Solution Specification

Angad Ahuja

February 3, 2026

1 Solution-Space Specification and Methodological Rationale

1.1 Overview

The problem formulation fixes the latent layout, the observation channel (hit/miss/sunk), and the objective (minimize shots-to-win). Any additional decisions about reward signals, policy classes, baselines, training loops, and evaluation protocols are *methodological* choices: they specify how we operationalize learning and how we assess learned behavior, but they are not required to define the game itself. This section records those choices explicitly and justifies them in terms of (i) sample efficiency, (ii) stability of optimization under partial observability, and (iii) interpretability and comparability of results.

1.2 Reward Design

Method choice. We distinguish between the *task objective* (minimize expected shots-to-win) and the *training reward* supplied to an RL algorithm. Two reward families are considered:

$$\textbf{Sparse step penalty:} \quad r_t = -1 \text{ for } t < \tau, \qquad r_\tau = 0. \tag{1}$$

$$\textbf{Shaped reward:} \quad r_t = -1 + \alpha \cdot \mathbf{1}\{\text{hit at } t\} + \beta \cdot \mathbf{1}\{\text{sunk at } t\}, \qquad \beta > \alpha > 0. \tag{2}$$

The step-penalty reward in (1) is directly aligned with minimizing episode length: maximizing cumulative return is equivalent to minimizing the number of actions taken, up to discounting. This makes it a principled default when the evaluation metric is shots-to-win. The shaped reward in (2) provides denser feedback and can accelerate early learning by creating intermediate gradients before the terminal condition is reliably reached. However, it can introduce *objective distortion*: the agent may learn to optimize frequent hits rather than minimizing total shots, which is especially problematic if the shaping weights (α, β) are large relative to the step cost. In a partially observable search task, dense shaping can encourage "locally greedy" behavior (e.g., over-committing to exploitation after a hit) and can reduce exploration of information-rich actions. Therefore, shaping is treated as an explicit experimental factor rather than a default assumption.

Operational decision. We treat (1) as the *baseline reward* (maximally faithful to shots-to-win), and we treat (2) as a controlled ablation to test whether learning speed improvements come at a generalization cost.

1.3 Policy Class and Parameterisation

Method choice. Because the task is partially observable, a policy must, in principle, condition on the interaction history h_t. We consider two policy classes:

$$\textbf{History-dependent policy:} \quad \pi(a_t \mid h_t), \qquad h_t = (o_1, a_1, \ldots, o_{t-1}, a_{t-1}), \tag{3}$$

$$\textbf{Recurrent (finite-memory) policy:} \quad z_t = f_\theta(z_{t-1}, \phi(o_t, a_{t-1})), \qquad \pi_\theta(a_t \mid z_t, \text{mask}_t). \tag{4}$$

Here z_t is an internal state (memory), f_θ is a learnable update rule, ϕ is an observation-action encoder, and mask_t enforces infeasible actions.

The observation at time t reveals only the outcome of a single shot; optimal play depends on integrating evidence across many steps. A recurrent state z_t is a standard mechanism to represent sufficient statistics approximately when exact belief tracking is computationally expensive. The recurrent policy can be interpreted as learning an approximate belief updater: z_t aims to encode the posterior-relevant information contained in h_t without explicitly enumerating ship layouts. Direct policies on full histories are impractical. Recurrent parameterisations yield bounded computation per step and integrate naturally with modern RL training pipelines.

Operational decision. We implement a feedforward policy as the simplest baseline (state derived from the public board record), and a recurrent policy as the main method when empirical results indicate that memory improves generalization or reduces sample complexity. The recurrent design is therefore a *solution choice*, not a requirement of the problem definition.

1.4 Probability Heatmap Baseline as Non-RL Reference Policy

Method choice. We define the posterior marginal occupancy probability for each cell:

$$p_t(c) \;=\; \mathbb{P}(B(c) = 1 \mid h_t),$$

and consider the greedy baseline that selects:

$$a_t \in \arg\max_{c \in \mathcal{A}(s_t)} p_t(c).$$

Battleship is fundamentally a search-under-uncertainty problem. The heatmap baseline directly encodes the inference structure humans use (hunt/target) while remaining fully explainable as "shoot the most likely cell." This baseline provides a reference point for whether an RL policy is learning genuine inference-like behavior or merely exploiting training artifacts. If RL cannot match the heatmap baseline on shots-to-win under the same placement distribution, it indicates representation/training deficiencies rather than an intrinsic limitation of the task. The baseline can be near-optimal in practice under standard placement assumptions, but such a claim depends on (i) how p_t is computed (exact enumeration vs. sampling), (ii) whether tie-breaking incorporates targeting heuristics, and (iii) the ship set. Therefore we treat "near-optimal" as an empirical observation to be tested, not a theoretical guarantee.

Operational decision. We include the heatmap policy as a mandatory baseline for performance comparison and as an optional teacher for imitation warm-start, while explicitly reporting how p_t is approximated (enumeration, rejection sampling, or particle filtering).

1.5 Self-Play Training Procedure (Adversarial Coupled Learning)

Method choice. In the two-player setup, we require a training procedure to learn both attacker and defender policies. A standard approach is alternating optimization:

$$\pi_A \leftarrow \text{Train}(\pi_A \mid \pi_D \text{ fixed}), \qquad \pi_D \leftarrow \text{Train}(\pi_D \mid \pi_A \text{ fixed}),$$

repeating in cycles.

If both players update simultaneously, each player experiences a moving target distribution, which can destabilize learning. Alternating updates partially restores stationarity during each training block. Fixing one player while training the other approximates a best-response dynamic, which is the simplest conceptual route toward equilibrium behavior in zero-sum games. Alternating schedules are straightforward to implement with existing single-agent RL code: each block reduces to standard RL in a fixed environment distribution induced by the opponent.

Operational decision We treat alternating optimization as the baseline self-play protocol. More complex game-theoretic stabilizers (e.g., opponent pools, fictitious play variants, regularization toward mixtures) are optional extensions and are not required for an initial hobby-scale implementation.

1.6 Evaluation Protocol and Metrics

Method choice. We evaluate policies using metrics that quantify both *task performance* and *robustness/generalization*. Core metrics include:

$$\textbf{Shots-to-win:} \quad \mathbb{E}[\tau] \text{ under a specified placement distribution,} \tag{5}$$

$$\textbf{Generalization gap:} \quad \Delta_{\text{gen}} = \mathbb{E}[\tau]_{\text{adversarial}} - \mathbb{E}[\tau]_{\text{uniform}}, \tag{6}$$

and optional calibration diagnostics when the agent outputs (or can be probed for) per-cell hit probabilities:

$$\textbf{Calibration:} \quad \text{Calib}(p,\ y) \text{ computed over predicted probabilities and realized hits.}$$

Since the task goal is to win in the fewest shots, $\mathbb{E}[\tau]$ is the most direct measure of success. Reporting win-rate is typically uninformative because the attacker almost always wins eventually; what matters is efficiency. A policy may perform well under the training placement distribution yet fail under shifted or adversarial placements. The generalization gap (6) quantifies this fragility in a single scalar. If a method claims to learn an internal belief-like representation, calibration diagnostics test whether predicted probabilities correspond to empirical frequencies. This is especially relevant when comparing learned approaches to belief-based baselines.

Operational decision We report $\mathbb{E}[\tau]$ under at least: (i) uniform random placements, (ii) a biased scripted placement family, and (iii) placements from a learned/adversarial defender (when applicable). We report Δ_{gen} whenever more than one placement distribution is used, and we include calibration diagnostics only when the model emits interpretable probabilities or a proxy can be extracted.

1.7 Summary of What Is Fixed vs. What Is Chosen

Fixed by the problem. Hidden layout, hit/miss/sunk observation channel, feasibility of actions, termination condition, and the shots-to-win objective.

Chosen by the method. Reward surrogate, policy parameterisation (feedforward vs. recurrent), inclusion of heatmap baseline, self-play optimization schedule, and evaluation protocol beyond the primary objective metric.