.e., probability that the UV is neither detected nor reaches an asset. Initial values for 'win', 'tie' and 'lose' with the untrained model are marked before the first training step.

B. Search Pattern and Estimated Enemy Position

Given the nautical chart, vehicle capabilities, and mission objective, the prototype generates a search path by sampling from the (approximated) Nash equilibrium. As an illustration, Fig. 4 shows the computed (approximate) Nash distributions for sampling dipping points during earlier parts of the 8 hour long search in the Flaming datum scenario. The distributions seem intuitively reasonable given the simplified detection model (Fig. 3). The first dipping point is sampled roughly uniformly within the Area of Uncertainty (AoU), with a marked emphasis on the outer perimeter. The second dipping point is sampled with a clear preference for travel distance and some preference for southward direction. The tenth dipping point is sampled between moving to the southwest (i.e, suddenly reversing direction) and moving to the east (i.e., continuing vaguely in the direction of the opposite infrastructure asset), with an emphasis on the latter. The sixteenth dipping point is sampled with a tendency towards the south.

During the search, the prototype updates a common tactical picture in the form of a probability distribution over the enemy submarine's possible positions (μ_t), visualized as a heat map. The probability distribution is computed analytically given the trained strategy for the UV (Π_1) and the history of search positions a_2 so far. To illustrate, Fig. 5 shows the heat map during the earlier parts of the search in the Flaming datum scenario. Again, the distributions seem intuitively reasonable. After the first dipping point, the distribution is roughly uniform

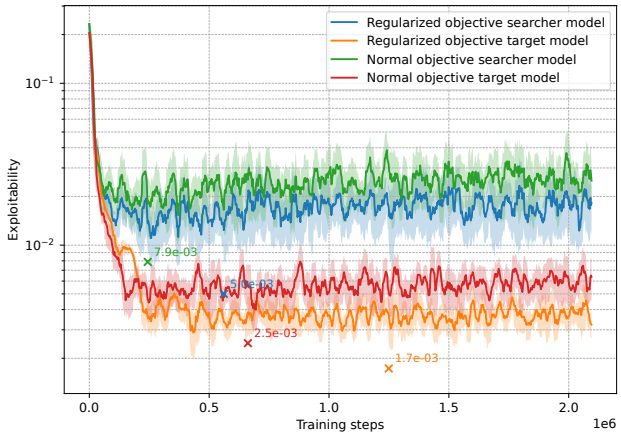


Fig. 7. Exploitability in the solvable ASW scenario (Fig. 8) when training on the corresponding public game, with and without regularized objective ($\alpha_{\mathbf{KL}} = 0.1$ and $\alpha_{\mathbf{KL}} = 0$ respectively). The exploitability for the searcher- and target model (θ_{search} and θ_{tgt} respectively) is shown for each objective. Training was conducted over 2097152 training steps, with 1024 checkpoints used for computing tabular exploitability. For the graphs, we use a sliding window averaging and standard deviation with a window-size of 10 datapoints (20480 steps).

within the AoU, with an emphasis on the perimeter, i.e., an emphasis on the UV having moved (quietly) as far as possible from its mothership. After the second dipping point, the distribution is still roughly uniform within the (expanded) AoU, but now with a ring of reduced probability at the center, eminating from the first dipping point. A bit later, after the tenth and sixteenth dipping point respectively, the distribution has become considerably less uniform, with a clear emphasis around the infrastructure asset at the opposite end of where the SV is presently.

The probability distribution over the enemy submarine's possible positions is a form of worst-case analysis; the submarine, it is assumed, tries to counteract the mission objective for the search party as effectively as possible, i.e., the submarine follows the *most dangerous enemy course of action* [29]. In general, this may entail randomizing its reactions to the active sonars so as to increase the uncertainty for the search units, e.g., by moving with some probability towards a recent dipping point, 'reestablishing' probability mass there (as in Fig. 5).

VIII. Nash Convergence Experiment

We verify that the solution method proposed in Section VI, and as implemented in Section VII, converges to the game-theoretic equilibrium. Since it is intractable to verify the solution method on larger maps such as in Section VII, we verify the solution method for a small scenario that can be solved with tabular methods.

Solvable scenario. *We consider a solvable, small scenario in the ASW game (see Fig. 8). The board consists of only 12 uniform water hexagons, and the UV is restricted to moving either southeast or southwest at each time step. The UV starts in hexagon* $(2, 0)$ *or* $(3, 0)$ *with uniform probability, and the game terminates if the UV passes the last row of hexagons or if it is detected. There is only one search unit.*

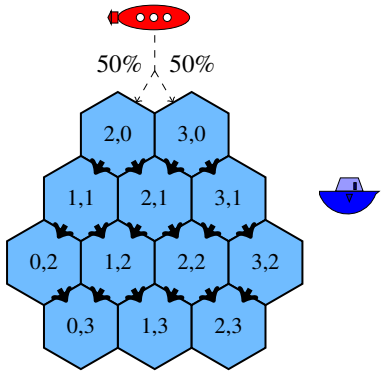


Fig. 8. Solvable ASW scenario. The UV (red) is restricted to moving either southeast or southwest at each time step. The SV (blue) dips its sonar sensor in one of the hexagons along the row of the UV.

To measure exploitability in the solvable scenario, we first train models with, respectively without, a regularized objective on the public game, storing checkpoints of θ_{search} and θ_{tgt} respectively along the way. Our training parameters are $\alpha_{\text{tgt}} = 0.001$, $\alpha_{\mathbf{KL}} = 0.1$, $\Delta m = 2048$, $N = 32$, and $n = 16$. The training is conducted for 2,097,152 steps (32,768 × 64), during which 1,028 checkpoints are saved. For each stored checkpoint, we then construct a tabular policy in the original imperfect information game (which will later be used to compute exploitability). To construct a tabular policy, we replay the histories leading to each information state I in the original game, which yields the corresponding public belief state β consistent with I. Given β and the private information contained in I, we obtain the policy $\pi(I, \beta)$, which we assign as the policy for information state I in the table for the tabular policy. Finally, we compute the tabular exploitability from an equivalent game implemented in open-spiel [30].

The resulting exploitabilities for all checkpoints during the training are shown in Fig. 7, for the target model as well as the searcher model. As can be seen, the models converge to a low exploitability, with the target model θ_{tgt} converging to a lower exploitability than the searcher model θ_{search} in both cases. Moreover, the model converges to a slightly smaller exploitability when using a regularized objective. Surprisingly, perhaps, the model still converges to a very low exploitability when using an unregularized objective (with an entropy term), somewhat contrary to what might be expected from [12]. A possible explanation is that the policy-entropy regularization loss used in PPO (and other RL algorithms) shares similar effect on training updates as a regularized objective with uniform policy, especially since we perform a hyperparameter search for the entropy coefficient to minimize the exploitability.

IX. Conclusions

In this paper, we considered the problem of how to search for a stealthy underwater vehicle in an unpredictable way with mobile active sonars, a fundamental planning problem in

modern naval warfare. We formalized the planning problem as a fog-of-war strategy game played on a (discretized) nautical chart, with momentary detection probabilities that may depend on any aspect of the current game state (such as distance to target, speed of target, impact angle on target, or local seabed topography). We solved the game using public belief states and self-play following a method from celebrated milestones in poker-AI. We presented a parallelized implementation in JAX, and illustrated the implementation with an instance (Fig. 1) of a classic ASW search problem—beyond the reach of earlier game-theoretic solution methods in the operational research literature—and showed that the solution method converges (Fig. 6) producing plausible strategies for search vehicles (Fig. 4) and the intruding underwater vehicle (Fig. 5). Finally, we provided evidence (Fig. 7) that the proposed solution method is game-theoretically sound, showing empirically that the implementation closely approximates Nash in a restricted scenario that is small enough to be solvable by tabular methods.

To the best of our knowledge, the work reported here represents the first publicly reported application of superhuman poker AI to a real-world decision problem.

Looking ahead, it is hoped that the proposed approach may help open up the possibility of unpredictable decision support for mobile sensor planning in ASW similar to the unpredictable decision support being used today for planning patrolling in other arenas (such as PROTECT, PAWS, etc.).

Code Availability

The program code along with more details are available at https://github.com/ChrisLimer/ASW-planning.git.

Acknowledgment

The work reported here would not have been possible without the ASW expertise at the Swedish Defence Research Agency. The authors gratefully acknowledge the assistance from Per Brämming, Martin Garmelius, Peter Krylstedt, Ron Lennartsson, Manne Miltion, Erik Ovegård, Johan Pelo, and Stefan Petrovic Wångerud in providing critical domain knowledge and helpful suggestions.

References

[1] W. P. Hughes, "Naval warfare," in *Encyclopedia Britannica*. Encyclopaedia Britannica, Inc, 2025. [Online]. Available: https://www.britannica.com/topic/naval-warfare

[2] N. Brown and T. Sandholm, "Superhuman ai for multiplayer poker," *Science*, vol. 365, no. 6456, pp. 885–890, 2019.

[3] A. Washburn *et al.*, *Two-person zero-sum games*. Springer, 2014.

[4] R. B. Yoash, M. P. Atkinson, and M. Kress, "Where to dip? search pattern for an antisubmarine helicopter using a dipping sensor," *Military Operations Research*, vol. 23, no. 2, pp. 19–40, 2018.

[5] A. Washburn and R. Hohzaki, "The diesel submarine flaming datum problem," *Military Operations Research*, pp. 19–30, 2001.

[6] A. A. Soto, "The flaming datum problem with varying speed," Ph.D. dissertation, Monterey, California. Naval Postgraduate School, 2000.

[7] J. M. Danskin, "A helicopter versus submarine search game," *Operations Research*, vol. 16, no. 3, pp. 509–517, 1968.

[8] M. Moravčík et al., "DeepStack: Expert-level artificial intelligence in heads-up no-limit poker," *Science*, vol. 356, no. 6337, pp. 508–513, 2017, doi: 10.1126/science.aam6960.

[9] N. Brown, A. Bakhtin, A. Lerer, and Q. Gong, "Combining deep reinforcement learning and search for imperfect-information games," in *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS 2020)*. San Diego, CA: NeurIPS, 2020, pp. 17 057–17 069.

[10] M. Schmid et al., "Student of Games: A unified learning algorithm for both perfect and imperfect information games," *Science Advances*, vol. 9, no. 46, pp. 1–14, 2023, Art. no. eadg3256, doi: 10.1126/sciadv.adg3256.

[11] M. Oscarsson, J. Brynielsson, M. Cohen, F. Kamrani, and C. Limér, "The cost of uncertainty in self-play reinforcement learning and search," in *Proceedings of the 2025 IEEE International Conference on Intelligence and Security Informatics (ISI 2025)*. Piscataway, New Jersey: IEEE, 2025.

[12] S. Sokota, R. D'Orazio, C. K. Ling, D. J. Wu, J. Z. Kolter, and N. Brown, "Abstracting imperfect information away from two-player zero-sum games," in *Proceedings of the 40th International Conference on Machine Learning (ICML 2023)*, vol. 202. PMLR, 2023, pp. 32 169–32 193. [Online]. Available: https://proceedings.mlr.press/v202/sokota23a.html

[13] M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione, "Regret minimization in games with incomplete information," in *Proceedings of the 20th Conference on Neural Information Processing Systems (NIPS 2007)*. San Diego, CA: NeurIPS, 2007, pp. 1729–1736.

[14] B. An et al., "PROTECT: A deployed game-theoretic system for strategic security allocation for the United States Coast Guard," *AI Magazine*, vol. 33, no. 4, pp. 96–110, 2012, doi: 10.1609/aimag.v33i4.2401.

[15] M. Walsh et al., "Exploring the feasibility and utility of machine learning-assisted command and control: Volume 1, findings and recommendations," RAND Corporation, Santa Monica, CA, Research Report RR-A263-1, 2021, doi: 10.7249/RR-A263-1.

[16] F. Fang et al., "PAWS: A deployed game-theoretic application to combat poaching," *AI Magazine*, vol. 38, no. 1, pp. 23–36, 2017, doi: 10.1609/aimag.v38i1.2710.

[17] A. Sinha, F. Fang, B. An, C. Kiekintveld, and M. Tambe, "Stackelberg security games: Looking beyond a decade of success," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI 2018)*. International Joint Conferences on Artificial Intelligence, 2018, pp. 5494–5501, doi: 10.24963/ijcai.2018/775.

[18] P. Hew and N. Yiap, "Optimally randomized patrolling of chokepoints for theatre antisubmarine warfare," *Military Operations Research*, vol. 23, no. 1, pp. 49–56, 2018.

[19] Y. Wang et al., "Deep reinforcement learning for green security games with real-time information," in *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI 2019)*, vol. 33, no. 1. Palo Alto, CA: AAAI Press, 2019, pp. 1401–1408, doi: 10.1609/aaai.v33i01.33011401.

[20] S. S. Brown, "Optimal search for a moving target in discrete time and space," *Operations research*, vol. 28, no. 6, pp. 1275–1289, 1980.

[21] D. P. Kierstead and D. R. DelBalzo, "A genetic algorithm applied to planning search paths in complicated environments," *Military Operations Research*, pp. 45–59, 2003.

[22] L. D. Stone, "Search and screening: General principles with historical applications (bo koopman)," *SIAM Review*, vol. 23, no. 4, pp. 533–539, 1981.

[23] D. H. Wagner, W. C. Mylander, and T. J. Sanders, Eds., *Naval Operations Analysis*, 3rd ed. Annapolis, MD: Naval Institute Press, 1999.

[24] A. Nayyar, A. Mahajan, and D. Teneketzis, "Decentralized stochastic control with partial history sharing: A common information approach," *IEEE Transactions on Automatic Control*, vol. 58, no. 7, pp. 1644–1658, 2013.

[25] J. F. Nash Jr, "Equilibrium points in n-person games," *Proceedings of the national academy of sciences*, vol. 36, no. 1, pp. 48–49, 1950.

[26] J. F. Nash, "Non-cooperative games," in *The Foundations of Price Theory Vol 4*. Routledge, 2024, pp. 329–340.

[27] J. Perolat et al., "Mastering the game of Stratego with model-free multiagent reinforcement learning," *Science*, vol. 378, no. 6623, pp. 990–996, 2022, doi: 10.1126/science.add4679.

[28] J. Bradbury et al., "JAX: composable transformations of Python+NumPy programs," 2018. [Online]. Available: http://github.com/jax-ml/jax

[29] *Navy planning NWP 5-01*. Navy warfare publication, 2013.

[30] M. Lanctot et al., "OpenSpiel: A framework for reinforcement learning in games," *CoRR*, vol. abs/1908.09453, 2019. [Online]. Available: http://arxiv.org/abs/1908.09453