CONTEXT-AWARE LLM AGENTS AS WORLD-TO-GAME BRIDGES FOR SOLVING OPEN GAMES

Anonymous authors
Paper under double-blind review

ABSTRACT

Game theory underpins many high-stakes applications—security, defense, resource allocation, auctions, and negotiation—yet practical deployment is blocked by a fundamental tension: models faithful enough to capture reality are typically intractable, while models simple enough to solve discard the information that actually matters. In real deployments the game is also dynamic and open: its parameters change over time, and the decisive information arrives as unstructured, semantic, and drifting context (traffic reports, road closures, observed trajectories, market signals). Existing solvers—equilibrium algorithms, deep RL, or pure LLM-generated policies—cannot consume such context, because they assume a fully specified, fixed, numeric game. We introduce the Context-Aware Game-Theoretic Agent (CAGTA), a general framework that resolves this tension. An LLM acts solely as a *world-to-game bridge*: it interprets the unstructured context stream and maps it into the numeric parameters of a game that a solver can handle, thereby deciding *what to be robust against*, while the solver decides *how to be optimal*. Experiments on urban network security games demonstrate substantial gains in scalability and generalization over strong context-blind baselines.

1 INTRODUCTION

Game theory (Shoham & Leyton-Brown, 2008) models strategic interaction in a wide range of real-world applications—security, defense, adversarial resource allocation, auctions, negotiation, and beyond—where agents with competing objectives must be reasoned about. Researchers have made substantial progress in solving games, including computing equilibria via linear programs (Von Neumann & Morgenstern, 1953; von Stengel, 1996; Shoham & Leyton-Brown, 2008); developing scalable algorithms such as double oracle (McMahan et al., 2003; Zhang et al., 2024), counterfactual regret minimization (Zinkevich et al., 2008; Meng et al., 2026a), and, more recently, stochastic optimization methods (Gemp et al., 2024a; Meng et al., 2025; Zhuang et al., 2026b). Breakthroughs—most notably superhuman performance in Go (Silver et al., 2016; 2017) and heads-up no-limit Texas hold'em (Moravčík et al., 2017; Brown & Sandholm, 2018; 2019)—are key milestones in AI. However, applying game theory in practice faces a fundamental dilemma: a model faithful enough to capture the real situation is typically too complex to solve, whereas a model simple enough to solve discards information that actually matters. The usual resolution is to commit to one side—oversimplify until the game is solvable, or keep the model rich and accept intractability.

That trade-off is sharper when the game is not a fixed object. In deployment the true parameters are unknown and drift, so a one-shot simplification is soon the wrong game, while a model that stays fully faithful remains intractable. We therefore study *open games with varying context*: strategic settings in which the true game is not only unknown but changes over time, and the information that matters arrives as *context*—unstructured, semantic reports and observations that a numeric solver cannot consume. Existing solvers (equilibrium computation, deep RL, pure LLM-generated policies) assume a fully specified, fixed, numeric game. That assumption fails in two ways. First, the game is *dynamic*: parameters drift after planning, so a once-optimal strategy becomes exploitable. Second, the decisive information arrives as context that a context-blind solver cannot consume, so a strategy that ignores it can be badly wrong for the situation at hand. A concrete instance is urban network security games (UNSGs) (Jain et al., 2011; 2013; Zhang et al., 2017; 2019; Zhuang et al., 2026a), which model the deployment of patrol officers on a city road network to intercept a criminal escaping from an incident. The information that matters—traffic bulletins, road closures, observed trajectories,

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.

operational signals—arrives as unstructured, semantic, and drifting context. Planning is NP-hard, and existing solvers (Li et al., 2021; Xue et al., 2021; 2022; Li et al., 2023; 2024) cannot consume this stream.

Closing the gap requires a component that can dynamically interpret varying context into the numeric parameters of a solvable game. Large language models (LLMs) (Ke et al., 2025) can *read* heterogeneous context and reason about what it implies. We use that capability as a *world-to-game bridge*: the LLM interprets the context stream into the numeric parameters of a game that a solver can handle, thereby deciding *what to be robust against*, while exact, fast components decide *how to be optimal*. We organize this division of labor into the **Context-Aware Game-Theoretic Agent** (CAGTA), a re-planning loop—plan, run, observe context, detect drift, re-plan—in which the LLM interpreter stays off the decision path and is invoked only when the context leaves the planned-for uncertainty set.

Contributions.

1. *A general framework for open games with varying context.* CAGTA is an LLM agent that acts as a world-to-game bridge. We prove that a correct context refinement can improve the robust value of the plan (Propositions 1 and 2).
2. *Theoretical guarantees.* Re-plans compose into an approximate maximin strategy of the meta-game under evolving context, with an explicit, additive error budget (Theorem 1).
3. *Experimental evaluation.* We instantiate CAGTA on GraphChase (Zhuang et al., 2026a) with a context layer and an LLM planner. Results show significant gains in scalability and generalization over context-blind baselines.

2 Related Work

Solving imperfect-information extensive-form games. A large literature develops techniques for solving extensive-form games, including subgame solving (Burch et al., 2014; Brown & Sandholm, 2017; Moravčík et al., 2017; Brown & Sandholm, 2018) together with counterfactual regret minimization (Meng et al., 2026b), methods for repeated games with uncertain payoffs (Zhang et al., 2022a), and methods for multiplayer games (Zhang et al., 2022b). These techniques assume that the game is given and fixed, and therefore do not apply to open games whose parameters must be inferred from varying context.

LLM agents for games. LLMs are increasingly used to generate policies and reason about games. Code-Space Response Oracles (CSRO) (Hennes et al., 2026) generate executable Python policies and refine them via best-response loops. Game of Thoughts (Kempinski et al., 2025) uses LLMs as best-response oracles in a double-oracle/PSRO style loop, and *States as Strings as Strategies* (Gemp et al., 2024b) makes language itself the game object—modeling dialogue as an extensive-form game over LLM string transitions and steering the LLM with game-theoretic equilibrium solvers, with the games synthesized from natural language. RHM (Liu et al., 2026a) deploys LLM agents to solve repeated games by modeling changing behaviors and adapting strategically. PrimeNash (Liu et al., 2026b) has LLM agents derive closed-form Nash equilibria with machine-checkable proofs, and LegoNE (Li et al., 2026) discovers approximate-equilibrium algorithms whose worst-case guarantees are automatically certified. These methods use LLMs as *proposers* but not as *context fusers*: they solve a game whose parameters are given, rather than interpreting the world to obtain those parameters.

From language to games: autoformalization and code world models. A complementary line uses the LLM to produce the *model* a solver consumes, rather than a policy. Mensfelt et al. (Mensfelt et al., 2024; 2025) autoformalize natural-language game descriptions into executable formal rules, repairing generated code against a solver's feedback and, in a follow-up, running tournaments between LLM-augmented agents over the formalized scenarios; Deng et al. (Deng et al., 2025) translate game descriptions into extensive-form game representations, first recovering the information-set structure and then emitting a complete game (in pygambit) from which equilibria are computed directly. Code World Models (Lehrach et al., 2026) bring the same idea into planning: the LLM writes an executable Python simulator of a game from its natural-language rules, and a Monte-Carlo tree-search planner

(with hidden-state inference for imperfect information) searches it—the LLM supplies the model, not the moves. These works formalize a *complete, static* description into a single concrete model. CAGTA's interpreter faces a *drifting, incomplete* world (open game).

3 PRELIMINARIES

We formalize the games we solve as finite, two-player zero-sum imperfect-information extensive-form games with perfect recall (Shoham & Leyton-Brown, 2008). An extensive-form game is a finite tree of *states* s; at each non-terminal state a designated player chooses an action, and the pair of choices determines the transition to a successor state (in the simultaneous-move case, both players choose at the same state). Each player's *information set* is a set of states the player cannot distinguish; *perfect recall* means each player remembers its own actions and observations. A *behavior strategy* of player 1 (the *defender*, who maximizes) is a map π assigning to each of its information sets a distribution over actions; write Π_D for the set of behavior strategies of the defender. A *behavior strategy* of player 2 (the *adversary*, who minimizes) is analogously a map $y \in \Pi_A$. Write $\pi(a; s)$ for the probability that π places on action a at the information set containing s (constant across that set), and likewise $y(b; s)$. For a finite two-player zero-sum game the value is well-defined, $v^* = \max_{\pi\in\Pi_D} \min_{y\in\Pi_A} U(\pi, y) = \min_{y\in\Pi_A} \max_{\pi\in\Pi_D} U(\pi, y)$, where $U(\pi, y)$ is the expected payoff; a pair (π^*, y^*) with $U(\pi^*, y^*) = v^*$ is a Nash equilibrium (the defender's component is a maximin strategy). The *exploitability* of a defender strategy π (Johanson et al., 2011) is $\mathsf{Expl}_D(\pi) = \max_{\pi'\in\Pi_D} \min_{y\in\Pi_A} U(\pi', y) - \min_{y\in\Pi_A} U(\pi, y)$, the gap between the best the defender could guarantee ($\max_{\pi'} \min_y U(\pi', y) = v^*$) and what π guarantees against the adversary's best response; it is zero iff π is maximin.

4 THE CONTEXT-AWARE GAME-THEORETIC AGENT (CAGTA)

To solve open games with varying context, we propose the **Context-Aware Game-Theoretic Agent** (CAGTA). CAGTA is a re-planning agent for games whose true parameters are unknown and drift over time, given a stream of context. An LLM acts as a *world-to-game bridge*: it interprets the context stream into the numeric parameters a solver can consume, so that the solver can compute a maximin strategy against the worst-case adversary. Drift detection covers unforeseen changes by triggering re-planning when the context leaves the uncertainty set the current plan was designed for.

4.1 CONTEXT-TO-GAME BRIDGE: CONTEXT INTERPRETER

In deployment, the open game is typically neither fixed nor fully known. Over time the agent receives a stream of semantic observations—traffic bulletins and closure reports in a network setting, market or operational reports in others, observed adversary behavior, and domain priors. Each observation is a natural-language or semi-structured report; the stream is the agent's world input, which a numeric solver cannot consume. Let θ denote the *environment parameters* of the game—in a network setting, the edge costs; in general, any payoff-relevant quantities. Each θ defines a game $\Gamma(\theta)$ with payoff U_θ and value $v^*(\theta)$, over the same game tree. The true parameters θ^* are unknown; a priori the planner only knows $\theta^* \in \Theta$ (or has a belief over Θ).

A *context* at time t is a set C_t of observations relevant to the game—reports, bulletins, observed trajectories, and potentially heterogeneous modalities such as images, sensor or video feeds. The sequence $C_1, C_2, \ldots$ is generated by an unknown kernel $\kappa(\cdot \mid C_t)$: a probability distribution over the next context C_{t+1} given the reports so far (it may also depend on the reached state; we suppress that argument). The agent does not know κ; the kernel appears only in the analysis, as the law that averages successor values over future context. The *context interpreter* is an LLM—a multimodal LLM when the context is multimodal—that maps the current context stream C_t and the current game state to the *refined uncertainty set* $\Theta(C_t)$, the parameter bounds $\hat{\Theta}_t$ consistent with the reports: for each payoff-relevant quantity, a bound from the reports, so $\Theta(C_t) = \hat{\Theta}_t \subseteq \Theta$.

The context-to-game bridge turns the refined set into the concrete game the solver will see, i.e., the interpreted parameters are written into the game. Because the re-planning engine solves a *concrete* game, the bridge selects a representative point $\hat{\theta}_t \in \Theta(C_t)$—e.g., the worst-case parameter vector within the refined set—and realizes $\Gamma(\hat{\theta}_t)$: effective parameters update the game's costs, and closures

set the availability of actions. This step makes context *actionable*: it produces the concrete game $\Gamma(\hat{\theta}_t)$ that the re-planning step consumes.

When drift fires, the incumbent's coverage set no longer contains the new interpretation, so the new coverage set replaces it at the next epoch. Let Θ_t denote the uncertainty set under which the current strategy was planned (its *coverage set*); the system *re-plans* when the current context is no longer covered: $\mathrm{Drift}(C_t, \Theta_t) \equiv \left[\, \Theta(C_t) \not\subseteq \Theta_t \,\right]$.

4.2 The Advantage of Context-to-Game Bridge

We now formalize why exploiting context can improve planning. The planner faces a game whose true parameters θ^* are unknown; we model this with an uncertainty set Θ (or a belief over Θ). A *context-blind* plan must be robust to the entire set Θ; a *context-aware* plan conditions on the refined set $\Theta(C)$ obtained from interpreting context C.

For a strategy π, write $V_\theta(\pi) = \min_{y \in \Pi_A} U_\theta(\pi, y)$ for its worst-case payoff in game θ. Let $\mathsf{RV}(\pi; \Theta') = \min_{\theta \in \Theta'} \min_{y \in \Pi_A} U_\theta(\pi, y) = \min_{\theta \in \Theta'} V_\theta(\pi)$ be the *robust value over* Θ', the payoff that π guarantees against the worst-case environment in Θ' and the worst-case adversary, and write $\pi^*_C = \arg\max_\pi \mathsf{RV}(\pi; \Theta(C))$ and $\pi^*_\Theta = \arg\max_\pi \mathsf{RV}(\pi; \Theta)$ for the *context-aware* and *context-blind* optima. The context-blind plan maximizes over Θ; the context-aware plan maximizes over the refined set.

The value of context is captured by the following claim: refining the uncertainty set never hurts the robust value the plan is designed to guarantee.

Proposition 1 (Context refinement can improve the robust value)**.** *If the true parameter $\theta^* \in \Theta$ and context C yields a correct refinement, i.e., $\theta^* \in \Theta(C) \subseteq \Theta$, then $\max_\pi \mathsf{RV}(\pi; \Theta(C)) \geq \max_\pi \mathsf{RV}(\pi; \Theta)$, and, evaluated at the true parameter θ^*, the context-aware optimum π^*_C achieves at least the context-blind plan's* worst-case *robust value $\mathsf{RV}(\pi^*_\Theta; \Theta)$, i.e., $\min_{y \in \Pi_A} U_{\theta^*}(\pi^*_C, y) \geq \mathsf{RV}(\pi^*_\Theta; \Theta)$. (All proofs are in Appendix A.)*

Remark 1 (What the refinement guarantees against θ^*). The above proposition implies that, evaluated at the true parameter θ^*, the context-aware optimum π^*_C achieves $\min_y U_{\theta^*}(\pi^*_\Theta, y) = \mathsf{RV}(\pi^*_\Theta; \Theta) + E$ for some $E \geq 0$. The per-θ^* domination $\min_y U_{\theta^*}(\pi^*_C, y) \geq \min_y U_{\theta^*}(\pi^*_\Theta, y)$ therefore holds whenever $\mathsf{RV}(\pi^*_C; \Theta(C)) \geq \min_y U_{\theta^*}(\pi^*_\Theta, y)$, and in particular under either sufficient condition: (i) *perfect refinement*, $\Theta(C) = \{\theta^*\}$, in which case π^*_C maximizes $V_{\theta^*}(\cdot)$ outright; or (ii) θ^* is a *worst-case* parameter for the context-blind plan, $V_{\theta^*}(\pi^*_\Theta) = \mathsf{RV}(\pi^*_\Theta; \Theta)$ (i.e., $E = 0$), in which case $\min_y U_{\theta^*}(\pi^*_C, y) \geq \mathsf{RV}(\pi^*_\Theta; \Theta) = \min_y U_{\theta^*}(\pi^*_\Theta, y)$. Otherwise π^*_Θ may be "lucky" at θ^* while being dragged down elsewhere in Θ, and the domination can fail.

The next proposition shows that this stronger comparison does hold once the refinement is *accurate enough*—when the residual uncertainty left inside $\Theta(C)$ is small relative to the optimality gap the context-blind plan already concedes at θ^*, the context-aware plan dominates it even at the true parameter.

Proposition 2 (Small uncertainty sets yield per-θ^* domination)**.** *If the true parameter $\theta^* \in \Theta$ and context C yields a correct refinement, i.e., $\theta^* \in \Theta(C) \subseteq \Theta$, and $V_\theta(\pi) = \min_{y \in \Pi_A} U_\theta(\pi, y)$ is the worst-case payoff in game θ, let $v^*(\theta^*) = \max_\pi V_{\theta^*}(\pi)$ be the value of the true game and $g = v^*(\theta^*) - \min_y U_{\theta^*}(\pi^*_\Theta, y) \geq 0$ the context-blind plan's optimality gap at θ^*. Then $\min_y U_{\theta^*}(\pi^*_C, y) \geq \min_y U_{\theta^*}(\pi^*_\Theta, y)$ whenever the* residual refinement loss $\rho(\Theta(C))$ *is at most the gap, i.e., $\rho(\Theta(C)) := v^*(\theta^*) - \mathsf{RV}(\pi^*_C; \Theta(C)) \leq g$. If in addition $V_\theta(\pi)$ is L-Lipschitz in θ with respect to a metric d on the parameter space Θ (e.g., the sup-norm on parameter vectors)—that is, $|V_\theta(\pi) - V_{\theta'}(\pi)| \leq L\, d(\theta, \theta')$ for every strategy π and all $\theta, \theta' \in \Theta$—and $\delta := \mathrm{diam}(\Theta(C)) = \sup_{\theta, \theta' \in \Theta(C)} d(\theta, \theta')$ is the diameter of the refined set, then it suffices that $L\delta \leq g$.*

This proposition makes the value of interpreter accuracy precise: shrinking $\Theta(C)$ reduces the residual loss $\rho(\Theta(C))$, and domination at θ^* is achieved once $\rho(\Theta(C))$ falls below the context-blind gap g—a more accurate context interpreter provably turns the robust-value improvement of Proposition 1 into a per-θ^* guarantee.

Example 1 (Traffic context improves the defender's value)**.** *Consider a tiny network in the Urban Network Security Game (UNSG), where the criminal starts at s, must reach a target t within the budget τ, and two disjoint roads connect s to t: a direct road through edge e_1 and a longer one through*

Algorithm 1 CAGTA

Require: game G, base strategy π_0 with coverage set Θ_0, context generator, solver engine (trained solver or LLM strategy generator)
Ensure: active strategy π_{act}
1: $\pi_{\text{act}} \leftarrow \pi_0$; $\Theta_t \leftarrow \Theta_0$ ▷ base strategy and its coverage set
2: run π_{act} on the decision path
3: **repeat**
4: observe context stream C_t and game state
5: $\Theta(C_t) \leftarrow \text{INTERPRET}(C_t, \text{state})$ ▷ LLM: parameter bounds $\hat{\Theta}_t$; off the decision path
6: **if** $\text{Drift}(C_t, \Theta_t)$ **then** ▷ $\Theta(C_t) \not\subseteq \Theta_t$
7: $\Gamma_{\text{sol}} \leftarrow \Gamma(\hat{\theta}_t)$ ▷ select the game parameter $\hat{\theta}_t$
8: $\pi_{\text{act}} \leftarrow \text{REPLAN}(\Gamma_{\text{sol}}, \pi_{\text{act}})$ ▷ run the solver
9: $\Theta_t \leftarrow \Theta(C_t)$ ▷ update the coverage set to the new interpretation
10: run π_{act} (decision path)
11: **until** end of deployment
12: **return** π_{act}

e_2; given the true travel times, only one road is feasible within τ. The defender has a single patrol unit and does not know a priori which road the criminal can use, so it must hedge by randomizing between e_1 and e_2, intercepting with probability at most $1/2$. Now a traffic bulletin arrives: "edge e_1 is closed." Interpreting this bulletin refines the weight set to $\Theta(C_t) = \{w : w(e_1) = +\infty\}$, so only e_2 is feasible; the defender places its unit on e_2 and intercepts with certainty. The context improves the defender's guaranteed value from $1/2$ to 1—an instance of Propositions 1 and 2 in which the bulletin removes over-hedging by shrinking Θ to $\Theta(C_t)$.

4.3 THE CAGTA FRAMEWORK

Propositions 1 and 2 make the contribution precise: the value of context is the amount of over-hedging removed by a correct refinement, $\max_\pi \mathsf{RV}(\pi; \Theta(C)) - \max_\pi \mathsf{RV}(\pi; \Theta) \geq 0$. The LLM's distinctive role is producing $\Theta(C)$ from unstructured, heterogeneous context—a semantic translation no context-blind solver performs. CAGTA operationalizes this refinement and re-plans accordingly. The LLM *interprets*—deciding, from context, what to be robust against; the solver computes a maximin strategy against the worst-case adversary in the specified game; and drift detection covers unforeseen cases reactively.

In CAGTA, the agent repeats: *plan* (initialize with the best available strategy), *run* (execute the active strategy on the decision path), *observe* (collect context C_t and game state), *detect drift* (evaluated on events and periodically), and *re-plan*. Algorithm 1 is the agent's loop. It starts with a base strategy π_0 and its coverage set Θ_0—the uncertainty set under which π_0 was planned (line 1)—and repeats three steps: *observe* the context stream and game state (line 4); *interpret* it into $\Theta(C_t)$ (line 5, off the decision path); and, only when the drift predicate fires ($\Theta(C_t) \not\subseteq \Theta_t$, i.e., the new context reveals a game outside the set the current plan is robust to; line 6), *re-plan* on the context-informed game, deploy the result (line 8), and update the coverage set to the new interpretation (line 9). Gating re-planning by drift, rather than at every step, lets the loop adapt rarely but exactly when the game changes.

4.4 THEORETICAL GUARANTEES OF CAGTA

We state guarantees that hold for *any* interpreter and solver satisfying the assumptions below. The key structural fact is that the stage decomposition of the horizon—the *meta-game*—has a 1-Lipschitz value recursion, so per-stage errors accumulate additively rather than compounding.

Stage decomposition. To analyze the re-planning loop under varying context we decompose the horizon into stages.

Definition 1 (Stage decomposition)**.** *Let $\Gamma(\theta)$ be a finite extensive-form game tree of depth T parameterized by environment parameters θ (Section 4.1). A node is a* state *s; at each internal state the defender chooses an action a and the adversary an action b. A* re-plan *is a behavior strategy from*

the reached state onward. Stage t begins at the reached state s_t and ends at the stage-$(t+1)$ states, where the next re-plan occurs.

The pair (a, b) at s_t under model θ determines two quantities:

- *the* within-stage payoff $U_t^\theta(a, b; s_t) \in \mathbb{R}$*, the payoff settled (or accrued) during stage t before the next re-plan—in UNSG this is 0 on non-terminal stages and $\{+1, -1\}$ at capture or escape;*
- *the* transition $p^\theta(\cdot : s_t, a, b)$*, a probability distribution over stage-$(t+1)$ states, encoding how θ (edge costs, closures, remaining time) maps the joint action to a successor s'.*

When the model is unambiguous we write $p(s' : s, a, b)$ for $p^\theta(s' : s, a, b)$. Both objects are part of $\Gamma(\theta)$: changing θ changes payoffs and transitions, not the tree of information sets.

Given a successor-value function W on the stage-$(t+1)$ states and next contexts, the mixed-action stage payoff at model θ is

$$Q_W^\theta(\pi, y) \;:=\; \sum_{a,b} \pi(a; s_t)\, y(b; s_t) \Big[\, U_t^\theta(a, b; s_t) + \sum_{s'} p^\theta(s' : s_t, a, b)\, \mathbb{E}_{C_{t+1}\sim\kappa(\cdot|C_t)}\, W(s', C_{t+1}) \,\Big],$$

where the expectation over C_{t+1} is omitted if the context is static (κ a point mass). The induced stage value at model θ is $V_W(s_t) = \max_\pi \min_y Q_W^\theta(\pi, y)$. The whole-game value of $\Gamma(\theta^*)$ is $V(s_1) = v^*(\theta^*)$.

Dynamic context: the meta-game. The true parameter follows a path $\{\theta_t^*\}_{t=1}^T$, revealed gradually by the context stream $\{C_t\}_{t=1}^T \sim \kappa$. We model the deployment as a single *meta-game* Γ_{meta}: an extensive-form game in which a chance node (or a worst-case node) resolves the path $\theta_1^*, \ldots, \theta_T^*$ and emits $C_1, \ldots, C_T$, and the payoff $U_{\Gamma_{\text{meta}}}(\pi, y)$ is the terminal payoff of $\Gamma(\theta_T^*)$ along the resulting play. Drift events ($\Theta(C_{t+1}) \not\subseteq \Theta_t$) partition the path into subgames $\Gamma_1, \ldots, \Gamma_T$. For subgame t reached at (s_t, C_t), the value $w_t(s_t, C_t)$ is the stage recursion of Definition 1 at the true model, $w_t(s_t, C_t) = \max_\pi \min_y Q_{w_{t+1}}^{\theta_t^*}(\pi, y)$, with w_{T+1} the terminal payoff of $\Gamma(\theta_T^*)$, so $w_1 = v^*_{\Gamma_{\text{meta}}}$. (If the world is adversarial rather than stochastic, $\mathbb{E}_{C_{t+1}\sim\kappa(\cdot|C_t)}$ is replaced by $\min_{C_{t+1}}$; the argument is unchanged.) Write $\|\cdot\|_\infty$ for the sup-norm over stage-$(t+1)$ states and next contexts.

At re-plan t the loop does not know w_{t+1}. It uses a *successor-value estimate* $\hat{w}_{t+1}$, a function of the same domain computed from (s_t, C_t) and the past (lookahead, a heuristic, or the value of a solver-size abstract game), and a representative model $\hat{\theta}_t \in \Theta(C_t)$ (Section 4.1). It therefore optimizes the perturbed objective $P(\pi) := \min_y Q_{\hat{w}_{t+1}}^{\hat{\theta}_t}(\pi, y)$ rather than the true objective $P^*(\pi) := \min_y Q_{w_{t+1}}^{\theta_t^*}(\pi, y)$. The *deployed strategy* $\pi = \pi_1 \circ \cdots \circ \pi_T$ is the concatenation that plays the re-plan π_t throughout stage t. Note that we do not need to assume that the game is perfect-information.

Assumptions. We have the following assumptions.

Assumption 1 (Correct interpretation)**.** *The true parameter stays inside the refined set: $\theta_t^* \in \Theta(C_t)$.*

Assumption 2 ($\varepsilon_{\text{sol}}^t$-optimal solver)**.** *The strategy π_t the solver returns is $\varepsilon_{\text{sol}}^t$-optimal for the stage objective it is given—the representative model $\hat{\theta}_t$ together with the successor-value estimate $\hat{w}_{t+1}$:* $\min_y Q_{\hat{w}_{t+1}}^{\hat{\theta}_t}(\pi_t, y) \;\geq\; \max_\pi \min_y Q_{\hat{w}_{t+1}}^{\hat{\theta}_t}(\pi, y) - \varepsilon_{\text{sol}}^t$.

Let d be a metric on Θ (e.g. the sup-norm on non-closed edge weights) and $\operatorname{diam}(\Theta(C_t)) := \sup_{\theta,\theta'\in\Theta(C_t)} d(\theta, \theta')$.

Assumption 3 (Parametric smoothness)**.** *For every successor function W and every strategy pair (π, y), and for all $\theta, \theta' \in \Theta(C_t)$ that induce the same set of available actions (the same closed-edge pattern), the stage payoff is L-Lipschitz in the model, with L independent of W:* $\big|Q_W^\theta(\pi, y) - Q_W^{\theta'}(\pi, y)\big| \;\leq\; L\, d(\theta, \theta')$.

Write $\delta_t := L \operatorname{diam}(\Theta(C_t)) + \|\hat{w}_{t+1} - w_{t+1}\|_\infty$ and define the model error $\varepsilon_{\text{model}}^t := 2L \operatorname{diam}(\Theta(C_t))$, the estimation error $\varepsilon_{\text{est}}^t := 2\|\hat{w}_{t+1} - w_{t+1}\|_\infty$, and $\varepsilon_v^t := \varepsilon_{\text{model}}^t + \varepsilon_{\text{est}}^t = 2\delta_t$.

Theorem 1 (Dynamic context: approximate maximin strategy of the meta-game)**.** *Under Assumptions 1–3 at every re-plan, the deployed strategy π is an ε-maximin strategy of Γ_{meta} with $\varepsilon = \sum_{t=1}^T (\varepsilon_{\text{sol}}^t + \varepsilon_v^t)$:* $\min_y U_{\Gamma_{\text{meta}}}(\pi, y) \;\geq\; v^*_{\Gamma_{\text{meta}}} - \sum_{t=1}^T \big(\varepsilon_{\text{sol}}^t + \varepsilon_v^t\big)$.

5 EXPERIMENTS

This section evaluates CAGTA on urban network security games. We first describe the experimental setup (Section 5.1), then present results on exploiting context (Section 5.2), generalization (Section 5.3), scaling (Section 5.4), and a real-world case (Section 5.5). Implementation details and the exact prompts appear in Appendices B and C.

5.1 EXPERIMENTAL SETUP

Platform. All experiments run on **GraphChase** (Zhuang et al., 2026a), whose games we adopt without modification: an urban network security game (UNSG) on a graph with k police officers (the defender) with real-time information, one adversary (the criminal) without real-time information, a set of exit nodes, and a lockdown horizon T (a time budget, not a discrete step count) bounding the criminal's path. We use three families of instances: 7×7 grids at two difficulty levels ($7(0.5)$ and $7(1.0)$, with ground-truth capture probabilities 0.5 and 1.0), rescaled $n \times n$ grids with $n \in \{30, 50, 100, 200\}$, and a real district built from OpenStreetMap with a real road-closure record.

Setting. An episode plays one criminal path: an officer co-located with the criminal at a junction captures it; a criminal that reaches an exit escapes; a timeout scores for the patrol. Each table cell reports the fraction of criminal paths the patrol stops. Where we follow GraphChase's own aggregation we say so explicitly: on the 7×7 rows of Section 5.3 the paths are GraphChase's own set. On the scaling and real-world rows the paths are a fixed sample.

Implementation. CAGTA plans from whatever context the patrol has been given, and each experiment reports three arms that differ only in that information: BLIND (nothing announced; the patrol plans on the pre-context map), INFORMED (the context is announced at $t{=}0$), and TOLD AT $t{=}L$ (the announcement arrives mid-episode, i.e., varying the context). In all experiments CAGTA uses an LLM interpreter and an LLM solver (any other solver that takes the game as given can be substituted): no training. We use DeepSeek-V4.1-Flash as a showcase; stronger models would be expected to perform at least as well. Two roles are exercised. *Role 1, the interpreter.* An LLM reads the bulletin text and returns Θ—for a closure, the road and span it refers to, which we resolve to the edge set it shuts. This is the role that must map, e.g., *"Governors Bdge between Douglas Cres and Astley Ave"* onto the right stretch of a real map (Section 5.5). *Role 2, the solver.* An LLM produces one plan per decision. It is given, for every officer, its position and its *exact travel time to every exit* together with the exit it is currently walking to; for the criminal, its position, travel times, and the trail of observed junctions, from which the prompt states a heading; and the clock in minutes with the remaining decisions. Prompts appear in Appendix C.

Baselines. We report one stochastic control separately: **Random**—at each decision each officer picks uniformly among its available actions (a uniformly random neighbour), reseeded per episode and, on the scored tables, averaged over ten seeds. We also report a Vanilla LLM that consumes the bulletin as text but does not map it into game parameters (see Appendix B.5 for details). GraphChase also provides several learned solvers for UNSGs: NSG-NFSP (Xue et al., 2021), NSGZero (Xue et al., 2022), Pretrained PSRO (Li et al., 2023), and Grasper (Li et al., 2024). All of these are context-blind.

5.2 EXPLOITING CONTEXT

We use GraphChase's 7×7 grid (node id $7 \cdot \mathrm{row} + \mathrm{col} + 1$), with the criminal at the centre $(3, 3)$ (node 25), four officers on the border midpoints (nodes 4, 22, 28, 46), four exits at nodes 2, 6, 43, and 49, and four express links that skip the streets, each costing one minute. The horizon is 5 minutes with one decision per minute. The graph is shown in Appendix B.1. **Bulletin.** The express link $(2, 3) \to (0, 1)$ is shut. That is the criminal's fast route to exit 2, whose cost then rises from 2 to 4 minutes while exit 6 stays at 2, so all 100 attacker paths go to exit 6.

Table 1 reports five findings. (i) Context decides the outcome: the same CAGTA solver on the same paths captures none of the paths when blind and all of them when informed (told at $t = 0$ or $t = 1$). (ii) The $t = 1$ column shows that CAGTA can exploit context that arrives mid-episode, not only at the start. (iii) The instance is not vacuous: the random baseline captures nothing under any of ten seeds, so the capture rate of 1 is produced by the decisions. (iv) The context-blind solvers NSG-NFSP,

Setting	NSG-NFSP	NSGZero	PSRO	Grasper	Random	Vanilla LLM	**CAGTA**
blind	0	0	0	0	0	0	**0**
told at t=0	0	0	0	0	0	0	**1**
told at t=1	0	0	0	0	0	0	**1**

Table 1: Capture rate with and without context on the 7×7 closure instance.

instance	horizon	**CAGTA**	PSRO	Grasper	NSGZero	NSG-NFSP
7(0.5) hard	T=4	**0.500**	0.477	0.481	0.396	0.466
	T=8	**0.952**	0.385	0.236	0.220	0.107
7(1.0) easy	T=4	**1.000**	0.996	0.989	0.741	0.982
	T=8	**1.000**	0.077	0.195	0.145	0.267

Table 2: Robustness to horizon shift (capture rate).

NSGZero, Pretrained PSRO, and Grasper all fail: they were trained on unweighted grids and cannot consume the bulletin. (v) That the Vanilla LLM captures nothing shows that the world-to-game bridge (interpreting context into game parameters) is critical.

5.3 GENERALIZATION

GraphChase's robustness tables show that its learned policies fail to transfer along two axes: extending the horizon, and reweighting the edges. We insert CAGTA into those two tables (Tables 2 and 3), evaluated on GraphChase's own protocols. The setting is detailed in Appendix B.2.

In Table 2, the assignment, path set, and aggregation are GraphChase's; CAGTA is evaluated on exactly those paths. On the hard instance we match the ground-truth value of the optimal policy (0.500) at T=4 and are well above all four learned policies at T=8 (0.952 against 0.107–0.385). For T=8 the paths are not sampled from an optimal policy, so a capture rate above 0.5 is possible. On the easy instance no path escapes at either horizon, whereas the platform's own policies fall from 0.996 to 0.077.

Table 3 shows that CAGTA remains at the optimal capture rate when edges are reweighted, whereas GraphChase's policies—trained on unweighted graphs and tested under weights they never saw—drop to 0.111–0.343. On the hard weighted instance the path set is again not an optimal sample (or the game value itself has changed with the weights), so a capture rate above 0.5 is possible.

5.4 SCALING

GraphChase's learned algorithms do not scale: on a 30×30 grid, "after two days of training, the run still stalled at the path-enumeration stage"; its appendix reports that enumeration explodes beyond 20×20; and on 100×100 none of the algorithms completes training (Zhuang et al., 2026a). CAGTA requires no such training.

Table 4 reports the sweep. Instances are rescaled so that *time*, not hops, is the invariant: τ=60 min at every size, seven decisions per episode ($\Delta=\tau/7 \approx 8.57$ min), edge cost $1.5 \cdot 29/(n-1)$ so the centre is always ≈ 43.5 min from a corner, and 100 attacker paths per configuration from a fixed seed. The evader starts at the centre, the exits are the four corners, and the patrol occupies the

		CAGTA	PSRO	Grasper	NSG-NFSP
7(0.5)	unweighted graph	**0.5**	0.483	0.484	0.435
	weighted graph	**1**	0.293	0.343	0.159
7(1.0)	unweighted graph	**1**	0.983	0.978	0.679
	weighted graph	**1**	0.111	0.259	0.254

Table 3: Robustness to edge weight variations.

	n=30(30 × 30)				n=50(50 × 50)				n=100(100 × 100)				n=200(200 × 200)			
	uniform		freeway		uniform		freeway		uniform		freeway		uniform		freeway	
	2	4	2	4	2	4	2	4	2	4	2	4	2	4	2	4
CAGTA	**0.53**	**0.72**	**0.48**	**1**	**0.54**	**0.77**	**0.59**	**1**	**0.44**	**0.76**	**0.52**	**1**	**0.41**	**0.74**	**0.48**	**1**
Hold	0.03	0.05	0.07	0.09	0.02	0.04	0.05	0.06	0	0.01	0.06	0.06	0.02	0.02	0.09	0.09
Random	0.04	0.06	0.03	0.06	0.01	0.03	0.03	0.03	0	0	0.02	0.03	0.01	0.01	0.03	0.04

Table 4: Capture rates on sixteen scaling configurations. Column groups are grid size ($n \times n$), graph type (*uniform*/*freeway*), and number of officers (2/4).

Algorithm	blind	informed	told at t=0.6
CAGTA	**0**	**1**	**1**
Vanilla LLM	0	0	0
Random	0	0	0
Hold	0	0	0

Table 5: Capture rate on the Toronto closure instance.

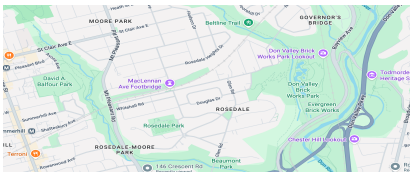


Figure 1: The real-world map.

four edge midpoints (or the west/east pair). In `uniform`, all edges have the same weight. In the `freeway` variant an east–west express row costs $0.3/1.5$ of the local cost, rescaled with n. Details are in Appendix B.3. CAGTA does not collapse with n: capture rates of 0.41–0.76 on `uniform` and 0.48–1.00 on `freeway` across a $44\times$ change in node count ($900 \to 40{,}000$), and capture rate 1 on every four-officer `freeway` configuration. The random baseline is never close (0.00–0.06), and at n=100 `uniform` with two officers every seed captures nothing. We also report `Hold`, in which officers never move; it likewise performs poorly.

5.5 A Real-World Case

Finally, we build one instance with no synthetic component: a real district, a real closure, and the city's own wording. **Location**: the Rosedale district of Toronto, on the western side of the Don River valley. **Network**: OpenStreetMap with every drivable class kept — 2,531 junctions, 2,632 edges and 49.4 km of street across about 2.6×5.0 km, every edge costed in minutes — of which the closure severs a pocket of 597 junctions spanning 1.2×1.2 km whose only link to the rest of the city is a single crossing, 45 m of the Governor's Road bridge over Mud Creek. **Bulletin**: the record `Tor-RD062025-1100` of Toronto's *Road Restrictions* feed shuts that bridge in both directions for 431 days, and the patrol reads the city's own sentence: *"CLOSURE: Governors Bdge between Douglas Cres and Astley Ave is shut. Toronto-TMC: These closures are in place to allow for the rehabilitation works on the Governor's Road Bridge over Mud Creek."* **Players**: the criminal starts on Roxborough Drive, and the patrol is a *single* officer at Glen Road, inside the pocket. **Exits:** the Glen Road bridge, inside the pocket, and the first junction past the shut bridge on Governors Road. We play 20 criminal paths. The map is shown in Figure 1 and more details are in Appendix B.4. Table 5 shows the same pattern as on the synthetic graphs: CAGTA captures every path once the bulletin is read, and none when it is withheld.

6 Conclusion

We presented CAGTA, a framework for solving open games. An LLM acts as a world-to-game bridge that interprets unstructured context into solver-ready parameters, while a re-planning loop adapts when the context drifts. We proved that a correct refinement can improve the robust value of the plan, and that re-plans compose into an approximate maximin strategy of the meta-game with an explicit, additive error budget. On GraphChase, CAGTA exploits context, generalizes beyond trained horizons and edge weights, and scales to 200×200 grids and a real urban closure, settings in which context-blind baselines fail. In future work we will evaluate CAGTA on other open games.

AI USE STATEMENT

In this work, we used generative AI tools to implement methods (writing and editing experimental code). We have not used generative AI tools to generate synthetic datasets, formulate mathematical claims, write proofs, or interpret results; those tasks were performed by the authors. Additionally, we used generative AI tools to edit the manuscript for readability and to format tables. Separately, an LLM is a component of the proposed method (interpreter and planner; Section 5). We have reviewed all AI-assisted text and code. We take responsibility for the final content of this work.

REFERENCES

Noam Brown and Tuomas Sandholm. Safe and nested subgame solving for imperfect-information games. *Advances in neural information processing systems*, 30:689–699, 2017.

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.

Neil Burch, Michael Johanson, and Michael Bowling. Solving imperfect information games using decomposition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28, 2014.

Shilong Deng, Yongzhao Wang, and Rahul Savani. From natural language to extensive-form game representations. In *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems*, pp. 593–601, 2025.

Ian Gemp, Luke Marris, and Georgios Piliouras. Approximating Nash equilibria in normal-form games via stochastic optimization. In *International Conference on Learning Representations*, 2024a.

Ian Gemp, Roma Patel, Yoram Bachrach, Marc Lanctot, Vibhavari Dasagi, Luke Marris, Georgios Piliouras, Siqi Liu, and Karl Tuyls. Steering language models with game-theoretic solvers. *arXiv preprint arXiv:2402.01704*, 2024b.

Daniel Hennes, Zun Li, John Schultz, and Marc Lanctot. Code-space response oracles: generating interpretable multi-agent policies with large language models. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, pp. 3525–3527, 2026.

Manish Jain, Dmytro Korzhyk, Ondřej Vaněk, Vincent Conitzer, Michal Pěchouček, and Milind Tambe. A double oracle algorithm for zero-sum security games on graphs. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, pp. 327–334, 2011.

Manish Jain, Vincent Conitzer, and Milind Tambe. Security scheduling for real-world networks. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, pp. 215–222, 2013. ISBN 978-1-4503-1993-5.

Michael Johanson, Kevin Waugh, Michael Bowling, and Martin Zinkevich. Accelerating best response calculation in large extensive games. In *International Joint Conferences on Artificial Intelligence*, volume 11, pp. 258–265, 2011.

Zixuan Ke, Fangkai Jiao, Yifei Ming, Xuan-Phi Nguyen, Austin Xu, Do Xuan Long, Minzhi Li, Chengwei Qin, Peifeng Wang, Silvio Savarese, et al. A survey of frontiers in llm reasoning: Inference scaling, learning to reason, and agentic systems. *arXiv preprint arXiv:2504.09037*, 2025.

Benjamin Kempinski, Ian Gemp, Kate Larson, Marc Lanctot, Yoram Bachrach, and Tal Kachman. Game of thoughts: Iterative reasoning in game-theoretic domains with large language models. In *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems*, pp. 1088–1097, 2025.

Wolfgang Lehrach, Daniel Hennes, Miguel Lazaro-Gredilla, Xinghua Lou, Carter Wendelken, Zun Li, Antoine Dedieu, Marc Lanctot, Atil Iscen, John Schultz, et al. Code world models for general game playing. In *International Conference on Learning Representations*, volume 2026, pp. 133870–133927, 2026.

Hanyu Li, Dongchen Li, and Xiaotie Deng. Discovering expert-level Nash equilibrium algorithms with large language models. *Nature Communications*, 2026.

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*, pp. 1147–1155, 2024.

Shuxin Li, Youzhi Zhang, Xinrun Wang, Wanqi Xue, and Bo An. CFR-MIX: Solving imperfect information extensive-form games with combinatorial action space. In *International Joint Conferences on Artificial Intelligence*, pp. 3663–3669, 2021.

Shuxin Li, Xinrun Wang, Youzhi Zhang, Wanqi Xue, Jakub Černỳ, 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.

Naming Liu, Youzhi Zhang, and Ying Wen. Solving repeated games with large language model. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, pp. 569–577, 2026a.

Wenxuan Liu, Xiyuan Zhou, Xinlei Wang, Yuheng Cheng, Lixin Ye, Randall Berry, Leandros Tassiulas, Jianwei Huang, and Junhua Zhao. An LLM agent-based framework for analytical characterization of Nash equilibria. *Nexus*, 3(1), 2026b.

H Brendan McMahan, Geoffrey J Gordon, and Avrim Blum. Planning in the presence of cost functions controlled by an adversary. In *International conference on machine learning*, pp. 536–543, 2003.

Linjian Meng, Wubing Chen, Wenbin Li, Tianpei Yang, Youzhi Zhang, and Yang Gao. Reducing variance of stochastic optimization for approximating Nash equilibria in normal-form games. In *International Conference on Machine Learning*, 2025.

Linjian Meng, Tianpei Yang, Youzhi Zhang, Zhenxing Ge, and Yang Gao. Faster game solving via asymmetry of step sizes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 17161–17169, 2026a.

Linjian Meng, Tianpei Yang, Youzhi Zhang, Zhenxing Ge, Shangdong Yang, Tianyu Ding, Wenbin Li, Bo An, and Yang Gao. Efficient last-iterate convergence in solving extensive-form games. In *Advances in Neural Information Processing Systems*, volume 38, pp. 100628–100675, 2026b.

Agnieszka Mensfelt, Kostas Stathis, and Vince Trencsenyi. Autoformalization of game descriptions using large language models. *arXiv preprint arXiv:2409.12300*, 2024.

Agnieszka Mensfelt, Kostas Stathis, and Vince Trencsenyi. Generative agents for multi-agent autoformalization of interaction scenarios. In *ECAI 2025: 28th European Conference on Artificial Intelligence 25-30 October 2025, Bologna, Italy-Including 14th Conference on Prestigious Applications of Intelligent Systems (PAIS 2025) Proceedings*, pp. 3759–3766. SAGE Publications 1 Oliver's Yard, 55 City Road, London, EC1Y 1SP, 2025.

Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisỳ, 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.

Yoav Shoham and Kevin Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, 2008.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 2017.

John Von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1953.

Bernhard von Stengel. Efficient computation of behavior strategies. *Games and Economic Behavior*, 14(2):220–246, 1996.

Wanqi Xue, Youzhi Zhang, Shuxin Li, Xinrun Wang, Bo An, and Chai Kiat Yeo. Solving large-scale extensive-form network security games via neural fictitious self-play. In *International Joint Conferences on Artificial Intelligence*, pp. 3713–3720, 2021.

Wanqi Xue, Bo An, and Chai Kiat Yeo. Nsgzero: Efficiently learning non-exploitable policy in large-scale network security games with neural monte carlo tree search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 4646–4653, 2022.

Mengxiao Zhang, Peng Zhao, Haipeng Luo, and Zhi-Hua Zhou. No-regret learning in time-varying zero-sum games. In *International Conference on Machine Learning*, pp. 26772–26808. PMLR, 2022a.

Youzhi Zhang, Bo An, Long Tran-Thanh, Zhen Wang, Jiarui Gan, and Nicholas R Jennings. Optimal escape interdiction on transportation networks. In *International Joint Conferences on Artificial Intelligence*, pp. 3936–3944, 2017.

Youzhi Zhang, Qingyu Guo, Bo An, Long Tran-Thanh, and Nicholas R Jennings. Optimal interdiction of urban criminals with the aid of real-time information. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 1262–1269, 2019.

Youzhi Zhang, Bo An, and VS Subrahmanian. Correlation-based algorithm for team-maxmin equilibrium in multiplayer extensive-form games. In *International Joint Conferences on Artificial Intelligence*, pp. 606–612, 2022b.

Youzhi Zhang, Bo An, and Daniel Dajun Zeng. Dag-based column generation for adversarial team games. In *International conference on machine learning*, 2024.

Shuxin Zhuang, Shuxin Li, Tianji Yang, Muheng Li, Xianjie Shi, Bo An, and Youzhi Zhang. GraphChase: A platform and benchmark for urban network security games. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 10372–10383, 2026a.

Shuxin Zhuang, Linjian Meng, Shuxin Li, Minming Li, and Youzhi Zhang. Tree-based stochastic optimization for solving large-scale urban network security games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 1668–1675), 2026b.

Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. Regret minimization in games with incomplete information. In *Advances in Neural Information Processing Systems*, pp. 1729–1736, 2008.

A PROOFS

Proposition 1 (Context refinement can improve the robust value)**.** *If the true parameter $\theta^* \in \Theta$ and context C yields a correct refinement, i.e., $\theta^* \in \Theta(C) \subseteq \Theta$, then $\max_\pi \mathsf{RV}(\pi; \Theta(C)) \geq \max_\pi \mathsf{RV}(\pi; \Theta)$, and, evaluated at the true parameter θ^*, the context-aware optimum π^*_C achieves at least the context-blind plan's* worst-case *robust value $\mathsf{RV}(\pi^*_\Theta; \Theta)$, i.e., $\min_{y \in \Pi_A} U_{\theta^*}(\pi^*_C, y) \geq \mathsf{RV}(\pi^*_\Theta; \Theta)$.*

Proof. Recall $\pi^*_C = \arg\max_\pi \mathsf{RV}(\pi; \Theta(C))$ and $\pi^*_\Theta = \arg\max_\pi \mathsf{RV}(\pi; \Theta)$, the context-aware and context-blind optima defined above.

(i) A correct refinement cannot hurt the robust value. For any strategy π, since $\Theta(C) \subseteq \Theta$,

$$\mathsf{RV}(\pi; \Theta(C)) = \min_{\theta \in \Theta(C)} \min_{y \in \Pi_A} U_\theta(\pi, y) \;\geq\; \min_{\theta \in \Theta} \min_{y \in \Pi_A} U_\theta(\pi, y) = \mathsf{RV}(\pi; \Theta),$$

because the minimum over a smaller set is at least the minimum over a larger one. Maximizing both sides over π gives $\max_\pi \mathsf{RV}(\pi; \Theta(C)) \geq \max_\pi \mathsf{RV}(\pi; \Theta)$, i.e., $\mathsf{RV}(\pi^*_C; \Theta(C)) \geq \mathsf{RV}(\pi^*_\Theta; \Theta)$.

(ii) The context-aware optimum achieves at least the context-blind robust value against θ^*. Since $\theta^* \in \Theta(C)$ (correct refinement), for *every* strategy π, the worst-case payoff in the true game, $V_{\theta^*}(\pi) = \min_{y \in \Pi_A} U_{\theta^*}(\pi, y)$ (as defined above), satisfies

$$\min_{y \in \Pi_A} U_{\theta^*}(\pi, y) \;\geq\; \min_{\theta \in \Theta(C)} \min_{y \in \Pi_A} U_\theta(\pi, y) = \mathsf{RV}(\pi; \Theta(C)),$$

because the minimum over $\Theta(C)$—a set that contains θ^*—is at most the value taken at θ^* itself; hence $\mathsf{RV}(\pi; \Theta(C))$ is a lower bound on the payoff at the true parameter. Applying this to $\pi = \pi^*_C$ and chaining with (i),

$$\min_{y \in \Pi_A} U_{\theta^*}(\pi^*_C, y) \;\geq\; \mathsf{RV}(\pi^*_C; \Theta(C)) \;\geq\; \mathsf{RV}(\pi^*_\Theta; \Theta).$$

The right-hand side is the value the context-blind plan π^*_Θ is guaranteed to achieve against *every* $\theta \in \Theta$—in particular against the true θ^* (since $\theta^* \in \Theta$). Hence the context-aware plan, evaluated in the true environment, achieves at least what the context-blind plan is guaranteed to achieve. □

Proposition 2 (Small uncertainty sets yield per-θ^* domination)**.** *If the true parameter $\theta^* \in \Theta$ and context C yields a correct refinement, i.e., $\theta^* \in \Theta(C) \subseteq \Theta$, and $V_\theta(\pi) = \min_{y \in \Pi_A} U_\theta(\pi, y)$ is the worst-case payoff in game θ, let $v^*(\theta^*) = \max_\pi V_{\theta^*}(\pi)$ be the value of the true game and $g = v^*(\theta^*) - \min_y U_{\theta^*}(\pi^*_\Theta, y) \geq 0$ the context-blind plan's optimality gap at θ^*. Then $\min_y U_{\theta^*}(\pi^*_C, y) \geq \min_y U_{\theta^*}(\pi^*_\Theta, y)$ whenever the* residual refinement loss $\rho(\Theta(C))$ *is at most the gap, i.e., $\rho(\Theta(C)) := v^*(\theta^*) - \mathsf{RV}(\pi^*_C; \Theta(C)) \leq g$. If in addition $V_\theta(\pi)$ is L-Lipschitz in θ with respect to a metric d on the parameter space Θ (e.g., the sup-norm on parameter vectors)—that is, $|V_\theta(\pi) - V_{\theta'}(\pi)| \leq L\, d(\theta, \theta')$ for every strategy π and all $\theta, \theta' \in \Theta$—and $\delta := \operatorname{diam}(\Theta(C)) = \sup_{\theta, \theta' \in \Theta(C)} d(\theta, \theta')$ is the diameter of the refined set, then it suffices that $L\delta \leq g$.*

Proof. First claim ($\rho(\Theta(C)) \leq g$ implies domination). Since $\theta^* \in \Theta(C)$, $\min_y U_{\theta^*}(\pi^*_C, y) \geq \mathsf{RV}(\pi^*_C; \Theta(C)) = v^*(\theta^*) - \rho(\Theta(C))$, while by definition $\min_y U_{\theta^*}(\pi^*_\Theta, y) = v^*(\theta^*) - g$. Hence

$$\min_y U_{\theta^*}(\pi^*_C, y) - \min_y U_{\theta^*}(\pi^*_\Theta, y) \;\geq\; \big(v^*(\theta^*) - \rho(\Theta(C))\big) - \big(v^*(\theta^*) - g\big) = g - \rho(\Theta(C)) \;\geq\; 0$$

whenever $\rho(\Theta(C)) \leq g$, which proves the first claim.

Lipschitz form. With the metric d, the Lipschitz constant L, and the diameter $\delta = \operatorname{diam}(\Theta(C))$ of the statement, we bound the residual loss $\rho(\Theta(C))$ in three steps. *(a) Value perturbation within* $\Theta(C)$. Since $\theta^* \in \Theta(C)$, every $\theta \in \Theta(C)$ satisfies $d(\theta, \theta^*) \leq \delta$. Applying the Lipschitz bound with $\theta' = \theta^*$ gives, for every π and every $\theta \in \Theta(C)$,

$$V_\theta(\pi) \;\geq\; V_{\theta^*}(\pi) - L\, d(\theta, \theta^*) \;\geq\; V_{\theta^*}(\pi) - L\delta,$$

so each strategy's payoff varies by at most $L\delta$ as the parameter ranges over the refined set. *(b) The robust value is within $L\delta$ of the true value.* Taking the minimum over $\theta \in \Theta(C)$ of (a) (the lower bound holds for every θ, hence for their minimum) and then the maximum over π,

$$\mathsf{RV}(\pi^*_C; \Theta(C)) = \max_\pi \min_{\theta \in \Theta(C)} V_\theta(\pi) \;\geq\; \max_\pi \big[V_{\theta^*}(\pi) - L\delta\big] \;=\; \max_\pi V_{\theta^*}(\pi) - L\delta \;=\; v^*(\theta^*) - L\delta,$$

where the second equality holds because $L\delta$ is independent of π, and the last equality is the definition of $v^*(\theta^*)$. Hence the residual loss satisfies $\rho(\Theta(C)) = v^*(\theta^*) - \mathsf{RV}(\pi^*_C; \Theta(C)) \le L\delta$. *(c) Conclusion.* If $L\delta \le g$, then $\rho(\Theta(C)) \le L\delta \le g$, and domination follows from the first claim. □

Definition 2 (1-Lipschitz map)**.** *A map F between function spaces equipped with the sup-norm is 1-Lipschitz (nonexpansive) if $\|F(W) - F(W')\|_\infty \le \|W - W'\|_\infty$.*

Lemma 1 (Backward induction is 1-Lipschitz)**.** *The map $W \mapsto V_W$ of Definition 1 is 1-Lipschitz: $\|V_W - V_{W'}\|_\infty \le \|W - W'\|_\infty$. The same bound holds for the stage payoff itself: for every (π, y) and every model θ,*

$$\left|Q^\theta_W(\pi, y) - Q^\theta_{W'}(\pi, y)\right| \;\le\; \|W - W'\|_\infty.$$

Proof. Fix s_t, θ, and (π, y). The two payoffs $Q^\theta_W(\pi, y)$ and $Q^\theta_{W'}(\pi, y)$ differ only in the successor term. That term is a convex combination of the values of W (resp. W') because π, y, $p^\theta(\cdot : s_t, a, b)$, and $\kappa(\cdot \mid C_t)$ are probability distributions, hence

$$\left|Q^\theta_W(\pi, y) - Q^\theta_{W'}(\pi, y)\right| \;\le\; \|W - W'\|_\infty.$$

Max and min are each 1-Lipschitz in the payoff, so $|V_W(s_t) - V_{W'}(s_t)| \le \|W - W'\|_\infty$ at every s_t. □

The nonexpansiveness is automatic: each step is a max-min of convex combinations, so successor errors are never amplified. Consequently per-stage errors *add* over the horizon rather than compound.

Lemma 2 (Cost of optimizing a perturbed objective)**.** *Let F^* be an objective and $\hat{F}$ a perturbed version that is pointwise close. If $|F^*(\pi) - \hat{F}(\pi)| \le \delta$ for every π and π_ε is ε-optimal for $\hat{F}$, then*

$$F^*(\pi_\varepsilon) \;\ge\; \max_\pi F^*(\pi) - 2\delta - \varepsilon.$$

Proof. Pointwise closeness at the returned action gives $F^*(\pi_\varepsilon) \ge \hat{F}(\pi_\varepsilon) - \delta$. ε-optimality gives $\hat{F}(\pi_\varepsilon) \ge \max_\pi \hat{F}(\pi) - \varepsilon$. Pointwise closeness at a maximizer π^* of F^* gives $\max_\pi \hat{F}(\pi) \ge \hat{F}(\pi^*) \ge F^*(\pi^*) - \delta = \max_\pi F^*(\pi) - \delta$. Chaining yields the claim. The perturbation enters twice: once in the value the solve targets, once in the action it returns. □

Theorem 1 (Dynamic context: approximate maximin strategy of the meta-game)**.** *Under Assumptions 1–3 at every re-plan, the deployed strategy π is an ε-maximin strategy of Γ_{meta} with $\varepsilon = \sum_{t=1}^T (\varepsilon^t_{\mathrm{sol}} + \varepsilon^t_v)$:*

$$\min_y U_{\Gamma_{\mathrm{meta}}}(\pi, y) \;\ge\; v^*_{\Gamma_{\mathrm{meta}}} - \sum_{t=1}^T \big(\varepsilon^t_{\mathrm{sol}} + \varepsilon^t_v\big).$$

Proof. Let $A_t(s_t, C_t)$ be the value the deployed re-plans $\pi_t, \dots, \pi_T$ actually achieve from subgame t onward against a best-responding adversary,

$$A_t(s_t, C_t) \;:=\; \min_y Q^{\theta^*_t}_{A_{t+1}}(\pi_t, y),$$

with $A_{T+1} = w_{T+1}$ the exact terminal payoff, and let $E_t = \sum_{s=t}^T (\varepsilon^s_{\mathrm{sol}} + \varepsilon^s_v)$. In particular $A_1(s_1, C_1) = \min_y U_{\Gamma_{\mathrm{meta}}}(\pi, y)$. We prove $A_t(s_t, C_t) \ge w_t(s_t, C_t) - E_t$ at every reachable (s_t, C_t) by backward induction on t.

Pointwise closeness of the two objectives. Fix a stage t, a pair (π, y), and write $Q^{\hat{\theta}}_{\hat{w}}$ and $Q^{\theta^*}_w$ for brevity. By the triangle inequality the two payoffs differ in the model and in the successor:

$$\begin{aligned}\left|Q^{\hat{\theta}_t}_{\hat{w}_{t+1}}(\pi, y) - Q^{\theta^*_t}_{w_{t+1}}(\pi, y)\right| \;\le\; &\underbrace{\left|Q^{\hat{\theta}_t}_{\hat{w}_{t+1}}(\pi, y) - Q^{\theta^*_t}_{\hat{w}_{t+1}}(\pi, y)\right|}_{\le L\,\mathrm{diam}(\Theta(C_t))\ \text{(Assumption 3)}} \\ &+ \underbrace{\left|Q^{\theta^*_t}_{\hat{w}_{t+1}}(\pi, y) - Q^{\theta^*_t}_{w_{t+1}}(\pi, y)\right|}_{\le \|\hat{w}_{t+1} - w_{t+1}\|_\infty\ \text{(Lemma 1)}}.\end{aligned}$$

The first summand uses $\theta_t^*, \hat{\theta}_t \in \Theta(C_t)$ (Assumption 1). Hence the payoffs differ by at most δ_t for every (π, y). Since P and P^* are the minima over y, and min is 1-Lipschitz,

$$\big|P(\pi) - P^*(\pi)\big| \;\le\; \delta_t \qquad \text{for every } \pi.$$

Base ($t = T$). The last subgame ends at the horizon, so the successor is the exact terminal payoff: $\hat{w}_{T+1} = w_{T+1}$ and $\varepsilon_{\text{est}}^T = 0$, leaving $\varepsilon_v^T = 2L\,\mathrm{diam}(\Theta(C_T))$ and $\delta_T = L\,\mathrm{diam}(\Theta(C_T))$. By Assumption 2, π_T is $\varepsilon_{\text{sol}}^T$-optimal for P. Lemma 2 with the pointwise bound above then gives $P^*(\pi_T) \ge w_T - 2\delta_T - \varepsilon_{\text{sol}}^T = w_T - (\varepsilon_{\text{sol}}^T + \varepsilon_v^T)$. At $t = T$ the realized continuation is exact, $A_{T+1} = w_{T+1}$, so $A_T = P^*(\pi_T)$ and $A_T \ge w_T - E_T$.

Inductive step. By Assumption 2, π_t is $\varepsilon_{\text{sol}}^t$-optimal for P. Combined with $|P - P^*| \le \delta_t$ and $\max_\pi P^*(\pi) = w_t(s_t, C_t)$, Lemma 2 yields

$$P^*(\pi_t) \;\ge\; w_t(s_t, C_t) - \big(\varepsilon_{\text{sol}}^t + \varepsilon_v^t\big).$$

The loop does not play against w_{t+1}: after executing π_t it continues with its own re-plans, which realize A_{t+1}. Thus $A_t(s_t, C_t) = \min_y Q^{\theta_t^*}_{A_{t+1}}(\pi_t, y)$. For every adversary action y the two payoffs $Q^{\theta_t^*}_{A_{t+1}}(\pi_t, y)$ and $Q^{\theta_t^*}_{w_{t+1}}(\pi_t, y)$ differ only in the successor, a convex combination of continuation errors (Lemma 1), so

$$\big|Q^{\theta_t^*}_{A_{t+1}}(\pi_t, y) - Q^{\theta_t^*}_{w_{t+1}}(\pi_t, y)\big| \;\le\; \|A_{t+1} - w_{t+1}\|_\infty.$$

Evaluating at a minimizer $y^\star$ of $Q^{\theta_t^*}_{A_{t+1}}(\pi_t, \cdot)$ and using $Q^{\theta_t^*}_{w_{t+1}}(\pi_t, y^\star) \ge P^*(\pi_t)$ gives

$$A_t(s_t, C_t) \;\ge\; P^*(\pi_t) - \|A_{t+1} - w_{t+1}\|_\infty.$$

It remains to bound the continuation gap. No defender strategy can exceed the subgame value, so $A_{t+1} \le w_{t+1}$ at every successor. The induction hypothesis is the matching lower bound $A_{t+1} \ge w_{t+1} - E_{t+1}$. Hence $0 \le w_{t+1} - A_{t+1} \le E_{t+1}$ and $\|A_{t+1} - w_{t+1}\|_\infty = \sup_{s_{t+1}, C_{t+1}} \big|A_{t+1}(s_{t+1}, C_{t+1}) - w_{t+1}(s_{t+1}, C_{t+1})\big| \le E_{t+1}$. Substituting,

$$A_t(s_t, C_t) \;\ge\; P^*(\pi_t) - E_{t+1} \;\ge\; w_t(s_t, C_t) - E_{t+1} - (\varepsilon_{\text{sol}}^t + \varepsilon_v^t) \;=\; w_t(s_t, C_t) - E_t.$$

Conclusion. At $t = 1$, $A_1(s_1, C_1) = \min_y U_{\Gamma_{\text{meta}}}(\pi, y)$ and $w_1 = v^*_{\Gamma_{\text{meta}}}$, so

$$\min_y U_{\Gamma_{\text{meta}}}(\pi, y) \;\ge\; v^*_{\Gamma_{\text{meta}}} - E_1 \;=\; v^*_{\Gamma_{\text{meta}}} - \sum_{t=1}^{T} \big(\varepsilon_{\text{sol}}^t + \varepsilon_v^t\big),$$

which is the claim. □

B Settings

B.1 Settings for Exploiting Contexts

The instance is GraphChase's 7×7 grid: a junction's node id is $7 \cdot \text{row} + \text{col} + 1$, row 0 north and col 0 west, and one street segment costs one minute. The criminal starts at the centre, four officers start on the border midpoints, the four exits sit on the top and bottom edges, and four *express links* skip the streets at one minute each. The horizon is $T = 5$ with one decision per minute, and the criminal's path set is 100 copies of $(3, 3) \to (2, 3) \to (0, 5)$ — the only two-minute escape left once the bulletin shuts the express link $(2, 3) \to (0, 1)$.

```
        col0   col1   col2   col3   col4   col5   col6
row0     .      E      .      D      .      E      .       E   exit:  2 (0,1), 6 (0,5),
row1     .      .      .      .      .      .      .                  43 (6,0), 49 (6,6)
row2     .      .      .      .      .      .      .       D   officer start: 4 (0,3),
row3     D      .      .      A      .      .      D          22 (3,0), 28 (3,6), 46 (6,3)
row4     .      .      .      .      .      .      .       A   criminal start: 25 (3,3)
row5     .      .      .      .      .      .      .       .   an ordinary junction
row6     E      .      .      D      .      .      E

express links, one minute each, skipping the streets:
      18 = (2,3)  ->  E2 = 2 (0,1)     SHUT by the bulletin
      18 = (2,3)  ->  E3 = 6 (0,5)     open
       4 = (0,3)  ->  E2 = 2 (0,1)     open
       4 = (0,3)  ->  E3 = 6 (0,5)     open
```

The shut link is the criminal's fast route to exit 2, whose cost rises from 2 to 4 minutes while exit 6 stays at 2 — so every path in the set goes to exit 6 and the informed patrol's ceiling is $100/100$.

Interpreter. The patrol's bulletin is a generated sentence rather than a city's prose: the solver is handed *"(t=0) CLOSURE: the express link between (2,3) and (0,1) is shut for traffic."* together with the belief line *"the express link from (2,3) to exit (0,1) is closed; only the link to (0,5) and the streets remain"*, and the rows of Table 1 were measured with Θ applied from those two strings. The prompt is quoted in Appendix C.1.

Solver. One completion per decision at temperature 0, cached on disk and keyed by tag, prompt revision, style, the officers' positions, the criminal's position and $\lfloor t \rfloor$. The reply is executed as given; parsing is strict JSON, a malformed reply holds every officer, and a named junction that does not exist is a failure rather than a new node.

B.2 Settings for Generalization

Both instances are GraphChase's and are evaluated in its protocol: its own path set (`Game.build_paths_by_exit`), every path of that set played, the cell is $1 - \text{escapes}/\text{paths}$. The *hard* instance is the shipped $7(0.5)$ configuration; the *easy* one is its mirror — the officers start on the four corners and the exits sit on the border midpoints, so a one-officer-per-exit assignment is exact and every officer arrives no later than the criminal. The weighted rows re-draw the same topology once per run, as GraphChase does (its own realization: symmetric weights over $[0.5, 3.0]$, `weight_seed` 77).

```
the 7x7 family: the 7(0.5) hard map; the easy map is its mirror

       col0           col3           col6
row0     .  ---------  E  ---------   .       E   exit
         |             |             |         D   officer start
row1      - D           .             .        A   criminal start
         |             |             |         .   ordinary junction
row3     E  ---------  A  ---------  D
         |             |             |
row6     E  --D-------  D  ---------  E

hard  :  criminal 25 (3,3);  officers 9 (1,1), 28 (3,6), 45 (6,2), 46 (6,3);
          exits 4 (0,3), 22 (3,0), 43 (6,0), 49 (6,6);  T = 4 and T = 8
mirror:  criminal 25 (3,3);  officers 1 (0,0), 7 (0,6), 43 (6,0), 49 (6,6);
          exits 4 (0,3), 22 (3,0), 28 (3,6), 46 (6,3);  T = 4 and T = 8
```

B.3 Settings for Scaling

All sixteen configurations share one shape, rescaled so that *time* and not hops is the invariant: $\tau = 60$ min at every size with seven decisions per episode ($\Delta = \tau/7 \approx 8.57$ min), an edge cost of $1.5 \cdot 29/(n-1)$ minutes so the centre is always ≈ 43.5 min from a corner, and 100 criminal paths per configuration from one fixed seed. The criminal starts at the centre and the four exits are the corners; the patrol is either the four edge midpoints or only the west and east ones. In the `freeway` variant an east–west express row — at the same relative depth as row 22 of 30 — costs $0.3/1.5$ of the local cost, rescaled with n. The prompt does not grow with the map: it is 2,182–2,480 characters in total across the sweep, and the variation is the patrol size and the number of digits, never n.

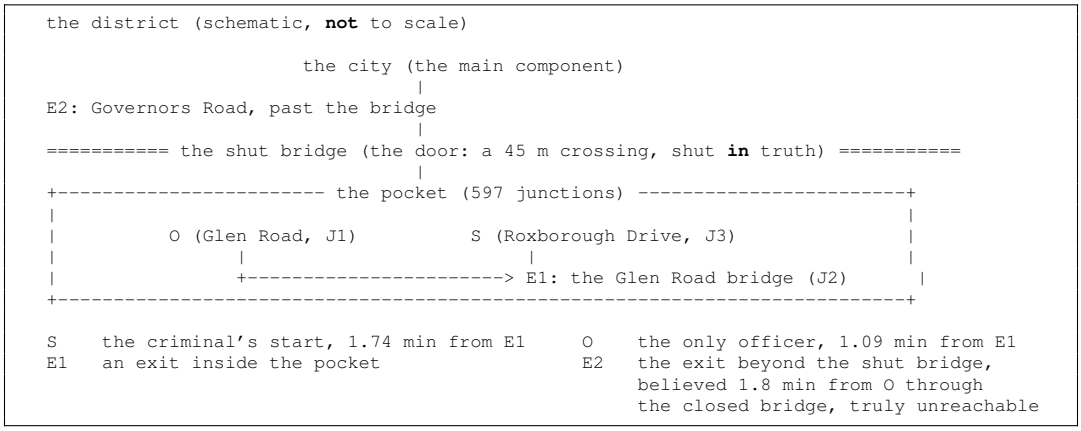
```
every n x n configuration, n = 30, 50, 100, 200

       col 0          col n/2         col n-1
row 0       E ------------- D ------------- E       E   an exit (the four corners)
            |               |               |       D   an officer's start (edge midpoints)
row n/2     D ------------- A ------------- D       A   the criminal's start (the centre)
            |               |               |
row n-1     E ------------- D ------------- E       the 2-officer rows drop the top and
                                                       bottom D (north and south)
```

Solver. The `caution` wording of Appendix C.3, temperature 0, one completion per decision, cached per configuration and executed as given.

B.4 SETTINGS ON REAL-WORLD CASES

The instance is a real closure on a real street network. The record is `Tor-RD062025-1100` from Toronto's *Road Restrictions (v3)* feed: *Governor's Road Bridge over Mud Creek*, a local road shut in both directions for 431 days, and the city's own prose as the bulletin. The network is OpenStreetMap with all drivable classes kept (2,531 junctions, 2,632 edges, 49 km) and every edge costed in minutes. The shut street severs a pocket of 597 junctions whose only link to the city is a single 45 m crossing — the bridge itself. The criminal starts on Roxborough Drive and the patrol is a *single* officer at Glen Road, inside the pocket; the exits are the *Glen Road bridge*, inside the pocket, and the first junction past the shut bridge. The horizon is 4.51 min with a decision every 0.644 min, and 20 criminal routes are played — one route, replayed: the evader is deterministic on this instance and follows its least-time route (1.74 min) to the live exit. The officer stands at the junction the prompt calls $J1$, 1.09 min from that exit, so a patrol that goes there directly arrives 0.65 min earlier than the criminal; the exits' two routes share 47.4% of their nodes.



Solver and interpreter. The same solver interface and cache discipline as above, with the real-map prompt of Appendix C.4: positions are named junctions and the reply names a junction label from a shortlist of 48. All policy variants are measured on the instance, all with the same state, the same reply schema and the same cache discipline. The interpreter is given the bulletin and the street names within 300 m, returns the street and span it closes, and its reading is resolved to edges by the same name-first matching that snaps the record; the world is then built from that reading, and it reproduces the table's arms exactly.

B.5 VANILLA LLM

Vanilla LLM runs the same language model, on the same instance, behind the same interface and reply schema as CAGTA's solver, and differs only in what the user message contains due to the lack of the game model. It is CAGTA with the formal model withheld: no travel times (no officer→exit or criminal→exit ETA), no "belief in force" line, and no critical-exit analysis — what is left is the map's own metric sentence (the cost of a segment, and the express row where the instance has one), the positions, the observed trail, the clock and the announced context in the city's own words, with the instruction to reason from those about which exit the criminal is heading for and which officer is closest to it. It isolates the *transfer* rather than the model: on the 7×7 closure the vanilla arm captures nothing in any arm (Table 1) and on the real-world instance likewise (Table 5), because the arithmetic that decides those episodes — the express link's cost, the shut bridge's route — is not in the prose.

C PROMPTS

Every prompt below is reproduced as the model receives it. Only the system message and the first user message of an episode are shown; later messages differ only in the positions, the ETAs, the observed trail and the clock line.

C.1 PROMPTS FOR EXPLOITING CONTEXTS

The interpreter. The system message is 1,095 characters and the user message 839; the model receives each paragraph unwrapped, and long lines are wrapped for the page here as everywhere below. The message the patrol reads is the harness's own sentence; the bulletin handed to the interpreter is the paraphrase it parses.

```
You are a traffic-and-security analyst for an urban patrol. You translate police and traffic
    bulletins into game parameters that a pursuit solver uses. Name roads by endpoints or
    corridor. Use EXACTLY one of these two forms -- 'segment (r,c)-(r,c)' (e.g. 'segment
    (2,3)-(0,1)') or 'row-N freeway, columns a-b' -- with (r,c) the row/column coordinates
    given to you. If a bulletin already states a road in one of those forms, copy it verbatim.
    Never invent edge IDs. Respond with STRICT JSON only, following exactly this schema:
{"weight_updates": [{"road": "<corridor>", "lo": <number|null>, "hi": <number|null>,
    "multiplier": <number|null>, "reason": "<string>"}], "anticipation": [{"road":
    "<corridor>", "eta_min": <number>, "note": "<string>"}], "threat_direction": "<toward E1
    | toward E2 | toward E3 | unknown>", "ignored": ["<bulletin id>", ...]}
Rules:
- A CLOSURE sets the road infeasible: set reason to 'closure' and lo/hi null.
- CONGESTION that doubles travel time: set multiplier 2.0 (or give lo/hi in minutes).
- If a bulletin is irrelevant or unparseable, list its id in 'ignored'.
```

```
City: a 7x7 grid road network.
Corridors: streets cost 1 minute per segment; express links, 1 minute: (2,3)-(0,1);
    (2,3)-(0,5); (0,3)-(0,1); (0,3)-(0,5).
Exits: E1 = node 2 = (0, 1); E2 = node 6 = (0, 5); E3 = node 43 = (6, 0); E4 = node 49 =
    (6, 6).
Criminal believed near node 25; remaining budget 5 min.
Current uncertainty set: 0 edges tracked, 0 already closed.

Bulletins:
[b2] (t=0.0) CLOSURE: the the fast link (2,3)-(0,1) is shut for traffic is closed.

Example output for "accident: the row-15 freeway is closed between columns 16 and 20":
    {"weight_updates": [{"road": "row-15 freeway, columns 16-20", "lo": null, "hi": null,
    "multiplier": null, "reason": "closure"}], "anticipation": [{"road": "row-15 freeway,
    columns 20-29", "eta_min": 5, "note": "congestion spreading east"}], "threat_direction":
    "toward E1", "ignored": []}
```

The prompt for solver. The system message is 1,864 characters and the user message 1,153; long lines are wrapped for the page, and the model receives each paragraph unwrapped.

```
You are the defence commander of an urban patrol. A criminal is fleeing toward one of the
    city's exits; you command four officers and must catch it.

The city is a 7x7 grid of intersections. Write a position as (row, col), row 0 north, col 0
    west, both 0..6.

Travel time: 1 minute per street segment. The city is a plain grid: every route between two
    junctions costs the number of streets along it. If a link is announced closed, neither may
    use it.

Exits: E1 (0, 1), E2 (0, 5), E3 (6, 0), E4 (6, 6).

An officer catches the criminal if it is at the same intersection when the criminal arrives.
    Arriving first and waiting is as good as intercepting.

For each officer choose the intersection it should walk to now (it will take the fastest route
    and wait there). An officer controls an exit ONLY if the officer's target is that exact
    exit and its given travel time to that exit is <= the criminal's given travel time to that
    exit. Assigning an officer to any other intersection covers no exit.

Critical exits are exits with criminal_time <= remaining_time. Every critical exit MUST have
     at
    least one officer assigned to that exact exit with officer_time <= criminal_time. If a
    critical exit has only one officer who can arrive on time, that officer MUST be assigned
     to
    that critical exit.

Before you answer, verify:
1. All 4 officer targets are distinct.
2. Every critical exit is covered by an on-time officer assigned to that exit.
3. No officer is sent to the same intersection as another.
4. Your reasoning sentence must not claim an exit is covered unless the assigned officer's
     time
    to that exit is <= the criminal's time to that exit.
```

```
Reply with STRICT JSON only: {"targets": [{"officer": 1, "row": R, "col": C}, {"officer": 2,
    "row": R, "col": C}, {"officer": 3, "row": R, "col": C}, {"officer": 4, "row": R, "col":
    C}], "reasoning": "<one sentence>"}
```

```
Patrol of 4 officers (travel time to each exit, exact minutes, open map -- no closures
    announced yet):
  officer 1 at (0, 3): E1 1.0, E2 1.0, E3 8.0, E4 8.0
  officer 2 at (3, 0): E1 4.0, E2 5.0, E3 3.0, E4 9.0
  officer 3 at (3, 6): E1 6.0, E2 4.0, E3 9.0, E4 3.0
  officer 4 at (6, 3): E1 7.0, E2 5.0, E3 3.0, E4 3.0
Criminal at (3, 3): E1 4.0, E2 2.0, E3 6.0, E4 6.0.
Criminal's observed trail: (none yet); moving unknown (first observation).
Elapsed 0.0 of 5.0 minutes, 5.0 remain (4 decisions after this one).
No announced context; no belief in force. ETAs are for the open map.
Critical exit analysis (remaining time = 5.0):
  E1 (0, 1): criminal 4.0 <= 5.0 -> CRITICAL. Officers on time: O1 1.0, O2 4.0.
  E2 (0, 5): criminal 2.0 <= 5.0 -> CRITICAL. Officers on time: O1 1.0 ONLY.
  E3 (6, 0): criminal 6.0 > 5.0 -> not critical. Officers on time: O2 3.0, O4 3.0.
  E4 (6, 6): criminal 6.0 > 5.0 -> not critical. Officers on time: O3 3.0, O4 3.0.

Mandatory: E2 must be assigned to officer 1. E1 must be assigned to an on-time officer
     (officer
    2 if officer 1 is used for E2). All targets must be distinct.
Give each officer a target intersection.
```

C.2 Prompts for Generalization

The hard instance, $T = 4$. GraphChase's own path set and aggregation accompany this prompt. The system message is 1,809 characters and the user message 517; long lines are wrapped for the page, and the model receives each paragraph unwrapped.

```
You are the defence commander of an urban patrol. A criminal is fleeing toward one of the
    city's exits; you command four officers and must catch it.
The city is a 7x7 grid of intersections. Write a position as (row, col), row 0 north, col 0
    west, both 0..6.
Travel time: 1 minute per street segment. The city is a plain grid: every route between two
    junctions costs the number of streets along it. Officers and the criminal move at the same
    speed.
Exits: E1 (0, 3), E2 (3, 0), E3 (6, 0), E4 (6, 6).
An officer catches the criminal if it is at the same intersection when the criminal arrives,
     so
    arriving first and waiting is as good as intercepting.
For each officer choose the intersection it should walk to now (it will take the fastest route
    and wait there). Before you answer, check your reply for this mistake: two officers sent
     to
    the same intersection, so that some exit has no officer walking to it. Officers sent to
     the
    same place guard one exit between them; naming an exit in your reasoning does not cover
     it,
    and the exit nobody walks to is the one the criminal escapes through. Give each officer a
    different target, and make sure every exit your own reasoning says must be covered has an
    officer actually walking to it. An officer controls an exit only if it gets there no later
    than the criminal does. If any exit the criminal can reach in time is left to nobody, or
    two officers share one intersection, change the assignment before you answer. The travel
    times given to you below are exact weighted shortest-path times on this map, in minutes;
    use them rather than estimating.
Reply with STRICT JSON only: {"targets": [{"officer": 1, "row": R, "col": C}, {"officer": 2,
    "row": R, "col": C}, {"officer": 3, "row": R, "col": C}, {"officer": 4, "row": R, "col":
    C}], "reasoning": "<one sentence>"}
```

```
Patrol of 4 officers (travel time to each exit, exact minutes):
  officer 1 at (1, 1): E1 3.0, E2 3.0, E3 6.0, E4 10.0
  officer 2 at (3, 6): E1 6.0, E2 6.0, E3 9.0, E4 3.0
  officer 3 at (6, 2): E1 7.0, E2 5.0, E3 2.0, E4 4.0
  officer 4 at (6, 3): E1 6.0, E2 6.0, E3 3.0, E4 3.0
Criminal at (3, 3): E1 3.0, E2 3.0, E3 6.0, E4 6.0.
Criminal's observed trail: (none yet); moving unknown (first observation).
Elapsed 0.0 of 4.0 minutes, 4.0 remain (3 decisions after this one).
Give each officer a target intersection.
```

The easy instance (the mirror), $T = 4$. The same prompt with the mirror's positions and exits. The system message is 1,809 characters and the user message 516; long lines are wrapped for the page, and the model receives each paragraph unwrapped.

```
You are the defence commander of an urban patrol. A criminal is fleeing toward one of the
    city's exits; you command four officers and must catch it.
The city is a 7x7 grid of intersections. Write a position as (row, col), row 0 north, col 0
    west, both 0..6.
Travel time: 1 minute per street segment. The city is a plain grid: every route between two
    junctions costs the number of streets along it. Officers and the criminal move at the same
    speed.
Exits: E1 (0, 3), E2 (3, 0), E3 (3, 6), E4 (6, 3).
An officer catches the criminal if it is at the same intersection when the criminal arrives,
     so
    arriving first and waiting is as good as intercepting.
For each officer choose the intersection it should walk to now (it will take the fastest route
    and wait there). Before you answer, check your reply for this mistake: two officers sent
     to
    the same intersection, so that some exit has no officer walking to it. Officers sent to
     the
    same place guard one exit between them; naming an exit in your reasoning does not cover
     it,
    and the exit nobody walks to is the one the criminal escapes through. Give each officer a
    different target, and make sure every exit your own reasoning says must be covered has an
    officer actually walking to it. An officer controls an exit only if it gets there no later
    than the criminal does. If any exit the criminal can reach in time is left to nobody, or
    two officers share one intersection, change the assignment before you answer. The travel
    times given to you below are exact weighted shortest-path times on this map, in minutes;
    use them rather than estimating.
Reply with STRICT JSON only: {"targets": [{"officer": 1, "row": R, "col": C}, {"officer": 2,
    "row": R, "col": C}, {"officer": 3, "row": R, "col": C}, {"officer": 4, "row": R, "col":
    C}], "reasoning": "<one sentence>"}
```

```
Patrol of 4 officers (travel time to each exit, exact minutes):
  officer 1 at (0, 0): E1 3.0, E2 3.0, E3 9.0, E4 9.0
  officer 2 at (0, 6): E1 3.0, E2 9.0, E3 3.0, E4 9.0
  officer 3 at (6, 0): E1 9.0, E2 3.0, E3 9.0, E4 3.0
  officer 4 at (6, 6): E1 9.0, E2 9.0, E3 3.0, E4 3.0
Criminal at (3, 3): E1 3.0, E2 3.0, E3 3.0, E4 3.0.
Criminal's observed trail: (none yet); moving unknown (first observation).
Elapsed 0.0 of 4.0 minutes, 4.0 remain (3 decisions after this one).
Give each officer a target intersection.
```

The $T = 8$ rows use this text with the horizon and the clock substituted (*Elapsed 0.0 of 8.0 minutes, 8.0 remain*), and the weighted rows use it with the ETAs recomputed under the drawn weights.

C.3 Prompts for Scaling

$n = 30$**, uniform.** One of the sixteen configurations; the others differ only in the numbers. The system message is 1,765 characters and the user message 546; long lines are wrapped for the page, and the model receives each paragraph unwrapped.

```
You are the defence commander of an urban patrol. A criminal is fleeing toward one of the
    city's exits; you command four officers and must catch it.
The city is a 30x30 grid of intersections. Write a position as (row, col), row 0 north, col 0
    west, both 0..29.
Travel time: 1.5 minutes per street segment everywhere -- the city has no express route.
    Officers and the criminal move at the same speed.
Exits: E1 (0, 0), E2 (0, 29), E3 (29, 0), E4 (29, 29).
An officer catches the criminal if it is at the same intersection when the criminal arrives,
     so
    arriving first and waiting is as good as intercepting.
For each officer choose the intersection it should walk to now (it will take the fastest route
    and wait there). Before you answer, check your reply for this mistake: two officers sent
     to
    the same intersection, so that some exit has no officer walking to it. Officers sent to
     the
    same place guard one exit between them; naming an exit in your reasoning does not cover
     it,
    and the exit nobody walks to is the one the criminal escapes through. Give each officer a
    different target, and make sure every exit your own reasoning says must be covered has an
    officer actually walking to it. An officer controls an exit only if it gets there no later
    than the criminal does. If any exit the criminal can reach in time is left to nobody, or
    two officers share one intersection, change the assignment before you answer. The travel
    times given to you below are exact weighted shortest-path times on this map, in minutes;
    use them rather than estimating.
Reply with STRICT JSON only: {"targets": [{"officer": 1, "row": R, "col": C}, {"officer": 2,
    "row": R, "col": C}, {"officer": 3, "row": R, "col": C}, {"officer": 4, "row": R, "col":
    C}], "reasoning": "<one sentence>"}
```

```
Patrol of 4 officers (travel time to each exit, exact minutes):
  officer 1 at (0, 15): E1 22.5, E2 21.0, E3 66.0, E4 64.5
  officer 2 at (15, 0): E1 22.5, E2 66.0, E3 21.0, E4 64.5
  officer 3 at (15, 29): E1 66.0, E2 22.5, E3 64.5, E4 21.0
  officer 4 at (29, 15): E1 66.0, E2 64.5, E3 22.5, E4 21.0
Criminal at (15, 15): E1 45.0, E2 43.5, E3 43.5, E4 42.0.
Criminal's observed trail: (none yet); moving unknown (first observation).
Elapsed 0.0 of 60.0 minutes, 60.0 remain (6 decisions after this one).
Give each officer a target intersection.
```

$n = 200$**, uniform.** The largest size. The prompt is the same size as at $n = 30$. The system message is 1,775 characters and the user message 554; long lines are wrapped for the page, and the model receives each paragraph unwrapped.

```
You are the defence commander of an urban patrol. A criminal is fleeing toward one of the
    city's exits; you command four officers and must catch it.
The city is a 200x200 grid of intersections. Write a position as (row, col), row 0 north, col
     0
    west, both 0..199.
Travel time: 0.2186 minutes per street segment everywhere -- the city has no express route.
    Officers and the criminal move at the same speed.
Exits: E1 (0, 0), E2 (0, 199), E3 (199, 0), E4 (199, 199).
An officer catches the criminal if it is at the same intersection when the criminal arrives,
     so
    arriving first and waiting is as good as intercepting.
For each officer choose the intersection it should walk to now (it will take the fastest route
    and wait there). Before you answer, check your reply for this mistake: two officers sent
     to
    the same intersection, so that some exit has no officer walking to it. Officers sent to
     the
    same place guard one exit between them; naming an exit in your reasoning does not cover
     it,
    and the exit nobody walks to is the one the criminal escapes through. Give each officer a
    different target, and make sure every exit your own reasoning says must be covered has an
    officer actually walking to it. An officer controls an exit only if it gets there no later
    than the criminal does. If any exit the criminal can reach in time is left to nobody, or
    two officers share one intersection, change the assignment before you answer. The travel
    times given to you below are exact weighted shortest-path times on this map, in minutes;
    use them rather than estimating.
Reply with STRICT JSON only: {"targets": [{"officer": 1, "row": R, "col": C}, {"officer": 2,
    "row": R, "col": C}, {"officer": 3, "row": R, "col": C}, {"officer": 4, "row": R, "col":
    C}], "reasoning": "<one sentence>"}
```

```
Patrol of 4 officers (travel time to each exit, exact minutes):
  officer 1 at (0, 100): E1 21.9, E2 21.6, E3 65.4, E4 65.1
  officer 2 at (100, 0): E1 21.9, E2 65.4, E3 21.6, E4 65.1
  officer 3 at (100, 199): E1 65.4, E2 21.9, E3 65.1, E4 21.6
  officer 4 at (199, 100): E1 65.4, E2 65.1, E3 21.9, E4 21.6
Criminal at (100, 100): E1 43.7, E2 43.5, E3 43.5, E4 43.3.
Criminal's observed trail: (none yet); moving unknown (first observation).
Elapsed 0.0 of 60.0 minutes, 60.0 remain (6 decisions after this one).
Give each officer a target intersection.
```

The `freeway` configurations replace the metric sentence with

```
Travel time: 1.5 minutes per street segment, EXCEPT the express row 22 (row index 22), where a
    segment costs 0.3 minutes.
```

rescaled the same way at the other sizes; nothing else in the prompt changes.

C.4 Prompts on Real-World Cases

The interpreter (role 1). The system message is 867 characters and the user message 425 on this record; long lines are wrapped for the page, as everywhere below.

```
You are the context interpreter of a pursuit game played on a real street network. You are
    given one bulletin about a road closure and the list of street names that exist in that
    neighbourhood. Your job is to say which street the bulletin closes, and between which two
    cross streets.

Reply with STRICT JSON only:
{"road": "<street name, exactly as in the list>", "from": "<cross street at one end>", "to":
```

```
    "<cross street at the other end>", "closed": true, "reasoning": "<one sentence>"}

Rules:
- Name streets ONLY as they appear in the list; an abbreviated form in the bulletin (e.g.
    "Bdge") refers to the full name in the list if they share their leading word.
- "from"/"to" must also be streets in the list; use null if the bulletin does not say.
- If the bulletin does not close a street in the list, reply {"road": null, "closed": false,
    "reasoning": "<why>"}.
```

```
Bulletin:
  (t=0) CLOSURE: Governors Bdge between Douglas Cres and Astley Ave is shut. Toronto-TMC:
     These
    closures are in place to allow for the rehabilitation works on the Governor's Road Bridge
    over Mud Creek.

Street names in the neighbourhood (the only roads you may name):
  Astley Avenue, Bayview Heights Drive, Douglas Crescent, Douglas Drive, Glen Road, Governors
    Road, Moorehill Drive, Nesbitt Drive, Standish Avenue
```

It answers `{"road": "Governors Road", "from": "Douglas Crescent", "to": "Astley Avenue", "closed": true}`, which resolves name-first to 13 edges: all 11 of the record's own, plus 2 beyond them — so the world of the CAGTA row of Table 5 is the one this reading builds.

The solver (role 2), CAGTA: the system message. The real-map prompt: named junctions, a 48-junction shortlist, and the travel times as exact minutes. The system message is 3,049 characters and is byte-identical in the blind and informed arms; the count in its first line is the template's, reproduced as sent. Long lines are wrapped for the page and the model receives each paragraph unwrapped, here and below.

```
You are the defence commander of an urban patrol. A criminal is fleeing through a district of
    a
    real city; you command one officers and must catch it.

The district is a network of real streets. Junctions are named, and the travel times below are
    exact driving times in minutes, from the district's own road network.

An officer catches the criminal if it is at the same junction when the criminal arrives.
    Arriving first and waiting is as good as intercepting.

For each officer choose the junction it should walk to now (it will take the fastest route and
    wait there). An officer controls an exit ONLY if the officer's target is that exact
    junction and its given travel time there is <= the criminal's given travel time. Assigning
    an officer to any other junction covers no exit.

Exits: E1 J2 (Glen Road (43.67845, -79.37483)), E2 J4 (Governors Road).

Critical exits are exits with criminal_time <= remaining_time. Every critical exit MUST have
    an
    officer assigned to that exact junction arriving no later than the criminal. If a critical
    exit has only one officer who can arrive on time, that officer MUST take it.

Candidate junctions you may send an officer to (reply with the label):
  J1 (Glen Road (43.68319, -79.37303))
  J2 (Glen Road (43.67845, -79.37483))
  J3 (Roxborough Drive (43.68089, -79.38011))
  J4 (Governors Road)
  J5 (Beaumont Road & Glen Road)
  J6 (Glen Road & Highland Avenue)
  J7 (Bin-Scarth Road & Glen Road)
  J8 (Whitney Avenue & Glen Road)
  J9 (Roxborough Drive & Glen Road)
  J10 (Glen Road (43.68490, -79.37251))
  J11 (Douglas Drive & Glen Road)
  J12 (Beaumont Road (43.68019, -79.37333))
  J13 (Beaumont Road (43.68018, -79.37308))
  J14 (Beaumont Road (43.68010, -79.37260))
  J15 (Beaumont Road (43.68007, -79.37228))
  J16 (Beaumont Road (43.68012, -79.37170))
  J17 (Beaumont Road (43.68015, -79.37128))
  J18 (Beaumont Road (43.68016, -79.37093))
  J19 (Beaumont Road (43.68008, -79.37061))
  J20 (Glen Road (43.68010, -79.37419))
  J21 (node 20954014)
  J22 (node 20954015)
```

```
  J23 (node 20954016)
  J24 (node 20954017)
  J25 (Highland Avenue (43.68194, -79.37696))
  J26 (node 20954019)
  J27 (node 20954020)
  J28 (Roxborough Drive (43.68206, -79.37701))
  J29 (node 20954022)
  J30 (node 20954023)
  J31 (node 20954024)
  J32 (node 20954025)
  J33 (Scholfield Avenue)
  J34 (node 20954027)
  J35 (node 20954028)
  J36 (node 20954029)
  J37 (Roxborough Drive (43.68189, -79.37717))
  J38 (Roxborough Drive (43.68081, -79.38216))
  J39 (Roxborough Drive (43.68090, -79.38174))
  J40 (Roxborough Drive (43.68095, -79.38124))
  J41 (Roxborough Drive (43.68080, -79.37868))
  J42 (Roxborough Drive (43.68083, -79.37841))
  J43 (Roxborough Drive (43.68088, -79.37823))
  J44 (Roxborough Drive (43.68097, -79.37809))
  J45 (Roxborough Drive & Old George Place)
  J46 (Roxborough Drive (43.68185, -79.37721))
  J47 (Highland Avenue (43.68192, -79.37681))
  J48 (Highland Avenue (43.68180, -79.37608))

Reply with STRICT JSON only: {"targets": [{"officer": 1, "junction": "J1"}],
    "reasoning": "<one sentence>"}
```

CAGTA, blind: the opening decision, nothing announced. The user message is 1,249 characters. It carries the one officer's position and its believed ETAs, the criminal, the clock, and the two blocks the digest adds: the critical-exit analysis over the *open* map — both exits critical, each with the same single officer as its only on-time cover — and the conflict paragraph, whose advice is to wait.

```
Patrol of 1 officers (positions):
  officer 1 at J1 (Glen Road (43.68319, -79.37303)): E1 1.1, E2 1.8
Criminal at J3 (Roxborough Drive (43.68089, -79.38011)): E1 1.7, E2 3.1.
Criminal's observed trail: (none yet); moving unknown (first observation).
Elapsed 0.0 of 4.5 minutes, 4.5 remain (6 decisions after this one).
No announced context; no belief in force. ETAs are for the open map.
Critical exit analysis (remaining time = 4.5):
  E1 J2 (Glen Road (43.67845, -79.37483)): criminal 1.7 <= 4.5 -> CRITICAL. Officers on
    time: O1 1.1 ONLY.
  E2 J4 (Governors Road): criminal 3.1 <= 4.5 -> CRITICAL. Officers on time: O1 1.8 ONLY.

Coverage conflict: E1 and E2 are critical and each has only officer 1 as an on-time
    officer; officer 1 cannot be assigned to both. As officer 1 is the only officer who can
    reach either of the two critical exits on time, and we do not know which exit the
    criminal will go to, and the traffic situation may change, it is better to wait for one
    more time step. Assign officer 1 to remain at or target J1 (Glen Road (43.68319,
    -79.37303)) to keep both E1 and E2 reachable after the next update. The other officers
    should take distinct targets that do not conflict with officer 1.
Give each officer a target intersection.
```

The model answers `J1` — it stays — and one decision later, with the criminal observed, it answers `J4`: *"E2 is critical and only officer 1 can reach J4 on time (1.8 <= 2.5), so officer 1 must take J4"*. `E2` is the far side of the shut bridge; the believed 1.8 min route crosses it and no route exists in truth.

CAGTA, informed: the opening decision, the closure announced. The user message is 945 characters: E_2 is ∞ in the belief, so the digest marks it non-critical and names the one mandatory assignment.

```
Patrol of 1 officers (positions):
  officer 1 at J1 (Glen Road (43.68319, -79.37303)): E1 1.1, E2 inf
Criminal at J3 (Roxborough Drive (43.68089, -79.38011)): E1 1.7, E2 inf.
Criminal's observed trail: (none yet); moving unknown (first observation).
Elapsed 0.0 of 4.5 minutes, 4.5 remain (6 decisions after this one).
Announced context:
  (t=0) CLOSURE: Governors Bdge between Douglas Cres and Astley Ave is shut. Toronto-TMC:
    These closures are in place to allow for the rehabilitation works on the Governor's
    Road Bridge over Mud Creek.
Belief in force: Governors Bdge is closed
Critical exit analysis (remaining time = 4.5):
```

```
  E1 J2 (Glen Road (43.67845, -79.37483)): criminal 1.7 <= 4.5 -> CRITICAL. Officers on
    time: O1 1.1 ONLY.
  E2 J4 (Governors Road): criminal inf > 4.5 -> not critical. Officers on time: O1 inf ONLY.

Mandatory: E1 must be assigned to officer 1. All targets must be distinct.
Give each officer a target intersection.
```

It answers `J2` — *"E1 is the only critical exit and officer 1 is the only officer who can reach J2 in time (1.1 <= 1.7), so officer 1 must be assigned to J2"* — and holds the live exit on every route. The two arms' user messages differ in the two lines the belief changes (E_2's ETAs) and in the conclusion the digest draws from them.