LEARNING PURSUIT POLICIES WITH BELIEF-WEIGHTED INFORMATION-CONSISTENT SEARCH IN THE SCOTLAND YARD BOARD GAME

Anonymous authors
Paper under double-blind review

ABSTRACT

Pursuit-evasion games in which the pursuers cannot see the evader are a natural testbed for search under imperfect information, and the board game Scotland Yard is their best-known large instance. CFR-based equilibrium solvers are costly over a long horizon with combinatorial joint actions. Search over sampled evader locations avoids this cost but fails in two opposite ways: the pursuers' search lets them act differently in worlds they cannot tell apart, and the evader's search lets the pursuers it imagines see it. We observe that in one-sided games both failures disappear if every pursuer decision, in either role's search, depends only on the public history. On this principle we build a self-play learning framework. Its pursuer nodes are keyed by the public history and shared by all possible evader locations, and the evader expands a root for every location the pursuers cannot rule out. A learned location prior reweights the simulation backups in the pursuer search, and an autoregressive policy represents the pursuers' joint action. Through self-play, two role-specific networks learn to match search policies and predict game outcomes. On an adapted version of Scotland Yard, our pursuers win at least 98% of games against every evader we test, even without search. The best baseline wins 86% of games against learned evaders on average.

1 INTRODUCTION

Pursuit-evasion games model a team of pursuers trying to capture an evader. They arise wherever someone must be found or intercepted, from urban security to search and rescue with mobile robots (Chung et al., 2011). Learning-based pursuers now scale to large urban graphs (Li et al., 2023; 2024; Lu et al., 2026b), but most assume that the pursuers see the evader. In many real chases the information runs the other way: the evader watches the pursuers, while the pursuers receive only sparse clues about where it might be. The board game Scotland Yard captures this situation in a well-defined form (Nijssen & Winands, 2012). Five detectives (the pursuers) chase Mr. X (the evader) across the transit network of London. The evader sees every pursuer move, whereas the pursuers see only which transport the evader takes and learn its location in a few fixed rounds. Hence, catching the evader requires inferring where it could be as much as coordinating the pursuit.

Scotland Yard is an imperfect-information game, and such games are hard for a reason that perfect-information games do not share: the value of an action depends on what the players know and believe. Whether the pursuers should close in from the east or from the west depends on where the evader is likely to be, which in turn depends on how it plays. A position therefore cannot be evaluated in isolation (Brown et al., 2020). Counterfactual regret minimization (CFR) confronts this directly and converges to a Nash equilibrium in two-player zero-sum games (Zinkevich et al., 2007; Lanctot et al., 2009). Combined with abstraction and online subgame re-solving over public belief states, CFR has enabled superhuman performance in poker (Moravčík et al., 2017; Brown & Sandholm, 2018; 2019; Brown et al., 2020). Student of Games extends CFR method to several games, including a tiny map of Scotland Yard with only 11 nodes (Schmid et al., 2023).

However, these methods are expensive for Scotland Yard. Tabular CFR repeatedly traverses the game tree, which is out of reach for a game of up to dozens of decision rounds and millions of joint pursuer actions per round. Re-solving methods avoid the full traversal but solve a subgame

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

equilibrium at every decision, at a cost that grows with the number of joint pursuer actions (Brown et al., 2020). Thus, the difficulty of Scotland Yard is not the belief, which ranges over at most 199 stations, but the equilibrium computation repeated at every step. This motivates a lighter alternative: Monte Carlo tree search (MCTS). MCTS estimates action values by sampling trajectories instead of enumerating the tree, concentrating its budget on promising moves (Kocsis & Szepesvári, 2006; Browne et al., 2012). Guided by learned policies and values, it has mastered Go and chess (Silver et al., 2018; Schrittwieser et al., 2020). By sampling candidate actions, it also handles large action spaces (Hubert et al., 2021) such as the pursuers' joint actions. For imperfect information, determinization samples a hidden state consistent with the observations and searches as if it were known (Ginsberg, 2001; Long et al., 2010). Determinized MCTS has been shown to work in Scotland Yard (Nijssen & Winands, 2012), and AlphaZero with sampled worlds is a surprisingly strong baseline in other imperfect-information games (Blüml et al., 2023).

Determinization treats the sampled state as if everyone knew it, and in Scotland Yard this error appears on both sides of the board. The pursuers overestimate their *own* knowledge. Searching each sampled world separately lets them act differently in worlds they cannot tell apart, such as chasing east in one world and west in another. Their search is then over-optimistic, a failure known as strategy fusion (Frank & Basin, 1998). The evader overestimates its *opponents'* knowledge. It knows its location, so its own decisions do not fuse. If it searches only from that location, however, the pursuers it imagines act as if they could see it (Wisser, 2015; Goodman, 2019). Its search is then over-pessimistic and never learns to exploit the pursuers' uncertainty. Both failures have remedies in MCTS without learned networks. Information-set MCTS makes the pursuers act the same in worlds they cannot tell apart (Cowling et al., 2012). Self-determinization lets the evader also search from locations it knows to be false, so that the pursuers it imagines cannot see it (Cowling et al., 2015). These remedies, however, come from search without learning. Self-determinization also keeps a separate tree for every information set a player could be in, and it has been demonstrated only on a game with a handful of them. AlphaZe**, which brings learned networks to sampled-world search, adopts neither remedy and keeps both failures (Blüml et al., 2023). This leads to our question:

Can belief-weighted search, combined with learned networks, produce strong pursuers in a one-sided partially observable game without equilibrium solving?

We show that this is possible because the one-sided partially observable game removes the key difficulty through the following rule. The pursuers hold no private information, so each public history corresponds to exactly one pursuer information set. The evader, in turn, always knows the true state. Hence, sampled worlds cause neither failure as long as *every pursuer decision, in either role's search, depends only on the public history*. Based on this rule, we propose **S**elf-play learning via belief-**W**eighted **I**n**F**ormation-consistent MC**T**S (SWIFT). Its search is information-consistent, with the same kind of graph for both roles: pursuer nodes are keyed by the public history and shared by all worlds, which rules out the first failure. The evader additionally expands one root for every location the pursuers cannot rule out and reads only the root of its true location, which rules out the second failure of search. Its search is also belief-weighted: in the pursuers' search, a learned location prior reweights the worlds toward where the evader is likely to be. Through self-play, SWIFT trains two networks to reproduce the results of these searches: a pursuer network, whose autoregressive policy represents the joint action, and an evader network. Our contributions are as follows.

- We show that in one-sided partially observable games, a single public-history search graph shared by both roles prevents strategy fusion in the pursuers' search and keeps the true evader location from leaking to the imagined pursuers in the evader's search (Sections 4.2 and 4.3).
- We prove that both searches are information-consistent and analyze the error that remains in the pursuer search, which motivates the location prior (Section 4.3 and Appendix A.3).
- We propose SWIFT, a self-play framework that trains role-specific networks on the search construction with learned belief weighting and an autoregressive joint policy (Section 4).
- On an adapted version of Scotland Yard, SWIFT wins at least 98% of games against each of six evaluated evader agents. Against the three learned evaders, it achieves a mean win rate of 98.7%, compared with 85.6% for the strongest baseline (Section 5).

2 Related Work

Equilibrium Computation in Imperfect-Information Games. CFR and its sampling variants (Zinkevich et al., 2007; Lanctot et al., 2009), with abstraction and subgame re-solving, underlie the poker agents Libratus and Pluribus (Brown & Sandholm, 2018; 2019). DeepStack, ReBeL and Student of Games (Moravčík et al., 2017; Brown et al., 2020; Schmid et al., 2023) soundly re-solve public belief states with learned value functions. Without search, fictitious self-play (Heinrich & Silver, 2016; Zhang et al., 2021), population-based training (Lanctot et al., 2017; McAleer et al., 2020) and regularized self-play (Perolat et al., 2022; Sokota et al., 2025) also approach equilibria. However, tabular CFR is out of reach on Scotland Yard and re-solving is costly at every decision.

MCTS with Imperfect Information. Perfect-information Monte Carlo search samples hidden states and plans in each as if it were fully observed. It is strong in practice (Ginsberg, 2001; Long et al., 2010) but suffers from strategy fusion (Frank & Basin, 1998). In Scotland Yard, Nijssen & Winands (2012) showed that sharing the pursuers' nodes across determinizations outperforms separate trees. Information-set MCTS (Cowling et al., 2012) makes such sharing its defining principle. Opponent modeling remains a weak spot: opponents simulated under the searching player's determinization act on information they do not have (Wisser, 2015; Goodman, 2019). Self-determinization with one tree per opponent information set (Cowling et al., 2015) and re-determinization at every opponent node (Goodman, 2019) repair this. AlphaZe** (Blüml et al., 2023) brings learning to sampled-world search. However, it runs an independent perfect-information search in each world and averages the root policies, which keeps both failures.

Pursuit-Evasion Games. Pursuit-evasion on graphs has a long theoretical history (Parsons, 1978; Chung et al., 2011). For one-sided partial observability, Horák & Bošanský (2017) and Horák et al. (2023) give provably convergent algorithms that represent the value as a function of the pursuers' belief. Their cost therefore grows with the belief space. Learning-based pursuers (Li et al., 2023; 2024; Lu et al., 2026b) mostly observe the evader. Closest to our setting, R2PS (Lu et al., 2026a) gives the pursuers only a set of possible evader positions. However, it distills dynamic-programming belief policies without online search, and it considers two pursuers with range-limited observation.

3 Problem Formulation

We study an adapted version of Scotland Yard (details in App. B.1), in which a team of five pursuers tries to catch one evader. The game is played on the classic map of London, a graph $G = (V, \mathcal{E})$ with $|V| = 199$ stations linked by taxi, bus, underground and ferry edges. In each of at most $T = 22$ rounds, the evader first moves along one edge by paying a ticket of the matching type. Then, the pursuers move jointly each along one edge, in an order they choose. Each pursuer holds 10 taxi, 8 bus and 4 underground tickets and cannot take the ferry. The evader has unlimited regular tickets and five black tickets that are valid on any edge. The pursuers win if one of them reaches the evader's station or the evader has no legal move. The evader wins if it survives T rounds or no pursuer can move. Moreover, the evader sees the whole state. The pursuers see their own positions and tickets. After every evader move, the pursuers learn which ticket it used. They learn its location only in the reveal rounds $R = \{3, 8, 13, 18\}$. The game starts by placing the evader and the five pursuers on distinct stations drawn uniformly from the 18 classic start stations. Compared with the classic rules, we remove the double-move ticket, give the evader unlimited regular tickets and end the game after 22 rounds. These changes marginally simplify the game, but keep its core features.

Formally, Scotland Yard is a one-sided partially observable pursuit-evasion game (Horák & Bošanský, 2017). Since the pursuers share all their observations and move as a team, we treat them as a single player P that chooses one joint action per round. This gives a two-player zero-sum game with perfect recall between P and the evader E, in which only P has imperfect information (Horák et al., 2023). Let $x_t \in V$ be the evader's location at the start of round t, with x_1 its start station. The public history h_t collects everything both players have observed so far: the pursuers' positions, tickets and moves, the evader's tickets, and the revealed locations. An information set groups the states that a player cannot tell apart. The pursuers' information set at h_t contains all states consistent with h_t, which differ only in the evader's hidden trajectory. The evader, in contrast, always knows the true state. Hence every public history defines exactly one pursuer information set, and the evader knows which state of this set is the true one. Section 4 builds on these two facts. A pursuer policy

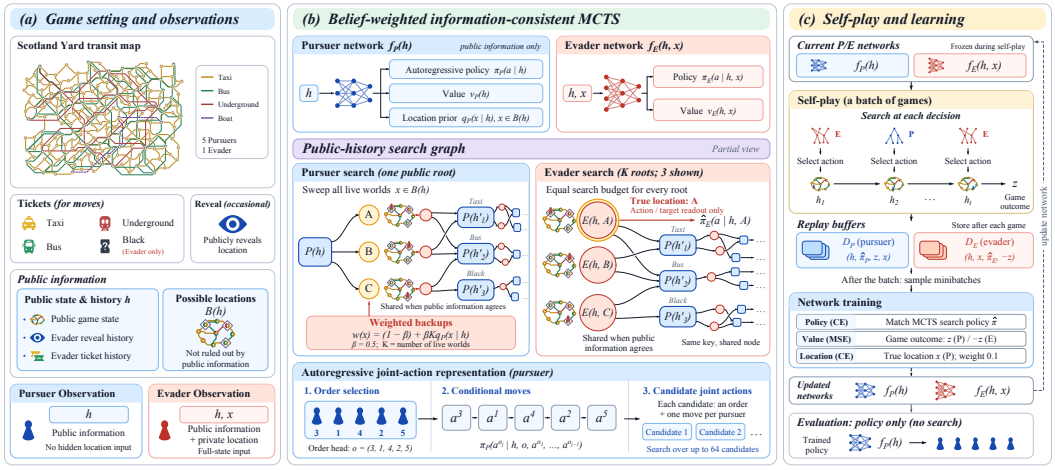


Figure 1: Overview of SWIFT. (a) Game setting: the pursuers observe only the public history h, from which they derive the possible locations $B(h)$, while the evader also observes its location x. (b) Belief-weighted information-consistent MCTS. Both searches use one kind of graph whose pursuer nodes $\mathsf{P}(h)$ are keyed by the public history, so no pursuer decision depends on the evader's hidden location. The pursuer search (left) sweeps all live worlds from one root and weights their backups by the location prior q_P; the evader search (right) expands one root per possible location and reads only the root of its true location. The pursuer network represents the joint action autoregressively (bottom). (c) Self-play with these searches produces the training targets of the two role-specific networks, which are evaluated without search.

is a map $\pi^\mathsf{P}(\cdot \mid h)$ from public histories to joint actions, and an evader policy a map $\pi^\mathsf{E}(\cdot \mid h, x_{1:t})$. A terminal history has utility $+1$ if the pursuers win and -1 otherwise. The pursuers maximize the expected utility and the evader minimizes it.

What the pursuers need to reason about is where the evader is now. Since transitions and utilities depend on the evader's past only through its current location, the relevant part of the pursuers' information set is the set $B(h_t)$ of locations consistent with the public history. We call them the *possible locations*. The pursuers can compute this set exactly. Let $\mathbf{p}_t$ denote the pursuers' positions, τ_t the ticket of the evader's move in round t, and $\mathcal{E}_\tau$ the edges usable with ticket τ, where $\mathcal{E}_{\text{black}} = \mathcal{E}$. Starting from the thirteen start positions not taken by pursuers, the set evolves as

$$B_t^+ = \begin{cases} \{x_{t+1}\}, & t \in R, \\ \{\, v' \notin \mathbf{p}_t : (v, v') \in \mathcal{E}_{\tau_t} \text{ for some } v \in B(h_t) \,\}, & t \notin R, \end{cases} \qquad B(h_{t+1}) = B_t^+ \setminus \mathbf{p}_{t+1}. \quad (1)$$

Each round thus has two decision points. The evader moves at h_t, where the possible locations are $B(h_t)$, and the pursuers move after it, where the possible locations are B_t^+. A belief turns this set into probabilities: given an evader policy π^E, the pursuers' belief $b(x \mid h_t) = \Pr(x_t = x \mid h_t; \pi^\mathsf{E})$ is positive only on $B(h_t)$. The values of $b(x \mid h_t)$ depend on π^E, which the pursuers do not know. The possible locations tell where the evader can be, and the belief where it is likely to be. A public history together with one possible location is called a *world*.

4 SWIFT

SWIFT trains a pursuer network f_P and an evader network f_E by self-play (Figure 1 and Algorithm 1). At every decision of a self-play game, the player to move runs a search guided by both networks. It plays a move chosen from the root statistics of the search and stores the improved policy $\hat{\pi}$ derived from them as a policy target. The outcome of the game later provides the value target. Search is thus the policy improvement step of training, and the trained networks can then play with or without it. Both searches build the same kind of graph and obey one rule: *every pursuer decision, in either search, depends only on the public history*. The pursuer search enforces it by keying pursuer nodes by the public history (Section 4.2). The evader search does the same and also expands one root per possible location (Section 4.3).

Algorithm 1 SWIFT

for iteration $= 1, 2, \ldots$ **do**
  for each self-play game **do**
    draw the start positions; $h \leftarrow$ initial public history; $x \leftarrow$ evader location
    while the game is not over **do**
      if the pursuers are to move **then** $\hat{\pi} \leftarrow \textsc{PursuerSearch}(h)$ ▷ Section 4.2
      else $\hat{\pi} \leftarrow \textsc{EvaderSearch}(h, x)$ ▷ Section 4.3
      store $(h, x, \hat{\pi})$ for the player to move
      select a move from the root statistics
      update h and x
    label the stored examples with the outcome $z \in \{\pm 1\}$ and add them to the replay buffers
  update f_{P} and f_{E} on their replay buffers ▷ Section 4.4

4.1 Role-Specific Networks

Each network sees exactly what its player sees. The pursuer network $f_{\mathsf{P}}(h)$ receives only public features of h and has three heads: a policy $\pi_{\mathsf{P}}(\cdot \mid h)$ over joint actions, a value $v_{\mathsf{P}}(h)$, and a location prior $q_{\mathsf{P}}(\cdot \mid h)$. This prior is a distribution over $B(h)$ that is trained to predict the evader's true location, so it is a learned approximation of the belief b under the self-play evader. The pursuer search uses it to weight worlds. The evader network $f_{\mathsf{E}}(h, x)$ also receives the evader's location and outputs a policy $\pi_{\mathsf{E}}(\cdot \mid h, x)$ over its legal moves and a value $v_{\mathsf{E}}(h, x)$. Both are relational graph networks over the 199 stations and share no parameters (Appendix B.2).

The pursuers' joint action needs a dedicated policy. It consists of a move order over the five pursuers and one move per pursuer, executed in this order. Since each move changes which stations are free, the order matters, and the joint space is huge (Section 3). We make it tractable in two steps. First, orders that differ only in how they rank pursuers whose moves cannot interfere (Appendix B.2) lead to the same result. The network therefore places probability only on one canonical order per equivalence class. Second, the network chooses the canonical order and then the pursuers' moves one at a time, each conditioned on the moves already chosen through a recurrent state (Metz et al., 2017). The probability of a joint action is thus a product of per-step probabilities and can be evaluated exactly, as exhaustive expansion in the search and the training loss require.

4.2 Public-History Search Graph and Pursuer Search

The pursuers cannot tell the worlds in $B(h)$ apart, so they must choose the same action in all of them. Planning separately in each sampled world breaks this rule and causes strategy fusion. Following single-tree determinization in Scotland Yard (Nijssen & Winands, 2012) and information-set MCTS (Cowling et al., 2012), we key the pursuers' decision nodes by the public history. The search graph therefore has two kinds of nodes (Figure 1b). A pursuer node $\mathsf{P}(h)$ is keyed by the public history alone and is shared by all worlds. An evader node $\mathsf{E}(h, x)$ is also keyed by the evader's current location, which the evader knows.

Simulations. Each simulation samples one world from the pursuers' information set at the root history h_0 (information-set sampling), namely one live possible location $x \in B(h_0)$ at which the game is not over. It then follows this world until it reaches a leaf. With a budget of N simulations, the draws cycle through all K live worlds $\lceil N/K \rceil$ times, each time in a fresh random order. Since the next public history records only the ticket, plus the location in reveal rounds, simulations from different worlds that use the same ticket reach the same pursuer node. There, the joint action is selected from statistics shared by all worlds, and only its result, such as a capture, is decided in the simulated world. Edges are selected by PUCT (Silver et al., 2018): at node n, the simulation takes the edge a that maximizes $Q(n, a) + c\,\pi_n(a)\sqrt{W(n)}/(1 + W(n, a))$. Here c is a constant and π_n is the policy of the player to move at n. The visit counts $W(n, a)$ and $W(n)$ of the edge and the node are weighted by world (see below), and $Q(n, a)$ is the weighted mean value from that player's perspective. A new leaf is evaluated by the network of the player to move, $f_{\mathsf{P}}(h)$ at pursuer nodes and $f_{\mathsf{E}}(h, x)$ at evader nodes. All values are stored from the pursuers' perspective. A pursuer node expands all legal canonical joint actions if there are at most 64 of them. Otherwise, it draws 64 joint actions from the policy with replacement and uses their frequencies as priors (Hubert et al., 2021).

Weighting Worlds with the Location Prior. Visiting all worlds equally often treats them as equally likely, although the evader's play makes some locations far more likely than others. The location prior corrects this by importance weighting: a simulation in world x contributes with the weight

$$w(x) \;=\; \frac{(1-\beta)/K + \beta\, q_{\mathsf{P}}(x \mid h_0)}{1/K} \;=\; (1-\beta) + \beta K\, q_{\mathsf{P}}(x \mid h_0), \tag{2}$$

which turns the uniform visits into visits from a mixture of the uniform distribution and the prior. We use $\beta = 0.5$. The weight is computed once at the root and applies to every node on the path: each backup adds $w(x)$ to the weights W and $w(x)\,v$ to the value sums. Reweighting instead of resampling keeps every world with a weight of at least $1-\beta$, so a wrong prior can shift the estimates but cannot remove a world. Appendix A.3 bounds one term of the error of a root value by the distance between the worlds' shares of the root weight and the belief under the evader's policy. Mixing in the prior shrinks this distance whenever the prior is closer to the belief than the uniform one is. Unlike the hand-crafted location categories of Nijssen & Winands (2012), the prior is learned. It predicts where the self-play evader is rather than computing a Bayesian posterior.

4.3 Evader Search

The evader knows its location, so its own decisions cannot fuse across worlds. The danger lies in the pursuers it imagines. Suppose the evader is at A and the pursuers consider A and B possible. If the evader searched from A alone, every pursuer node in its search would be reached only by world A. The imagined pursuers would then chase A as if they could see the evader, and the evader would never learn to exploit the real pursuers' uncertainty.

One Root per Possible Location. Following the self-determinization of Cowling et al. (2015), the evader search expands one root $\mathsf{E}(h_0, x')$ for every live possible location $x' \in B(h_0)$, including locations that the evader knows to be false. It runs $\lceil N/K \rceil$ simulations from each root on one graph, built as in Section 4.2 and shared by all roots. Simulations from the roots of A and B that produce the same public history meet at the same pursuer node, so the imagined pursuers must choose one move for both worlds, exactly as the real pursuers must (Figure 1b, right). After the search, only the root of the true location is read, to select the move and to form the policy target. The other roots influence the decision only through the nodes they share with it, mainly pursuer nodes.

Why One Graph Suffices. In general games, this remedy needs one tree for every information set an opponent could be in (Cowling et al., 2015), or a re-sampled hidden state at every opponent node (Goodman, 2019). In Scotland Yard, the evader always knows the pursuers' information set, which is the public history (Section 3). These opponent trees therefore collapse into one graph: the graph of Section 4.2, entered through one root per possible location. Deeper nodes need no re-sampling either: by Equation 1, every location that the pursuers consider possible at a later history descends from a location in $B(h_0)$, that is, from one of the roots.

All roots receive the same budget, because any bias toward the true world would leak it into the opponent model (Cowling et al., 2015). The price is that the true root receives only about N/K simulations. The evader search weights all worlds equally and ignores the location prior. Its imagined pursuers already inherit the prior through their policy, which is distilled from prior-weighted searches. Allocating simulations by the prior would also starve the true root exactly when the evader hides well. Evader nodes record only the current location rather than the whole trajectory, which loses nothing against pursuers who condition on the public history (Lemma 1 in Appendix A). Together, the two searches implement the rule of this section (proof in Appendix A):

Proposition 1 (Information consistency)**.** *For fixed networks and in either search, (i) the edge selected at a pursuer node depends only on that node's statistics, which pool all worlds, and not on the world of the current simulation; and (ii) the statistics of every node depend only on the root history h_0 and on randomness independent of the evader's true location.*

Part (i) rules out strategy fusion. Part (ii) implies that the pursuer search has the same output distribution for all states with the same public history. It also implies that in the evader search the true location only selects the root to read, so the imagined pursuers cannot see the evader. Appendix A.3 analyzes the error that remains in the pursuer search.

Table 1: Pursuer win rate (%) of the final networks without search, over 512 games per cell. †: the pursuer was trained against this evader. Bold marks the best pursuer against each evader.

	Learned evaders			Weak evaders		
Pursuer	SWIFT	P2SRO	AlphaZe**	Random	Greedy	Territory
PPO-GNN	66.2	74.8	91.8	**100.0**	**100.0**†	95.7
PPO-MLP	15.6	19.9	25.6	97.1	99.8†	47.3
MAPPO-GNN	74.4	88.3	94.1	**100.0**	**100.0**†	98.4
P2SRO	85.7	68.2†	92.8	**100.0**	99.6	98.6
AlphaZe**	26.2	29.9	32.4†	**100.0**	91.4	56.4
SWIFT	**98.2**†	**98.6**	**99.2**	**100.0**	**100.0**	**99.4**

4.4 Training

The policy target $\hat{\pi}$ is the distribution of root edge weights after two corrections from KataGo (Wu, 2019): visits that exist only because of forced playouts are pruned, and the weight of the action with the best lower confidence bound on its value is raised. The value target is the outcome $z \in \{\pm 1\}$ of the game from the pursuers' perspective. The pursuer network minimizes

$$\mathcal{L}_{\mathsf{P}} = -\sum\nolimits_{a} \hat{\pi}(a) \log \pi_{\mathsf{P}}(a \mid h) + \big(v_{\mathsf{P}}(h) - z\big)^2 - \lambda \log q_{\mathsf{P}}(x \mid h), \tag{3}$$

where $\log \pi_{\mathsf{P}}(a \mid h)$ is the log-probability of the complete joint action, including its order, and $\lambda = 0.1$. The last term uses the evader's true location x, which self-play records; it serves only as a label and is never an input of f_{P}. The evader network minimizes the same loss without the location term and with $-z$ as its value target. Each network has its own optimizer and replay buffer, and neither is trained with a policy gradient. Since the networks are fitted to the output of a search that acts as a policy improvement operator (Grill et al., 2020; Danihelka et al., 2022), training can be viewed as an approximate generalized policy iteration for two coupled players. We do not claim that it converges to an equilibrium. Appendix B lists the remaining settings, including further KataGo refinements and random relabeling of the pursuers.

5 Experiments

Research Questions. We evaluate SWIFT on four questions. **RQ1:** Are its pursuers stronger than those of existing methods, especially against learned evaders? **RQ2:** How does its strength grow during training? **RQ3:** Which key components of SWIFT contribute to its strength? **RQ4:** How much does the start configurations matter?

Training and Evaluation. All methods run on one machine (Intel Core Ultra 7 265K CPU, NVIDIA RTX 5090 GPU) and train for five hours, except P2SRO, which receives twelve hours so that its populations converge. SWIFT runs $N = 300$ simulations per move during training. All evaluations play the trained networks *without search*, so they measure what training has put into the networks rather than extra computation at test time (Appendix D.3 analyzes the effect of search at test time). Each pairing plays 512 games from one shared set of start positions.

Pursuer Baselines. We compare with five learned pursuers (Appendix C). *PPO-GNN* and *PPO-MLP* train the team as one PPO agent with an autoregressive joint policy. *MAPPO-GNN* (Yu et al., 2022) trains one actor shared by all pursuers. These three PPO-type baselines train against the Greedy evader. *P2SRO* (McAleer et al., 2020) grows pursuer and evader populations with PPO-GNN oracles (Lanctot et al., 2017). *AlphaZe*** (Blüml et al., 2023) shares our network architecture.

Evader Opponents. The *weak evaders* are Random and the heuristics Greedy and Territory (see Appendix C). The *learned evaders* are the final evaders of SWIFT and AlphaZe** and P2SRO.

5.1 Main Resutls

RQ1: SWIFT wins at least 98% of games against every evader, including learned evaders it never met during training. Against the three learned evaders (Table 1; Appendix D.6 shows com-

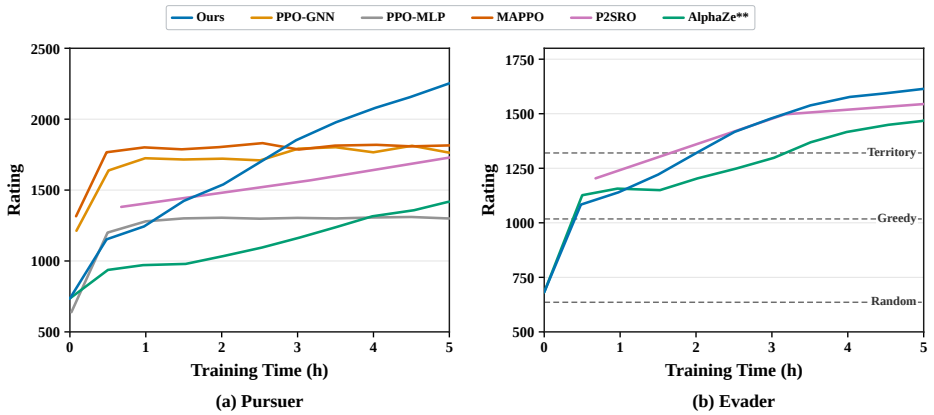


Figure 2: Ratings of the pursuer (a) and evader (b) checkpoints of all methods during training. The dashed lines in (b) mark the ratings of the three weak evaders.

plete games), SWIFT wins 98.2–99.2% of games, 98.7% on average. The best baseline MAPPO-GNN only wins 85.6% on average, and no baseline exceeds 94.1% against any learned evader. The evader of SWIFT is also the hardest opponent for every pursuer except P2SRO, which does worse against its own evaders. Against the weak evaders most methods win nearly all games. Only PPO-MLP and AlphaZe** often lose to Territory. Since the pursuers have a structural advantage in Scotland Yard (Nijssen & Winands, 2012), we compare methods only under the same evader.

Searching the sampled worlds separately leaves AlphaZe far behind.** Although its pursuers search with more simulations per move than those of SWIFT (up to 500 against about 300), AlphaZe** wins only 26.2–32.4% of games against the learned evaders. This is less than PPO-GNN, which uses no search at all. We attribute this to the two failures. Its pursuers plan in each world as if they could see the evader, and its evader plans against pursuers that can see it, so the targets that both networks distill are biased. Consistently, the evader of AlphaZe** is the weakest learned evader (Table 1 and Figure 2b). AlphaZe** also lacks the location prior, but this explains little of the gap: removing the prior costs SWIFT only about 200 rating points (Figure 3a).

RQ2: SWIFT keeps improving after every baseline levels off, and ends with both the strongest pursuer and the strongest evader. We rate the checkpoints of all methods by cross-play with a Bradley–Terry model (Appendix C). Among the pursuers (Figure 2a), MAPPO-GNN and PPO-GNN learn fastest and reach about 1700–1800 within the first hour, but then stay flat. PPO-MLP stays at about 1300. SWIFT starts lower, overtakes both within three hours and reaches about 2250 at five hours, still rising. AlphaZe** improves throughout but ends about 800 points below SWIFT. P2SRO improves steadily over the first five of its twelve hours. The PPO-type baselines likely plateau because they beat their training opponent, Greedy, in nearly all games from the first checkpoint on (Appendix D.2). SWIFT, in contrast, trains against its own evader, which keeps improving. The evaders follow the same order (Figure 2b): the evader of SWIFT passes Territory after about two hours and ends highest.

5.2 Ablation Study and Effect of the Start Configuration

RQ3: The location prior helps through its use in the search, not as an auxiliary loss. Two ablations separate how the location prior can act (Figure 3). SWIFT w/o belief weighting removes the head, so the search weights all possible locations equally. SWIFT w/ auxiliary belief loss trains the head with the same loss but does not use it in the search. The three variants coincide for about two hours. Then SWIFT pulls ahead, reaching about 2150 at five hours against about 1950 and 1910 for the two ablations. It still leads SWIFT w/o belief weighting by about 170 points after eight hours (insets). Training the head without using it therefore brings no gain. The gain comes from reweighting the backups by Equation 2, which targets the world-distribution term of Proposition 2 (Appendix A.3). Since the prior enters only the pursuers' search, the evaders of the three variants stay within about 50 points of one another (Figure 3b). Appendix D.4 illustrates the location prior.

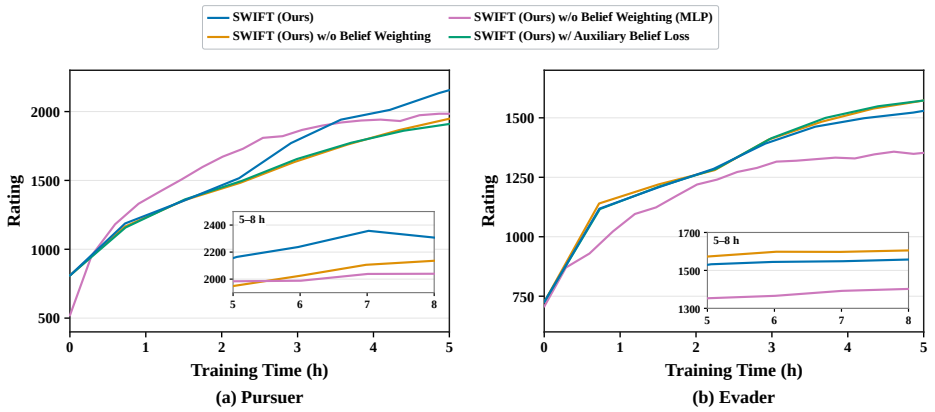


Figure 3: Ratings of SWIFT and its ablations.

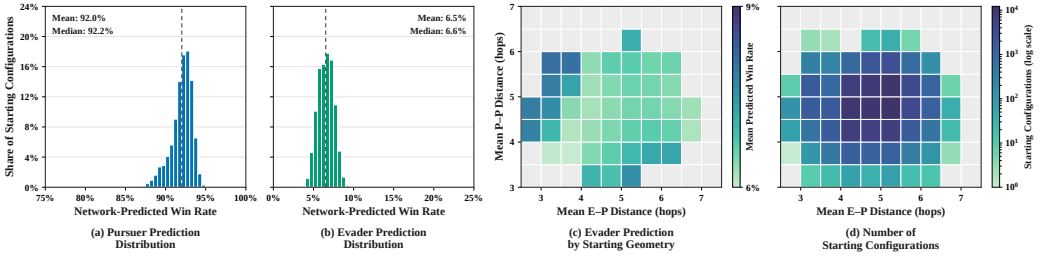


Figure 4: Analysis of the predicted win rates over all 111,384 start configurations.

With longer training, the graph encoder beats an MLP of about the same size. Figure 3 also compares encoders. SWIFT w/o belief weighting (MLP) replaces the graph encoder of SWIFT w/o belief weighting by a multilayer perceptron with about as many parameters (Appendix B.2). Its pursuer learns faster at first, probably because the MLP is cheaper to evaluate and runs twice as many self-play workers (Appendix B.3), and it is still slightly ahead after five hours. SWIFT w/o belief weighting overtakes it during the sixth hour, however, and leads by about 100 points after eight hours (insets). The MLP evader is weaker throughout, by about 200 points (Figure 3b).

RQ4: The start configuration shifts the predicted outcome by only about three percentage points. We evaluate the value heads of our strongest pursuer and evader, the final SWIFT networks, on all $18\binom{17}{5} = 111{,}384$ start configurations. This analysis thus rests on predictions, not simulated games. Most configurations have mean evader–pursuer (E–P) and pursuer–pursuer (P–P) distances of 4 to 5.5 hops (Figure 4d), where the predicted evader win rate is about 6–7% (Figure 4c). It reaches about 9% only at the sparse rim, when the evader starts about three hops from the pursuers.

6 Conclusion

In a one-sided partially observable game, sampled-world search can respect what each player knows and still be strong. SWIFT keys every pursuer node by the public history and gives the evader one root per location the pursuers cannot rule out, so a single search graph excludes both failures of sampled-world search. Distilled into role-specific networks by self-play, this search yields pursuers that win at least 98% of games against every evader we test, even without search at test time.

AI Use Statement

In this work, we used generative AI tools to assist with developing the conceptual framework, formulating mathematical claims and writing proofs, designing and implementing the methods and experiments, translation, and interpreting results; the remaining tasks with required disclosure are not applicable to this work. Additionally, we used generative AI tools to assist with writing and

editing code, figures and the paper, and with searching the literature and formatting references. The AI tools only assisted the authors, who reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Ethics Statement

All authors have read and adhere to the ICLR Code of Ethics. This work studies search and learning in a board game and uses no sensitive or personal data. In the human evaluation (Appendix D.1), one of the authors played the evader through the web interface of Appendix E, and only game records were kept. We do not foresee direct negative societal impacts of this work.

Reproducibility Statement

Section 4 and Algorithm 1 specify the search and the training procedure. Appendix B gives the rules of our environment (Table 2), the network inputs and architectures (Appendix B.2) and all hyperparameters (Table 4). Appendix C describes the baselines, the weak evaders, the evaluation protocol and the rating procedure, and Appendix A contains the complete proofs. The complete code are available at https://anonymous.4open.science/r/paper-code-6750/.

References

Jannis Blüml, Johannes Czech, and Kristian Kersting. AlphaZe**: AlphaZero-like baselines for imperfect information games are surprisingly strong. *Frontiers in Artificial Intelligence*, 6:1014561, 2023.

Noam Brown and Tuomas Sandholm. Superhuman ai for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359(6374):418–424, 2018.

Noam Brown and Tuomas Sandholm. Superhuman ai for multiplayer poker. *Science*, 365(6456): 885–890, 2019.

Noam Brown, Anton Bakhtin, Adam Lerer, and Qucheng Gong. Combining deep reinforcement learning and search for imperfect-information games. In *Advances in Neural Information Processing Systems*, volume 33, pp. 17057–17069, 2020.

Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, 2012.

Timothy H. Chung, Geoffrey A. Hollinger, and Volkan Isler. Search and pursuit-evasion in mobile robotics. *Autonomous Robots*, 31(4):299–316, 2011.

P. I. Cowling, E. J. Powley, and D. Whitehouse. Information set monte carlo tree search. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(2):120–143, 2012.

Peter I. Cowling, Daniel Whitehouse, and Edward J. Powley. Emergent bluffing and inference with Monte Carlo tree search. In *2015 IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 114–121, 2015.

Ivo Danihelka, Arthur Guez, Julian Schrittwieser, and David Silver. Policy improvement by planning with Gumbel. In *International Conference on Learning Representations*, 2022.

Ian Frank and David Basin. Search in games with incomplete information: a case study using bridge card play. *Artificial Intelligence*, 100(1-2):87–123, 1998.

M. L. Ginsberg. Gib: Imperfect information in a computationally challenging game. *Journal of Artificial Intelligence Research*, 14:303–358, 2001.

James Goodman. Re-determinizing information set monte carlo tree search in hanabi. *arXiv preprint arXiv:1902.06075*, 2019.

Jean-Bastien Grill, Florent Altché, Yunhao Tang, Thomas Hubert, Michal Valko, Ioannis Antonoglou, and Rémi Munos. Monte-Carlo tree search as regularized policy optimization. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3769–3778. PMLR, 2020.

Johannes Heinrich and David Silver. Deep reinforcement learning from self-play in imperfect-information games. *arXiv preprint arXiv:1603.01121*, 2016.

Karel Horák and Branislav Bošanský. Dynamic programming for one-sided partially observable pursuit-evasion games. In *Proceedings of the 9th International Conference on Agents and Artificial Intelligence*, pp. 503–510. SCITEPRESS - Science and Technology Publications, 2017.

Karel Horák, Branislav Bošanský, Vojtěch Kovařík, and Christopher Kiekintveld. Solving zero-sum one-sided partially observable stochastic games. *Artificial Intelligence*, 316:103838, 2023.

Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Mohammadamin Barekatain, Simon Schmitt, and David Silver. Learning and planning in complex action spaces. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4476–4486. PMLR, 2021.

Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *Machine Learning: ECML 2006*, Lecture Notes in Computer Science, pp. 282–293. Springer Berlin Heidelberg, 2006.

Marc Lanctot, Kevin Waugh, Martin Zinkevich, and Michael Bowling. Monte Carlo sampling for regret minimization in extensive games. In *Advances in Neural Information Processing Systems 22*, 2009.

Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Pérolat, David Silver, and Thore Graepel. A unified game-theoretic approach to multiagent reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 30, pp. 4190–4203, 2017.

Pengdeng Li, Shuxin Li, Xinrun Wang, Jakub Cerny, Youzhi Zhang, Stephen McAleer, Hau Chan, and Bo An. Grasper: A generalist pursuer for pursuit-evasion problems. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 1147–1155, 2024.

Shuxin Li, Xinrun Wang, Youzhi Zhang, Wanqi Xue, Jakub Černý, and Bo An. Solving large-scale pursuit-evasion games using pre-trained strategies. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 11586–11594, 2023.

Jeffrey Long, Nathan Sturtevant, Michael Buro, and Timothy Furtak. Understanding the success of perfect information monte carlo sampling in game tree search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pp. 134–140, 2010.

Runyu Lu, Ruochuan Shi, Yuanheng Zhu, and Dongbin Zhao. R2ps: Worst-case robust real-time pursuit strategies under partial observability. In *International Conference on Learning Representations*, volume 2026, pp. 28843–28868, 2026a.

Runyu Lu, Peng Zhang, Ruochuan Shi, Yuanheng Zhu, Dongbin Zhao, Yang Liu, Dong Wang, and Cesare Alippi. Equilibrium policy generalization: A reinforcement learning framework for cross-graph zero-shot generalization in pursuit-evasion games. In *Advances in Neural Information Processing Systems*, volume 38, pp. 48810–48841, 2026b.

Stephen McAleer, John B. Lanier, Roy Fox, and Pierre Baldi. Pipeline PSRO: A scalable approach for finding approximate Nash equilibria in large games. In *Advances in Neural Information Processing Systems*, volume 33, pp. 20238–20248, 2020.

Luke Metz, Julian Ibarz, Navdeep Jaitly, and James Davidson. Discrete sequential prediction of continuous actions for deep RL, 2017.

Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisý, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017.

Pim Nijssen and Mark H. M. Winands. Monte carlo tree search for the hide-and-seek game scotland yard. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(4):282–294, 2012.

T. D. Parsons. Pursuit-evasion in a graph. In *Theory and Applications of Graphs*, Lecture Notes in Mathematics, pp. 426–441. Springer Berlin Heidelberg, 1978.

Julien Perolat, Bart De Vylder, Daniel Hennes, Eugene Tarassov, Florian Strub, Vincent de Boer, Paul Muller, Jerome T. Connor, Neil Burch, Thomas Anthony, Stephen McAleer, Romuald Elie, Sarah H. Cen, Zhe Wang, Audrunas Gruslys, Aleksandra Malysheva, Mina Khan, Sherjil Ozair, Finbarr Timbers, Toby Pohlen, Tom Eccles, Mark Rowland, Marc Lanctot, Jean-Baptiste Lespiau, Bilal Piot, Shayegan Omidshafiei, Edward Lockhart, Laurent Sifre, Nathalie Beauguerlange, Remi Munos, David Silver, Satinder Singh, Demis Hassabis, and Karl Tuyls. Mastering the game of stratego with model-free multiagent reinforcement learning. *Science*, 378(6623):990–996, 2022.

Martin Schmid, Matej Moravčík, Neil Burch, Rudolf Kadlec, Josh Davidson, Kevin Waugh, Nolan Bard, Finbarr Timbers, Marc Lanctot, G. Zacharias Holland, Elnaz Davoodi, Alden Christianson, and Michael Bowling. Student of games: A unified learning algorithm for both perfect and imperfect information games. *Science Advances*, 9(46):eadg3256, 2023.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588 (7839):604–609, 2020.

David Silver and Joel Veness. Monte-carlo planning in large POMDPs. In *Advances in Neural Information Processing Systems*, volume 23, pp. 2164–2172, 2010.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

Samuel Sokota, Eugene Vinitsky, Hengyuan Hu, J. Zico Kolter, and Gabriele Farina. Superhuman AI for Stratego using self-play reinforcement learning and test-time search, 2025.

Florian Wisser. An expert-level card playing agent based on a variant of perfect information Monte Carlo sampling. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 125–131, 2015.

David J Wu. Accelerating self-play learning in go. *arXiv preprint arXiv:1902.10565*, 2019.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. In *Advances in Neural Information Processing Systems 35*, pp. 24611–24624. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2022.

Li Zhang, Yuxuan Chen, Wei Wang, Ziliang Han, Shijian Li, Zhijie Pan, and Gang Pan. A monte carlo neural fictitious self-play approach to approximate nash equilibrium in imperfect-information dynamic games. *Frontiers of Computer Science*, 15(5):155334, 2021.

Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. Regret minimization in games with incomplete information. In *Advances in Neural Information Processing Systems 20*, 2007.

A Analysis and Proofs

This appendix first shows that evader strategies may depend on the current location alone (Lemma 1), which justifies keying evader nodes by (h, x) and simplifies the analysis. It then proves Proposition 1 and analyzes the error that remains in the pursuer search (Proposition 2). We use one fact about the game of Section 3: the next public history, the next location and the terminal utility depend on the past only through the current public history, the current location and the players' actions. We call a pursuer policy *public* if it depends on the public history only. As in Section 3, h_t is the public history and x_t the evader's location at the start of round t.

A.1 Evader Strategies That Depend on the Current Location Suffice

Lemma 1. *Let the pursuers play any public policy. Then every evader strategy, which may depend on the whole trajectory $x_{1:t}$, has a payoff-equivalent strategy that depends only on the public history and the current location. This equivalent strategy is the same for every public pursuer policy.*

Proof. Fix an evader strategy $\sigma^{\mathsf{E}}(\cdot \mid h_t, x_{1:t})$. For a pair $(h_t, x_{1:t})$, let $F(h_t, x_{1:t})$ be the product of three factors: the chance factor of the start positions, the evader factors $\sigma^{\mathsf{E}}(a_k \mid h_k, x_{1:k})$ for $k < t$, and the indicator that $x_{1:t}$ is consistent with h_t (legal moves, revealed locations, and no capture so far). Here the move a_k is determined by x_k, x_{k+1} and the ticket recorded in h_t. Under any public pursuer policy π^{P}, the probability of $(h_t, x_{1:t})$ is $F(h_t, x_{1:t})$ times the probabilities under π^{P} of the joint actions recorded in h_t. This second factor depends on h_t only. Hence, whenever π^{P} reaches h_t,

$$\Pr(x_{1:t-1} \mid h_t, x_t) \;=\; \frac{F(h_t, x_{1:t})}{\sum_{x'_{1:t-1}} F(h_t, x'_{1:t-1}, x_t)},$$

which does not involve π^{P}. Define $\bar{\sigma}^{\mathsf{E}}(a \mid h_t, x_t) = \sum_{x_{1:t-1}} \Pr(x_{1:t-1} \mid h_t, x_t)\, \sigma^{\mathsf{E}}(a \mid h_t, x_{1:t})$ with this conditional wherever the denominator is positive, and by any fixed rule elsewhere. Then $\bar{\sigma}^{\mathsf{E}}$ is the same for every public π^{P}. Now fix a public π^{P}. We show by induction on t that (h_t, x_t) has the same distribution under σ^{E} and under $\bar{\sigma}^{\mathsf{E}}$. At $t = 1$, both distributions are given by the chance move. Suppose the claim holds at t. By the definition of $\bar{\sigma}^{\mathsf{E}}$, the evader's move given (h_t, x_t) is distributed as $\bar{\sigma}^{\mathsf{E}}(\cdot \mid h_t, x_t)$ under σ^{E}, and by construction also under $\bar{\sigma}^{\mathsf{E}}$. The rest of the round, namely the reveal, the pursuers' joint action drawn from π^{P} given the public history, and the resulting positions, depends only on (h_t, x_t) and the evader's move. Hence (h_{t+1}, x_{t+1}) has the same distribution under both strategies. Terminal utilities depend only on the final public history and location, so the two strategies have the same expected utility. □

Two consequences are used in the paper. First, against pursuers who condition only on the public history, restricting the evader to strategies $\sigma^{\mathsf{E}}(\cdot \mid h, x)$ is without loss of generality. This is why evader nodes are keyed by (h, x) rather than by the whole trajectory, and below we take evader policies to be of this form. Second, the same cancellation shows that the belief $b(x \mid h)$ of Section 3 does not depend on the pursuers' public policy, so it is well defined given π^{E} alone.

A.2 Proof of Proposition 1

Throughout this proof, the parameters of both networks are fixed, as in Proposition 1. We call the random sweep orders, the joint actions sampled at pursuer nodes, the exploration noise at the roots and the interleaving of the parallel simulations the *randomness* of a search. As in our implementation, it is drawn independently of the evader's true location. The budget N is set before the search from public information and independent randomness. In particular, reduced visits use the root value averaged over all roots of the evader search. Root noise and forced playouts apply to every root alike (Appendix B.3).

Part (i). The legal joint actions at a pursuer node depend only on public information, because the evader's location never restricts the pursuers' moves. PUCT selection at the node reads only the network prior, which is computed from the public history, and the node's statistics, namely its weights and value sums. These statistics accumulate the backups of all simulations through the node, whatever their worlds. The world of the current simulation enters only after the selection, when the

result of the selected joint action, such as a capture, is decided. A search that keeps one tree per sampled world, such as AlphaZe** (Blüml et al., 2023), selects pursuer actions from per-world statistics and gives up exactly this property.

Part (ii), pursuer search. The search receives only the public history h_0. Its worlds, the possible locations in $B(h_0)$ at which the game is not already over, are a function of h_0. So are the weights in Equation 2, which depend on $q_{\mathsf{P}}(\cdot \mid h_0)$. Within a simulation, every step is determined by the statistics of the current node, the simulated world and the randomness. Selection reads only the statistics of the current node, and a new leaf is evaluated by f_{P} on public inputs, by f_{E} on the public history and the *simulated* location, or by the terminal utility of the simulated world. By induction over the order in which simulations are backed up, the whole search run, including the statistics of every node, is therefore a function of h_0 and the randomness. Its distribution, and hence the distribution of the returned policy, is the same for any two states with the same public history.

Part (ii), evader search. The roots are the live locations in $B(h_0)$, each with the same number $\lceil N/K \rceil$ of simulations, so both depend on h_0 only. The simulations from all roots follow the same steps as above, so the statistics of every node are again a function of h_0 and the randomness. The true location is used only after the search, to select which root is read. Since the statistics do not depend on it, part (i) implies that the choice at every pursuer node does not depend on it either. □

A.3 What the Searches Avoid and What Error Remains

Fix an evader policy $\pi^{\mathsf{E}}(\cdot \mid h, x)$, a root history h_0 and an action a of the player to move at h_0. For a pursuer continuation σ, let $Q^{\sigma}(x, a) \in [-1, 1]$ be the pursuers' expected utility when the evader is at x, the action a is played, and afterwards the pursuers follow σ and the evader follows π^{E}. Let Σ_{pub} contain the continuations that depend only on the public history and $\Sigma_{\mathrm{all}} \supseteq \Sigma_{\mathrm{pub}}$ those that may also depend on the evader's locations. When the pursuers are to move at h_0, let $b(x) = \Pr(\text{the evader is at } x \mid h_0; \pi^{\mathsf{E}})$ be the belief over $B(h_0)$.

The two failures. Both failures replace the pursuers' public continuations by ones that see the evader's location. Since $\Sigma_{\mathrm{pub}} \subseteq \Sigma_{\mathrm{all}}$, every $\sigma \in \Sigma_{\mathrm{pub}}$ satisfies $Q^{\sigma}(x, a) \le \max_{\sigma' \in \Sigma_{\mathrm{all}}} Q^{\sigma'}(x, a)$ in every world x. Averaging over b and maximizing over σ gives

$$\sum_x b(x) \max_{\sigma \in \Sigma_{\mathrm{all}}} Q^{\sigma}(x, a) \;\ge\; \max_{\sigma \in \Sigma_{\mathrm{pub}}} \sum_x b(x)\, Q^{\sigma}(x, a). \tag{4}$$

A pursuer search that plans a separate continuation in each world estimates the left side of Equation 4, whereas the pursuers can achieve at most the right side. This gap is the over-optimism of strategy fusion. Conversely, an evader whose imagined pursuers may condition on its location values each world x by $\max_{\sigma \in \Sigma_{\mathrm{all}}} Q^{\sigma}(x, a)$, which no public continuation of the real pursuers can exceed. This gap is the over-pessimism of an opponent model in which the pursuers see the evader. Both gaps can be strictly positive, because a public continuation must act the same in every world that the pursuers cannot tell apart. By Proposition 1, neither gap is built into our searches. The pursuer nodes are still not independent of the worlds, however: the value of a pursuer node depends on how often each world reaches it, which the following proposition quantifies.

Error decomposition. Let the pursuers be to move at h_0, and let $\sigma^{\star}$ be a best public continuation after the root action a, that is, one that maximizes $\sum_x b(x) Q^{\sigma}(x, a)$ over $\sigma \in \Sigma_{\mathrm{pub}}$. In the pursuer search, let $\rho_a(x)$ be the share of the accumulated weight on the root edge a that comes from world x. Let $\hat{Q}_x(a)$ be the mean value of the world-x simulations through a (arbitrary if there are none). Since all simulations of one world carry the same weight $w(x)$, the search estimate is $\hat{Q}(a) = \sum_x \rho_a(x) \hat{Q}_x(a)$.

Proposition 2. *For every root action a of the pursuer search,*

$$\hat{Q}(a) - \sum_x b(x)\, Q^{\sigma^{\star}}(x, a) = \underbrace{\sum_x \big(\rho_a(x) - b(x)\big)\, Q^{\sigma^{\star}}(x, a)}_{\textit{world distribution}} + \underbrace{\sum_x \rho_a(x) \big(\hat{Q}_x(a) - Q^{\sigma^{\star}}(x, a)\big)}_{\textit{estimation within worlds}}, \tag{5}$$

and the first term is at most $\|\rho_a - b\|_1$ in absolute value.

Proof. Adding and subtracting $\sum_x \rho_a(x) Q^{\sigma^\star}(x,a)$ in $\hat{Q}(a) = \sum_x \rho_a(x) \hat{Q}_x(a)$ gives Equation 5. For the bound, $\left|\sum_x (\rho_a(x) - b(x)) Q^{\sigma^\star}(x,a)\right| \le \max_x |Q^{\sigma^\star}(x,a)| \, \|\rho_a - b\|_1 \le \|\rho_a - b\|_1$, because utilities lie in $[-1, 1]$. □

The estimation term collects the finite budget, the errors of the value network, and the suboptimality of the policies played inside the search relative to $\sigma^\star$ and π^{E}, including the mixture of worlds at deeper pursuer nodes. The location prior acts on the world-distribution term. After complete sweeps, every world has been simulated equally often. Since the prior is normalized over the K live worlds, the share of the total root weight that comes from world x is then exactly $w(x)/K = \rho(x)$, where $\rho = (1-\beta)u + \beta\, q_{\mathsf{P}}(\cdot \mid h_0)$ and u is uniform on the K worlds. The shares ρ_a on an individual root edge stay close to ρ as long as the choice of the root edge does not depend systematically on the world. The edge of a simulation is chosen from statistics of earlier simulations, while its world comes from a random sweep. The two are therefore independent apart from the dependence within a sweep. Only the finite interleaving of worlds and edges can leave small imbalances, so this step is heuristic. With $\rho_a \approx \rho$, the world-distribution term is governed by

$$\|\rho - b\|_1 \;\le\; (1-\beta)\, \|u - b\|_1 + \beta\, \|q_{\mathsf{P}}(\cdot \mid h_0) - b\|_1, \tag{6}$$

which follows from $\rho - b = (1-\beta)(u-b) + \beta(q_{\mathsf{P}} - b)$ and the triangle inequality. Since $\rho = u$ for $\beta = 0$, Equation 6 shows that for $\beta > 0$ the prior brings ρ strictly closer to b than uniform weighting whenever $\|q_{\mathsf{P}}(\cdot \mid h_0) - b\|_1 < \|u - b\|_1$. We expect this condition against the self-play evader that the prior was trained on, while the weight $1 - \beta$ on u limits the damage against other evaders. The term vanishes in every reveal round, where $B(h)$ collapses to the true location. Suppose instead that the root worlds were drawn from b, the evader inside the search sampled its moves from π^{E}, every pursuer node expanded all its joint actions and selection used UCB1. The pursuer search would then be a POMCP search (Silver & Veness, 2010), with network evaluations in place of rollouts, on the partially observable decision problem that π^{E} induces for the pursuers. Its estimate for the root action a would converge to $\sum_x b(x) Q^{\sigma^\star}(x,a)$ as the budget grows. The two terms of Equation 5 measure how far our search is from these conditions. These results are local: they concern one search for fixed networks, not the training dynamics.

B Implementation Details

B.1 Rules

Our environment follows the classic 1985 Milton Bradley edition, with the negotiable move order of the 2023 Ravensburger rulebook. Table 2 compares it with the classic rules. Two further conventions matter for the formal model. Capture is checked only after the whole pursuer team has moved. Since the evader cannot enter an occupied station, only a pursuer move can capture. No player may stay in place voluntarily: a pursuer with no legal move passes, and an evader with no legal move loses.

Table 2: Rules of our environment compared with the classic edition.

Kept rule	Classic edition	Our environment
Map	199 stations; taxi, bus, underground, ferry	same
Players	five pursuers against one evader	same
Pursuer tickets	10 taxi, 8 bus, 4 underground each	same
Evader's black tickets	5	same
Start positions	18 start cards	same
Changed rule	**Classic edition**	**Our environment**
Evader's regular tickets	limited, refilled with the tickets the pursuers use	unlimited
Double-move tickets	available	none
Pursuers' move order	fixed seating order	chosen by the team each round
Evader wins early if	all pursuers are out of tickets	no pursuer can move
Rounds; reveal rounds	24; 3, 8, 13, 18, 24	22; 3, 8, 13, 18

B.2 MODEL

Both networks encode the state with the same kind of relational graph network over the 199 stations and differ in their inputs and output heads. We describe the inputs, the shared encoder, and the heads in turn.

Inputs. Every station v carries a feature vector s_v, and the whole state carries a global vector g (Table 3). The station features place each pursuer and its remaining tickets on the station it occupies, mark the revealed locations and the possible locations $B(h)$, and describe the station's connections. The global vector holds the evader's ticket history, which is the only record of its hidden moves. The pursuer network receives exactly these public features. The evader network additionally receives a one-hot channel for its current location. Station features are given for every station, and per-pursuer features keep the pursuers' identities. Only the evader network receives the last station feature. Remaining tickets are divided by the initial stock; the unlimited regular tickets of the evader are encoded with a large stock, so their features stay near one.

Table 3: Input features.

Group	Feature	Size
Station features s_v	pursuer i is at v (one per pursuer)	5
	remaining taxi, bus, underground tickets of the pursuer at v (per pursuer)	15
	v was revealed in reveal round j (one per reveal round)	4
	$v \in B(h)$	1
	valid station; number of taxi, bus, underground, ferry edges at v	5
	evader is at v (*evader network only*)	1
Global features g	evader's ticket in each round (none, taxi, bus, underground, black)	22×5
	evader's remaining taxi, bus, underground, black tickets	4
	round $t/22$	1
	asymmetric playouts: active flag and signed budget ratio (Appendix B.3)	2

Encoder. A linear layer maps each station's features, together with the broadcast global vector, to a hidden vector of width 128, $e_v^{(0)} = \mathrm{SiLU}(W_{\mathrm{in}} s_v + W_g g)$. Three message-passing layers then propagate information along the map, with separate weights for the four transport types $r \in \{\text{taxi}, \text{bus}, \text{underground}, \text{ferry}\}$, where the ferry edges are those that only black tickets can use:

$$e_v^{(\ell+1)} = \mathrm{LayerNorm}\Big(e_v^{(\ell)} + \mathrm{SiLU}\Big(W_0^{(\ell)} e_v^{(\ell)} + \sum_r W_r^{(\ell)} \frac{1}{|\mathcal{N}_r(v)|} \sum_{u \in \mathcal{N}_r(v)} e_u^{(\ell)}\Big)\Big),$$

where $\mathcal{N}_r(v)$ contains the stations reachable from v along an edge of type r (the sum is zero if there is none). After three layers, the feature of a station summarizes its neighborhood within three moves. A state vector $z = \mathrm{SiLU}(W_z \frac{1}{199} \sum_v e_v^{(3)})$ is obtained by mean pooling. The heads read z and the final station features $e_v = e_v^{(3)}$. The graph is encoded once per decision.

Pursuer heads. The *order head*, a two-layer perceptron on z, scores all $5! = 120$ move orders; orders that are not canonical are masked before the softmax. Two pursuers *interfere* if one of them can reach the other's station or their possible destinations overlap. Two orders are equivalent if they agree on the relative position of every interfering pair. Each class is represented by the order that is lexicographically smallest in the ranking of the pursuers' current stations, so that the choice does not depend on how the pursuers are labeled. The *move head* is a recurrent decoder. Its state is initialized from z and a learned embedding of the chosen order. At step k, a GRU cell updates the state from the station feature of the k-th pursuer in the order and the features of the previous move (its origin, its destination and its ticket). A two-layer perceptron then scores every candidate move (y, τ) of this pursuer, a destination y with a ticket τ, from the state, e_y and τ. Destinations that are occupied at that step, including those just taken by earlier pursuers, are masked, and a pursuer without a legal move passes. The *value head* is linear in z and is regressed onto ± 1 without squashing. The *location-prior head* is a linear layer applied to every station feature e_v. A softmax over the possible locations gives $q_{\mathsf{P}}(\cdot \mid h)$.

Evader heads. A two-layer perceptron scores each legal move (y, τ) of the evader from z, e_y and τ. Moves to the same destination with a regular and with a black ticket are separate moves. The value head is linear in z.

Size and variants. The pursuer network has 0.58M parameters and the evader network 0.32M. Within a role, the policy and the value share the encoder. The two roles share no parameters. During training, every example is relabeled by a random permutation of the pursuers' identities, since the pursuers are interchangeable. The ablation SWIFT w/o belief weighting (MLP) replaces the graph encoder by a multilayer perceptron. Stations and tickets are embedded with learned vectors of size 16 and 4, and the embedded state is flattened into one vector and passed through one pre-activation residual block of width 304. The heads read the station embeddings in place of e_v. Its networks have 0.53M and 0.38M parameters, 0.91M in total against 0.90M for the graph networks.

B.3 Hyperparameters

Table 4 lists the settings of SWIFT; an arrow marks a temperature that decays between the two values over a game. The ablation studies use the same settings, with three exceptions. SWIFT w/o belief weighting has neither the location-prior head nor the weighting in Equation 2, and SWIFT w/ auxiliary belief loss trains the head but does not use the weighting. SWIFT w/o belief weighting (MLP) is SWIFT w/o belief weighting with the MLP encoder of Appendix B.2 and 16 instead of 8 CPU workers for self-play, so its wall-clock curves compare the whole configuration.

Table 4: Hyperparameters of SWIFT.

Search (both roles)	
Simulations per search N	300
World schedule	$\lceil N/K \rceil$ sweeps
PUCT constant c	1.5
Joint actions per pursuer node	≤ 64
Parallel simulations (virtual loss)	8 (1.0)
First-play urgency reduction	0.2 (root: 0)
Network policy temperature	1.1
Root policy temperature	$1.35 \rightarrow 1.15$
Move selection temperature	$0.75 \rightarrow 0.15$
Exploration and targets	
Shaped Dirichlet noise weight	0.25
Dirichlet concentration (P / E)	1.92 / 0.39
Forced playouts coefficient	2.0
LCB target (std. dev., min. share)	5, 0.15
Location prior	
Backup mixture β	0.5
Loss weight λ	0.1

Self-play	
Games per iteration	4096
CPU workers	8
Reduced visits (threshold, min. sims.)	0.9, 50
Weight of reduced-visit examples	0.1
Asymmetric playouts (prob., ratio)	0.01, ≤ 8
Side positions (prob.)	0.04
Surprise weighting (policy / value)	0.5 / 0.1
Random-evaluator warm-up	2 iterations
Optimization	
Optimizer	AdamW
Learning rate	10^{-4}
Weight decay	10^{-4}
Batch size	128
Value loss weight	1.0
Gradient clipping (norm)	5.0
Precision (train / infer.)	bf16 / fp16
Wall-clock budget	5 hours

Most of the search and self-play settings are refinements from KataGo (Wu, 2019), adapted to our two searches. *First-play urgency* gives unvisited edges an initial value slightly below the parent's. *Forced playouts* give every root action a minimum number of visits, and these extra visits are pruned from the policy target (Section 4.4). The *LCB* adjustment raises the target weight of the action whose value has the best lower confidence bound, among actions with enough visits. *Reduced visits* lower the budget, and the training weight of the resulting examples, once the root values of the last three decisions consistently predict the same winner. For the evader, the root value is averaged uniformly over all roots, so the true location does not influence the budget. *Asymmetric playouts* occasionally give the two sides different budgets, and the networks receive the budget ratio as input. *Side positions* branch off counterfactual positions from real games, at most once per game, to widen the coverage of the training data. *Surprise weighting* replays examples more often when the search target differs strongly from the network's prediction. During the first two iterations, a random evaluator replaces the networks to produce the initial data. Every root of the evader search receives its own shaped Dirichlet noise and forced playouts, so none of these refinements favors the true root. Playout cap randomization and tree reuse are disabled.

C Experimental Details

Evaluation protocol. A pursuer wins a game if it captures the evader or the evader has no legal move. All evaluations, including the ratings, play the networks without search. Each player chooses the most probable legal action of its policy. Each pairing plays 512 games. All pairings of one evaluation share the same sequence of start positions, drawn from the 18 classic start stations as in training. Table 1 reports mean win rates. With 512 games, the half-width of a 95% Wilson interval is at most 4.3 percentage points.

Rating. Figures 2 and 3 rate networks by cross-play, each with its own pool. The pool of Figure 2 contains the checkpoints of every method, taken along training, together with the evader baselines. The pool of Figure 3 contains the checkpoints of SWIFT and its ablations, including those of the eight-hour runs. Every pursuer checkpoint plays 512 games against every evader in its pool. For pursuer i and evader j with n_{ij} games and w_{ij} pursuer wins, a Bradley–Terry model predicts the pursuer's winning probability $p_{ij} = \sigma(s_i - s_j)$, where σ is the logistic function and each network has one strength s. A stronger pursuer captures more often, and a stronger evader escapes more often. We compute the maximum a posteriori estimate under independent standard normal priors, that is, we minimize

$$\mathcal{L}(s) \;=\; -\sum_{(i,j)} \Big[w_{ij} \log p_{ij} + (n_{ij} - w_{ij}) \log(1 - p_{ij}) \Big] \;+\; \frac{1}{2} \sum_{k} s_k^2 \tag{7}$$

by Newton's method with backtracking line search. The prior shrinks estimates based on few games toward zero and keeps ratings finite when a pairing is won or lost entirely. We report $R_k = 1500 + \frac{400}{\ln 10}\, s_k$, so that a pursuer rated 400 points above an evader is predicted to win 91% of games. The value 1500 is the center of the scale, not a fixed reference policy. Pursuers and evaders are ranked separately and compared within one role through their common opponents. Ratings depend on the pool, so they are comparable only within one figure. A single rating also cannot express non-transitive strengths, which is why we also report the win-rate matrix (Table 1).

PPO-GNN and PPO-MLP. Both baselines model the five pursuers as one agent. The actor sees the public observation and samples a move order from all 120 orders and then the pursuers' moves autoregressively. The critic additionally sees the evader's location. Both are trained with PPO against the Greedy evader, with 4096 parallel environments, a horizon of 128 steps, 2 epochs per update, minibatches of 1024 team transitions, learning rate 3×10^{-4}, discount 0.99, GAE parameter 0.95, clipping 0.2 and entropy coefficient 0.01. The reward is ± 1 at the end of the game and 0 otherwise. PPO-GNN uses a graph encoder of the same size as SWIFT, and PPO-MLP uses a residual block of width 304. Both train for five hours.

Other baselines. MAPPO-GNN (Yu et al., 2022) applies one actor with shared parameters to all five pursuers. The actor sees only public information, encoded by a relational graph network with three layers of width 128. It scores the moves of every pursuer from the same pre-move state, and the moves are then taken in a fixed order, each renormalized over the stations that earlier pursuers left free. A centralized critic with a separate graph encoder of the same size additionally sees the evader's location. The actor and the critic have about 349K and 282K parameters. MAPPO-GNN is trained for five hours against the Greedy evader with the PPO settings of PPO-GNN and PPO-MLP, a value-loss coefficient of 0.5 and early stopping at a KL divergence of 0.03. P2SRO (McAleer et al., 2020) trains pursuer and evader populations with PPO-GNN best-response oracles and computes the meta-strategy by projected replicator dynamics (Lanctot et al., 2017). It trains for twelve hours so that the populations converge, which yields about ten policies per population. We evaluate the meta-strategy of each role. AlphaZe** (Blüml et al., 2023) is adapted to our joint action and our two networks. It uses the networks, the training pipeline and the training time of SWIFT and differs in the search and the KataGo refinements. At a pursuer decision, it draws up to five possible locations uniformly without replacement and builds one tree per sampled world, in which the evader's location is known. It runs 100 simulations in each tree and averages the normalized root visit distributions. Inside a tree, the pursuer network receives the sampled location as its only possible location. The evader searches a single tree from its true location with 100 simulations, so the pursuers it imagines see it. AlphaZe** uses no location prior and, like the original method, none of the KataGo refinements that SWIFT uses (forced playouts, LCB policy targets, policy-target pruning and side positions; Appendix B.3).

Weak evaders. Greedy chooses, among its legal destinations, one that maximizes the graph distance to the nearest pursuer, breaking ties uniformly. Territory ranks destinations lexicographically

by whether the nearest pursuer is at least two moves away, by the number of stations that the evader reaches before every pursuer, and by the sorted distances to the pursuers. Both use black tickets on non-reveal rounds, with a probability that spreads the remaining black tickets over the remaining non-reveal rounds. In reveal rounds, they prefer regular tickets. Random picks a legal move uniformly.

D Additional Results

D.1 Human Evaluation

We conduct an exploratory human evaluation using the web interface described in Appendix E. One author familiar with the game controlled the evader against five SWIFT pursuers over 40 games, organized into two blocks of 20. We evaluated checkpoints from training iterations 2, 8, 16, 32, and 118, using deterministic network policies without test-time search. Checkpoint identities were hidden from the participant. Start positions for the evader and pursuers were sampled without replacement from the 18 classic start stations. The game used a 22-round limit, unlimited regular evader tickets, five black tickets, and no double moves. The participant selected legal destinations and tickets and could undo rounds without limit, including after capture. Each opening counted as one game, ending when the participant achieved an evader victory or conceded; undoing moves did not create additional games. Table 5 reports final human success rates, mean game lengths, and mean undo counts; game length counts the evader moves of the final trajectory, and all means include conceded games. The participant won 30 of 40 games, but none of the four games against checkpoint 118. These results characterize one participant's performance with unlimited undo and should not be interpreted as population-level human performance or performance without undo.

Table 5: Results of games between a human evader and the SWIFT pursuers.

Checkpoint	Games	Human wins	Success (%)	Mean rounds	Mean undos
2	4	4	100.0	21.00	0.75
8	10	10	100.0	21.60	3.50
16	14	10	71.4	19.79	6.14
32	8	6	75.0	18.63	3.50
118	4	0	0.0	7.75	7.75
Total	40	30	75.0	18.93	4.58

D.2 Win Rates against Fixed Evaders during Training

Figure 5 complements the ratings of Figure 2 with win rates against four fixed evaders. The three PPO-type baselines, trained against Greedy, beat it in nearly all games from their first checkpoint on, which is why their ratings level off early. SWIFT needs about two hours to beat Greedy and about three hours to beat Territory in nearly all games. Against the final evader of SWIFT, its pursuer rises from about 10% to 98%. MAPPO-GNN and PPO-GNN stay between about 50% and 77%, and P2SRO and AlphaZe** reach about 61% and 26% after five hours. The final P2SRO model, trained for twelve hours, wins 85.7% (Table 1).

D.3 Effect of the Search Budget at Test Time

Figure 6 shows how search at test time changes the strength of SWIFT. We use the checkpoint after three hours of training, because the final networks already win nearly all games without search, which leaves no room to measure a gain. At each point, one side searches with the given number of simulations per decision and plays the action with the largest visit weight, without root noise. The other side plays the most probable action of its network, and every point uses 512 games from the shared start positions. Search helps the pursuers substantially: their win rate rises from 53% without search to 77% with 200 simulations and 85% with 400. It barely helps the evader, whose win rate stays between about 46% and 48%. This asymmetry is consistent with how the two searches spend their budget (Section 4.3). Every pursuer simulation updates the single public root. The evader, in contrast, divides its simulations equally among the K roots, so the root of its true location receives

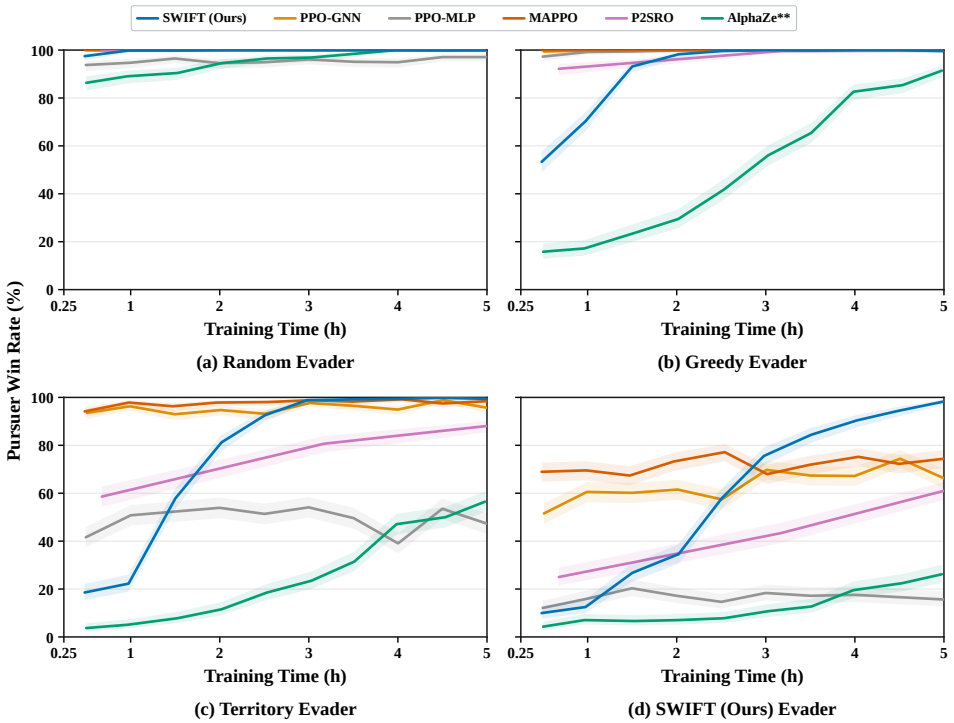


Figure 5: Win rate of each method's pursuer checkpoints, played without search, against three weak evaders and the final evader of SWIFT. Shaded bands: 95% Wilson intervals over 512 games. The P2SRO curves show the first five of its twelve training hours.

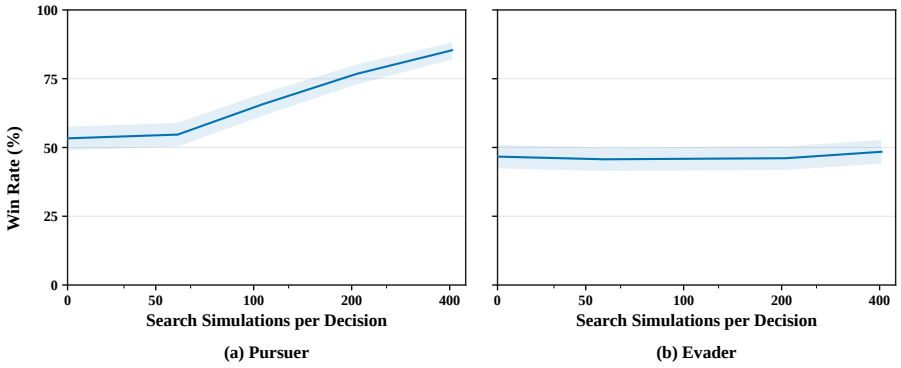


Figure 6: Win rate of a SWIFT checkpoint after three hours of training when one side searches with the given number of simulations per decision and the other plays its network without search. (a) Searching pursuers. (b) Searching evader. Shaded bands: 95% Wilson intervals over 512 games.

only about N/K of them. The evader search pays this price to keep its imagined pursuers from seeing it.

D.4 Location Prior

Figure 7 follows the possible locations and the location prior through the first three rounds of one game. At the start (a), the evader may be at any of the 13 classic start stations not occupied by a pursuer, and the figure shows only these possible locations. After an underground move (b), three locations remain, and the prior puts 54.7% on the true one, against 33.3% under uniform weighting. After a bus move (c), ten locations remain, and the true one again ranks first, with 34.1% against

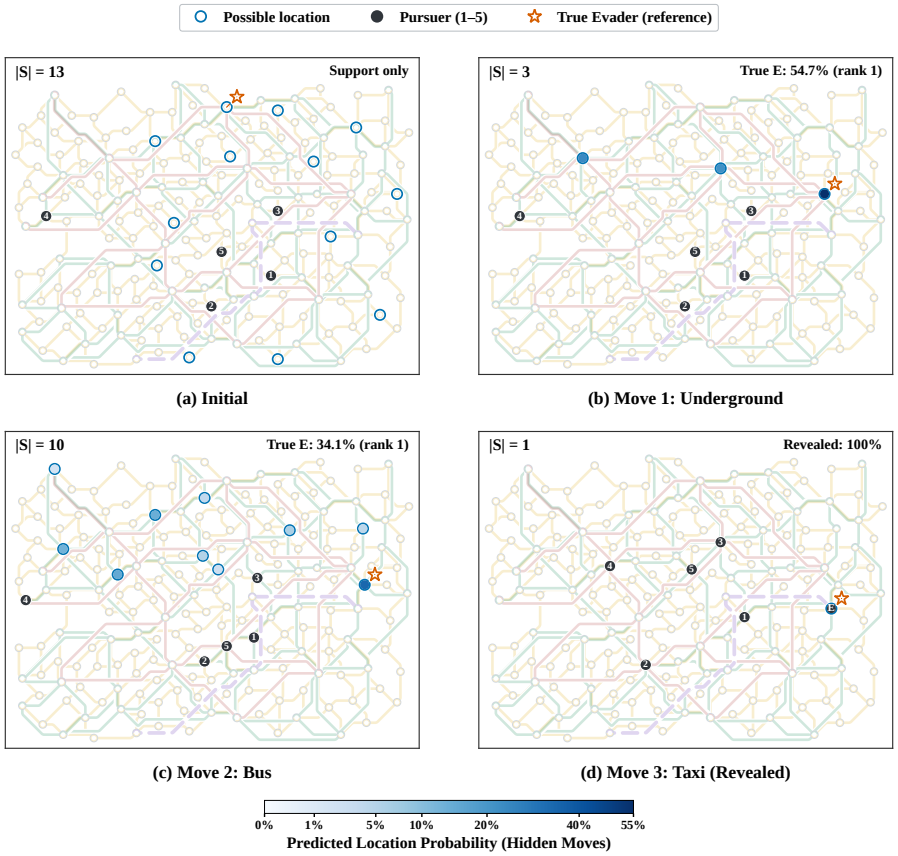


Figure 7: Possible evader locations and the location prior of the final SWIFT pursuer network during the first three rounds of one game. Color: prior probability on hidden moves; star: true evader location, shown for reference only.

10%. Round 3 (d) is a reveal round, so after the taxi move the location becomes public. The prior thus shifts the pursuers' search toward the true world, while the weight in Equation 2 keeps every possible location at least half of its uniform weight.

D.5 Number of Pursuers

The standard game gives the pursuers a structural advantage (Nijssen & Winands, 2012). To see how much of it comes from their number, we train SWIFT with three pursuers instead of five for eight hours. All other rules stay the same, so each pursuer keeps 10 taxi, 8 bus and 4 underground tickets, and the joint action chooses among $3! = 6$ move orders. All other settings follow Table 4. Figure 8 shows the win rates of its pursuer checkpoints, evaluated as in Appendix C.

With three pursuers, the game shifts toward the evader. The pursuers still beat Random in nearly all games after one hour and Greedy in about 95% of games after four hours. They learn much more slowly than with five pursuers, however, where SWIFT beats Greedy and Territory in nearly all games within about three hours (Appendix D.2). After eight hours, they win about 82% of games against Territory, against 99.4% with five pursuers (Table 1). Against the final evader of their own run, they win only about 23%, against 98.2% in the five-pursuer game. Both curves are still rising, from about 74% and 11% after five hours, so longer training would help the pursuers further. Even so, removing two pursuers turns a game that the learned pursuers win almost always into one that the learned evader wins in most games. The case study below suggests why. There, the capture follows from covering every station the evader could move to (Appendix D.6), and three pursuers can cover fewer stations per round than five.

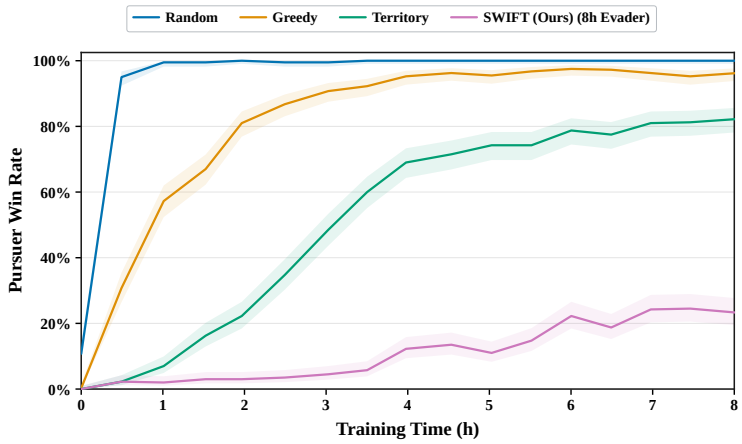


Figure 8: Win rate of the SWIFT pursuer checkpoints in the game with three pursuers, played without search, against three weak evaders and the evader of the same run after eight hours. Shaded bands: 95% Wilson intervals over 512 games.

D.6 Case Study

We examine four complete games against the final evader of SWIFT, with identical start positions, tickets, player identities and rules (Figures 9–12). The evader starts at station 132 and pursuers P1–P5 at 50, 174, 34, 53 and 94. All policies choose the most probable legal action of their networks, without search. The evader network is the same in all four games but responds to each pursuer's moves, so the games diverge after the first round. For P2SRO, we use a fixed pursuer policy from its population rather than the meta-strategy. The checkpoints also differ in training time, so this is not an equal-budget comparison. SWIFT captures the evader in round 11, whereas MAPPO-GNN, P2SRO and AlphaZe** lose in rounds 22, 21 and 22. In all three losses the pursuer team is left without a legal move. Below, we call the possible locations B_t^+ of a round (Section 3) its *candidates*. A candidate is *reachable* if some pursuer can move there in one step with its remaining tickets.

SWIFT: eliminating candidates while closing the exits. After the reveal at station 194 in round 8, the evader takes a taxi, and the candidates of round 9 are $\{192, 193, 195\}$. P3 and P5 move to 192 and 193 without capture, which leaves 195, while P2 moves to 197 toward the other side. In round 10, the evader spends its last black ticket. Of the candidates $\{182, 194\}$, P3 checks 194, which leaves 182, while P2 and P5 reach 196 and 181. At the start of round 11, the evader has no black ticket and only two legal moves, taxi to 183 or to 195, and both lead to the same public history. The joint action of the pursuers covers both: P2 moves to 183 and P3 to 195. Replaying both evader moves confirms that the recorded joint action is legal and captures in either case, so the capture follows from complete coverage rather than from guessing the hidden location. Earlier moves that did not capture still paid off, because each removed a candidate while the other pursuers moved toward the next exits.

MAPPO-GNN: reveals do not turn into reachable coverage. The evader makes 11 consecutive bus moves in rounds 10–20. After the reveal in round 13, none of the round-14 candidates can be reached in one step with the pursuers' tickets. In round 19, only station 72 among the candidates $\{7, 29, 72\}$ is reachable, but P5 moves from 91 to 105 instead. Moving to 72 would have been legal and would have removed a candidate, although it could not have captured the evader at the unreachable station 29. In round 20 no candidate is covered either, although some are reachable. The team runs out of bus tickets after round 19 and finally loses all legal moves. Since the evader's regular tickets are unlimited and the pursuers' are not, a long chase exposes this asymmetry.

P2SRO: pursuers become stranded and the chase lags behind. After round 14, P2 is stranded at station 192 without taxi tickets. It still holds six bus and two underground tickets, but 192 has only taxi connections. More pursuers become stranded later. In rounds 20 and 21, P5 reaches stations 115 and 114, each one round after the evader left it. When the game ends, the team still holds 11

bus and six underground tickets, none usable where their holders stand. The failure is therefore a mismatch between tickets and positions, not the exhaustion of every ticket.

AlphaZe: the reachable candidates are covered, but the rest are out of reach.** After round 18, P4 and P5 are stranded at stations 142 and 58, each holding four underground tickets but no usable connection. In round 19, the evader moves by underground from the revealed station 153 to 163, and the candidates are $\{111, 140, 163, 185\}$. The pursuers cover both reachable candidates, 111 and 140, but no pursuer can reach 163 or 185. The evader then escapes through 146, 147 and 164. This loss is not a wrong choice among the reachable candidates in round 19. The earlier play had already left too little mobility to cover the remaining exits.

Together, the four games show that capturing requires turning information about the hidden location into moves that cover it. SWIFT achieves this in the selected game, whereas the baselines fail at different points: reaching the candidates, keeping pursuers mobile, or covering every exit. The games were selected to illustrate divergent outcomes. They complement the aggregate results of Section 5 but do not by themselves establish superiority or the effect of any component.

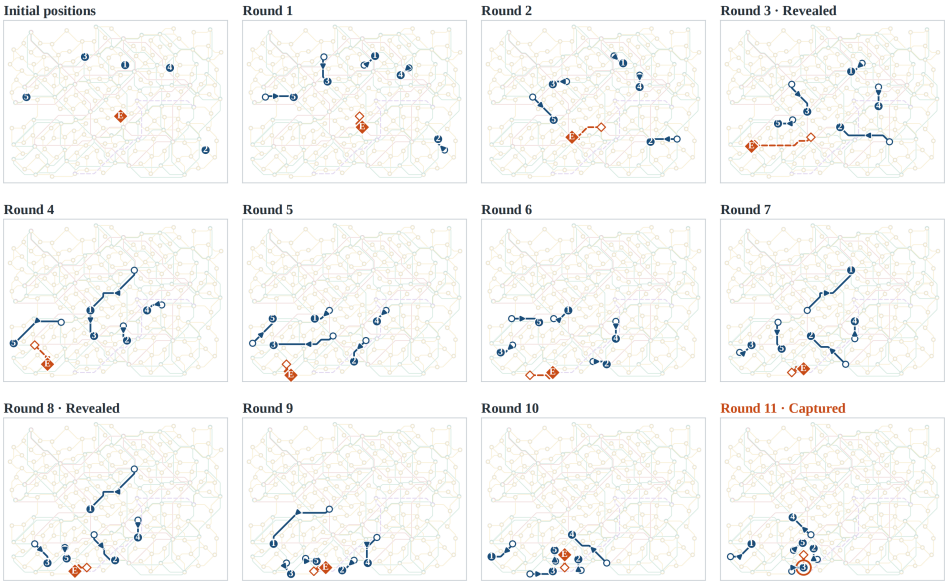


Figure 9: Complete game of the SWIFT pursuers against the SWIFT evader: the initial state and rounds 1–11 in reading order. SWIFT eliminates candidate locations one after another and captures the evader in round 11. Navy circles 1–5 are pursuers and the orange diamond is the evader; hollow markers are the origins of moves, arrows show the moves of each round, and the orange ring marks the capture. "Revealed" marks a public reveal. Hidden evader positions are shown for the reader only and are not available to the pursuers.

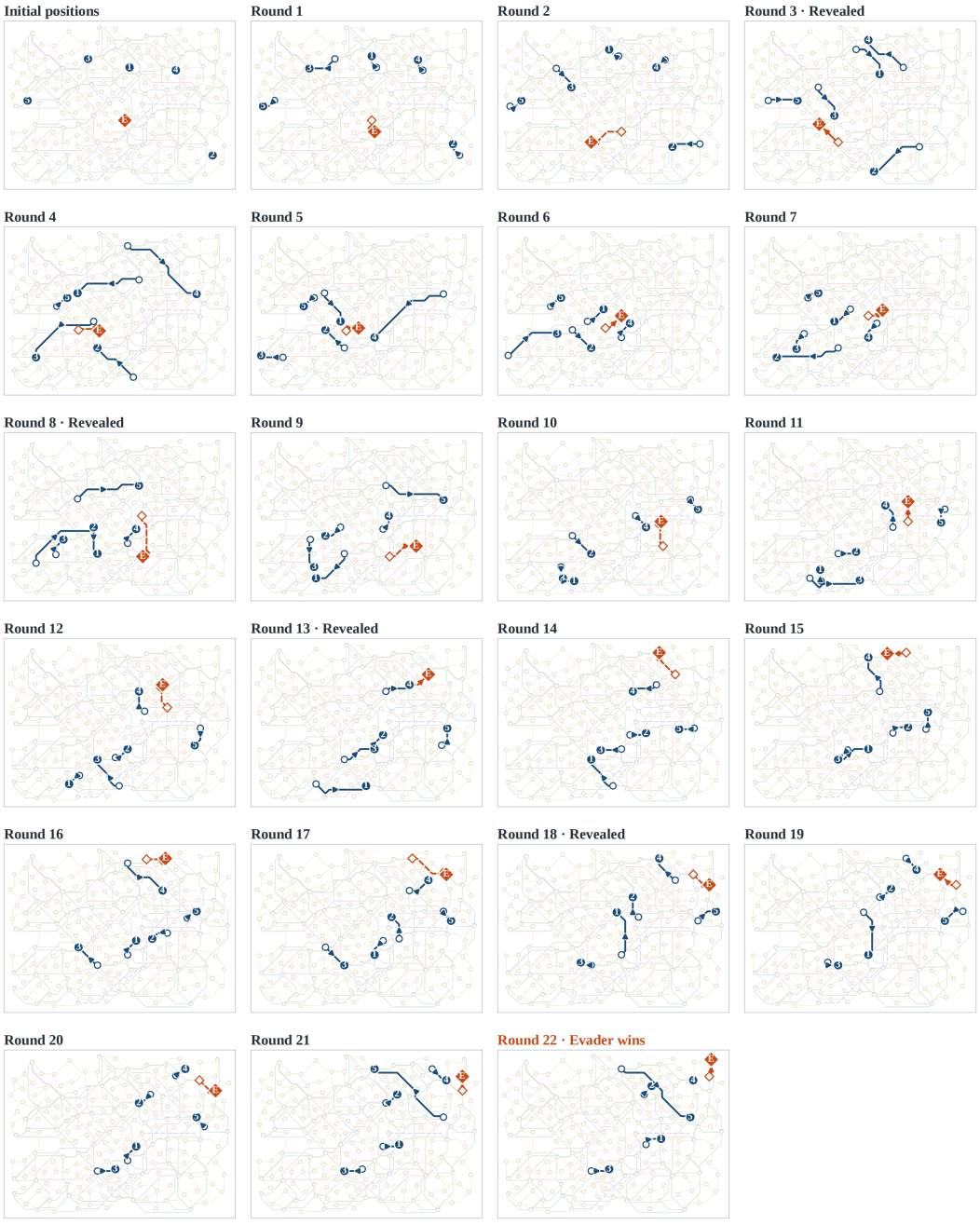


Figure 10: Complete game of MAPPO-GNN pursuers against the SWIFT evader: the initial state and rounds 1–22. The evader makes consecutive bus moves in rounds 10–20, remains uncaptured, and wins in round 22 when the pursuers have no legal move. Start positions, evader network and visual conventions as in Figure 9; the trajectories develop independently.



Figure 11: Complete game of P2SRO pursuers against the SWIFT evader: the initial state and rounds 1–21. P2 cannot move after round 14; in rounds 20 and 21, P5 reaches the evader's location of the previous round. The evader wins in round 21 when no pursuer can move, although 17 tickets remain that their holders cannot use at their stations. Conventions as in Figure 9.

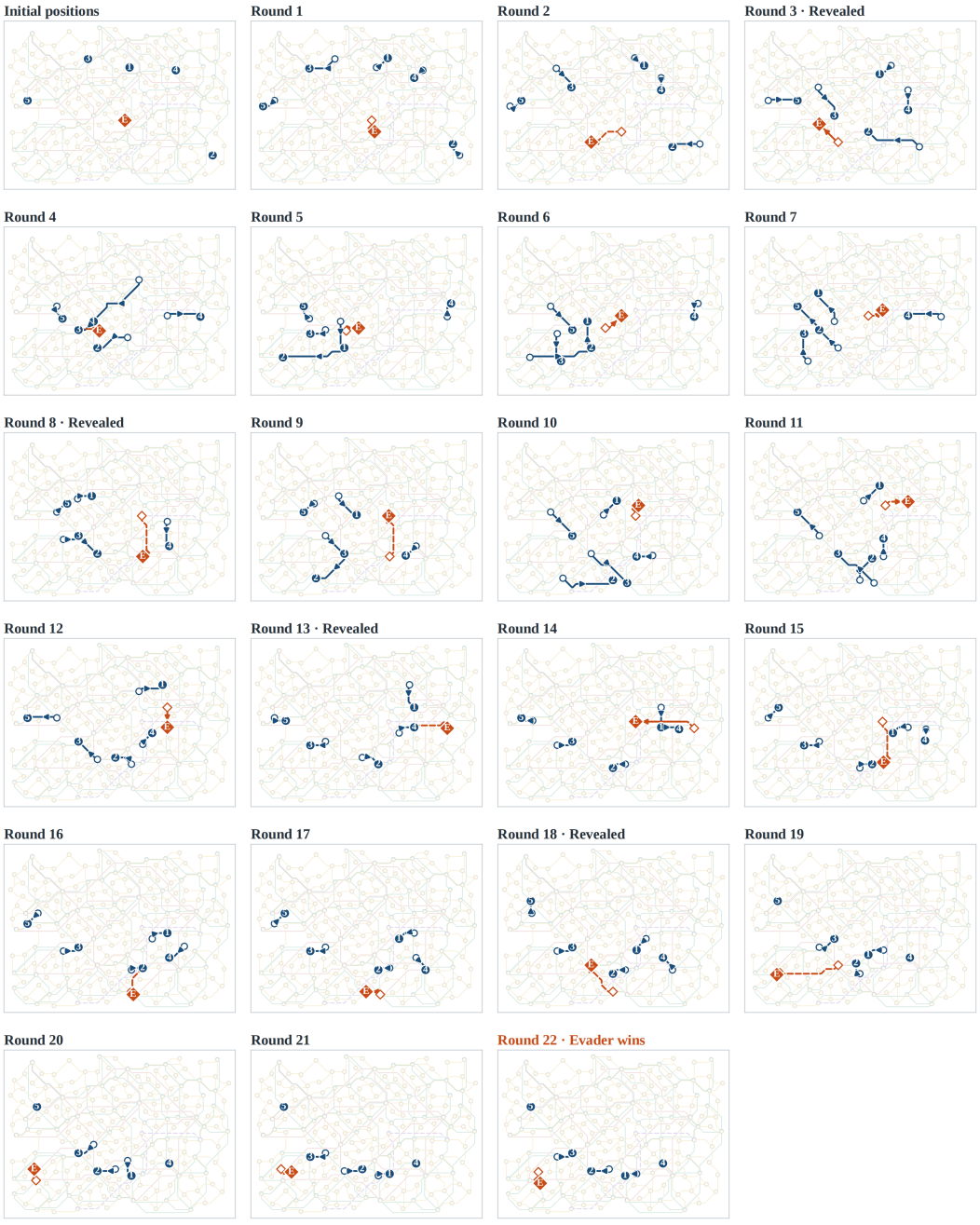


Figure 12: Complete game of AlphaZe** pursuers against the SWIFT evader: the initial state and rounds 1–22. After the reveal at station 153 in round 18, the evader moves by underground to 163, beyond the pursuers' one-step reach, and then visits 146, 147 and 164. It wins in round 22 when the pursuers have no legal move. Conventions as in Figure 9.

E WEB INTERFACE

We provide a web-based interface for interactive play against our trained agents. Users control the evader while SWIFT controls the pursuers. The interface displays the game state, the remaining tickets and the legal moves, so that users can explore the agents' behavior through direct interaction. Figure 13 shows the interface. Since the current deployment contains identifying information, we omit the link to preserve anonymity during double-blind review. We plan to make the interface publicly available once the anonymity restrictions are lifted.

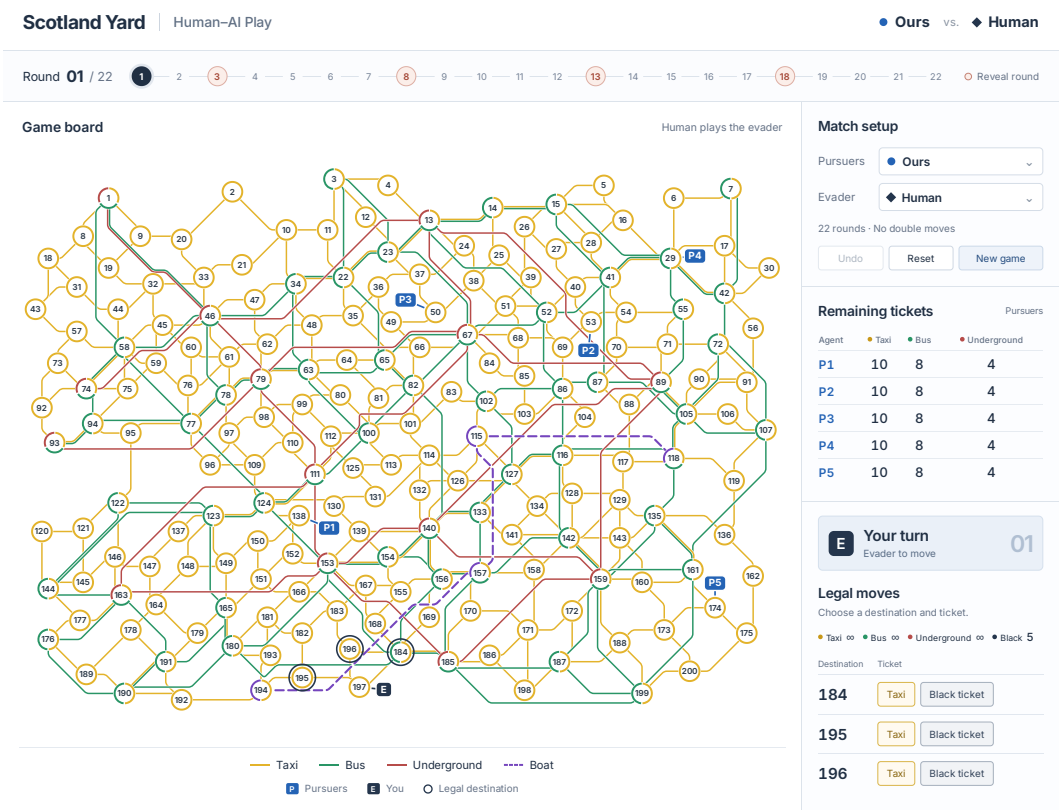


Figure 13: Web interface for playing the evader against the pursuers of SWIFT. The board shows the pursuers (P1–P5), the evader controlled by the user (E) and its legal destinations; the side panel shows the match setup, the remaining tickets and the legal moves with their tickets.