Online Decision-Making for Scalable Autonomous Systems

Kyle Hollins Wray\textsuperscript{1,2} and Stefan J. Witwicki\textsuperscript{2} and Shlomo Zilberstein\textsuperscript{1}

\textsuperscript{1}College of Information and Computer Sciences, University of Massachusetts, Amherst, MA 01002
\textsuperscript{2}Nissan Research Center - Silicon Valley, Sunnyvale, CA 94089

wray@cs.umass.edu, stefan.witwicki@nissan-usa.com, shlomo@cs.umass.edu

Abstract

We present a general formal model called MODIA that can tackle a central challenge for autonomous vehicles (AVs), namely the ability to interact with an unspecified, large number of world entities. In MODIA, a collection of possible decision-problems (DPs), known a priori, are instantiated online and executed as decision-components (DCs), unknown a priori. To combine the individual action recommendations of the DCs into a single action, we propose the lexicographic executor action function (LEAF) mechanism. We analyze the complexity of MODIA and establish LEAF’s relation to regret minimization. Finally, we implement MODIA and LEAF using collections of partially observable Markov decision process (POMDP) DPs, and use them for complex AV intersection decision-making. We evaluate the approach in six scenarios within a realistic vehicle simulator and present its use on an AV prototype.

1 Introduction

There has been substantial progress with planning under uncertainty in partially observable, but fully modeled worlds. However, few effective formalisms have been proposed for planning in open worlds with an unspecified, large number of objects. This remains a key challenge for autonomous systems, particularly for autonomous vehicles (AVs). AV research has advanced rapidly since the DARPA Grand Challenge [Thrun \textit{et al.}, 2006], which acted as a catalyst for subsequent work on low-level sensing [Sivaraman and Trivedi, 2013] and control [Dolgov \textit{et al.}, 2010], as well as high-level route planning [Wray \textit{et al.}, 2016a].

A critical missing component to enable autonomy in long-term urban deployments is the middle-level intersection decision-making (e.g., the second-to-second stop, yield, edge, or go decisions). As in many robotic domains, the primary challenges include the sheer complexity of real-world problems, wide variety of possible scenarios that can arise, and unbounded number of multi-step problems that will be actually encountered, perhaps simultaneously. These factors have limited the deployment of existing methods for mid-level decision-making [Ulbrich and Maurer, 2013; Brechtel \textit{et al.}, 2014; Bai \textit{et al.}, 2015; Jo \textit{et al.}, 2015]. We present a scalable, realistic solution, with strong mathematical foundations, via decomposition into problem-specific decision-components.

Our primary motivation is to provide a general solution for AV decision-making at any intersection, including n-way stops, yields, left turns at green traffic lights, right turns at red traffic lights, etc. In this domain, the AV approaches the intersection knowing only the static features from the map, such as road, crosswalk, and traffic controller information. Any number of vehicles and pedestrians can arrive and interact around the intersection, all potentially relevant to decision-making and unknown a priori. The AV must make mid-level decisions, using very limited hardware resources, including when to stop, yield, edge forward, or go, based on all possible interactions among all vehicles including the AV itself. Vehicles can be occluded, requiring the use of information gathering actions based on belief over partial observability. Pedestrians can jaywalk, necessitating that motion forward is taken only under strong confidence they will not cross. Uncertainty regarding priority and right-of-way exists, and must be handled under stochastic changes. Vehicles and pedestrians can block one another’s motion, and AV-related blocking conflicts must be discovered and resolved via motion-based negotiation.

We provide a general solution for domains concerning multiple online decision-components with interacting actions (MODIA). For the particularly difficult AV intersection decision domain, MODIA considers all vehicles and pedestrians as separate individual decision-components. Each component is a partially observable Markov decision process (POMDP) that maintains its own belief for that particular component problem and proposes an action to take at each time step. MODIA then employs an executor function to act as an action aggregator to determine the actual action taken by the AV. This decomposition enables a tractable POMDP solution, benefiting from powerful belief-based reasoning while only growing linearly in the number of encountered problems.

The primary contributions include: a formal definition of MODIA (Section 3), a rigorous analysis of the complexity and regret-minimization properties (Section 4), an AV intersection decision-making MODIA solution (Section 5), and an evaluation of the approach in simulation as well as integration with a real AV (Section 6). We begin with a review of POMDPs (Section 2), and conclude with a survey of related work (Section 7) and final reflections (Section 8).
2 Background Material

A partially observable Markov decision process (POMDP) is represented by the tuple \((S, A, \Omega, T, O, R)\) [Kaelbling et al., 1998]. \(S\) is a finite set of states. \(A\) is a finite set of actions. \(\Omega\) is a finite set of observations. \(T: S \times A \times S \to [0, 1]\) is a state transition function such that \(T(s, a, s') = Pr(s'|s, a)\). \(O: A \times S \times \Omega \to [0, 1]\) is an observation function such that \(O(a, s', \omega) = Pr(\omega|a, s')\). \(R: S \times A \to \mathbb{R}\) is a reward function.

The agent does not observe the true state of the system, and instead makes observations while maintaining a belief over the true state denoted \(b \in \Delta^{|S|}\). Given action \(a \in A\) and subsequent observation \(\omega \in \Omega\), belief \(b\) is updated to \(b'\) with:

\[
b'(s') = \eta O(a, s', \omega) \sum_{s} T(s, a, s') b(s) \quad \text{for all } s' \in S,
\]

with normalizing constant \(\eta\). A policy maps beliefs to actions \(\pi: \Delta^{|S|} \to A\). (Note: \(\Delta^n\) is the standard n-simplex.)

The value function \(V: \Delta^{|S|} \to \mathbb{R}\) for a belief is the expected reward given a fixed policy \(\pi\), a discount factor \(\gamma \in [0, 1]\), and a horizon \(h\). Also, it is useful to define the Q-value of belief \(b\) given action \(a\) as \(Q: \Delta^{|S|} \times A \to \mathbb{R}\) with \(Q(b, \pi(a))\). Since \(V^n\) is piecewise linear and convex, we describe it using sets of \(\alpha\)-vectors \(\Gamma = \{\alpha_1, \ldots, \alpha_p\}\) with each \(\alpha_i = [\alpha_i(s_1), \ldots, \alpha_i(s_n)]^T\) and \(\alpha_i(s)\) denoting value of state \(s\) in \(S\). The objective is to find optimal policy \(\pi^*\) that maximizes \(V\) denoted as \(V^*\). Given an initial belief \(b_0\), \(V^*\) can be iteratively computed for a step time \(t\), expanding beliefs at each update resulting in belief \(b\) by maximizing:

\[
Q^t(b, a) = \sum_{s \in S} b(s) R(s, a) + \sum_{\omega \in \Omega} \max_{ \alpha \in \Gamma} \sum_{s \in S} b(s) V^t_{s | \omega \alpha}
\]

and \(V^t_{s | \omega \alpha} = \gamma \sum_{s' \in S} O(a, s', \omega) T(s, a, s') V^t_{s' | \omega \alpha} - R(s)\) for \(s \in S\), \(\alpha^0(s) = R/(1 - \gamma)\) in \(\Gamma^0 = \{\alpha^0\}\) with \(R = \min_{s, a} R(s, a)\).

3 Problem Formulation

We begin with a general problem description that considers a single autonomous agent that encounters any number of decision processes online during execution. This paper focuses on collections of POMDPs primarily for their general form, self-consistency, and space limitations. It can be generalized to other decision-making models in the natural way. Finally, Figure 1 depicts a complete MODIA example for AVs, and is referenced throughout this section for each concept.

3.1 Decision-Making with MODIA

The multiple online decision-components with interacting actions (MODIA) model describes a realistic single-agent online decision-making scenario defined by the tuple \(\langle \mathcal{P}, \mathcal{A}\rangle\). \(\mathcal{P} = \{P_1, \ldots, P_k\}\) are decision-problems (DPs) that could be encountered during execution. For this paper, each \(P_i \in \mathcal{P}\) is a POMDP with \(P_i = \{S_i, A_i, \Omega_i, T_i, O_i, R_i\}\) (Section 2) starting from an initial belief \(b_0 \in \Delta^{|S_i|}\). We consider discrete time steps \(t \in \mathbb{N}\) over the agent’s entire lifetime. \(\mathcal{A} = \{a_1, \ldots, a_n\}\) are \(z\) primary actions that are the true actions taken by the agent that affect the state of the external system environment. Importantly, only \(\mathcal{P}\) and \(\mathcal{A}\) are known offline a priori.

AV Example Figure 1 has two pre-solved intersection decision-components: single vehicle (\(P_1\)) or pedestrian (\(P_2\)). Each are POMDPs with actions (recommendations) ‘stop’ or ‘go’. Primary actions \(A\) for the AV are also ‘stop’ or ‘go’.

Online, the DPs are instantiated based on what the agent experiences in the external system environment. Due to the nature of actually executing multiple decision-making models (e.g., POMDPs) in real applications, there is no complete model for which, when, or how many DPs are instantiated, or even how long they are relevant.

Formally, the online instantiations in MODIA are defined by the tuple \(\langle \mathcal{C}, \phi, \tau\rangle\). Over the agent’s lifetime, there are \(n\) DP instantiations called decision-components (DCs) denoted as \(C = \{C_1, \ldots, C_n\}\), with both \(C\) and \(n\) unknown a priori. Let \(\phi: C \to \mathcal{P}\) denote the DP for each instantiation. Let \(\tau: \mathcal{C} \to \mathbb{N} \times \mathbb{N}\) be the two time steps that each DC is instantiated and terminated. For notational convenience, for all \(C_i \in C\), let \(\tau_s(C_i)\) and \(\tau_{en}(C_i)\) be the start and end times; we have \(\tau_s(C_i) < \tau_{en}(C_i)\). Without loss of generality, we also assume for \(i < j\), \(\tau_s(C_i) \leq \tau_s(C_j)\). We call a DC \(C_i \in C\) instantiated at time step \(t \in \mathbb{N}\) if \(t \in [\tau_s(C_i), \tau_{en}(C_i)]\). Any instantiated \(C_i \in C\) includes POMDP \(\phi(C_i)\), its policy \(\pi_i: \Delta^{|S_i|} \to A_i\), and its current belief state \(b^i_{t} \in \Delta^{|S_i|}\) with local POMDP time step \(t_i = t - \tau_s(C_i)\).

AV Example (Continued) Online, the AV encounters an intersection and immediately (at time step 1) observes two vehicles and one pedestrian. Three DCs are instantiated; \(C_1\) and \(C_2\) are for each vehicle (\(\phi(C_1) = \phi(C_2) = P_1\)), and \(C_3\) is for the pedestrian (\(\phi(C_3) = P_2\)). The start times for all \(C_i\) are \(\tau_s(C_i) = 1\); the end times \(\tau_{en}(C_i)\) are still unknown. Each POMDP \(C_i\), with \(\phi(C_i) = P_j\): \(b^0_i = b^j_0\), \(t^i_1 = 1\), and \(\pi_i = \pi_j\).

3.2 The MODIA Executor

With DPs and primary actions \(\langle \mathcal{P}, \mathcal{A}\rangle\) (known a priori), and online execution of DCs \(\langle \mathcal{C}, \phi, \tau\rangle\) (unknown a priori), the primary actions taken from \(\mathcal{A}\) are determined by an action executor function \(\epsilon: \mathcal{A} \to \mathcal{A}\) with \(\mathcal{A} = \{\bigcup \mathcal{A}_i\}\). (Note: \(X^*\) is a Kleene operator on a set \(X\), and \(A_i\) is the set of actions for the POMDP from DP \(P_i\).) The executor takes DC action recommendations and converts them to a primary action taken by the agent in the external system environment. It also converts a primary action back to what that decision meant to individual DCs via their action sets. In this paper, we use the notation \(\epsilon^+(a)\) referring to an individual\( C_i\)’s action from POMDP \(\phi(C_i)\) for some \(a \in \mathcal{A}\).

It is important to note the requirement that the executor function \(\epsilon\) must be able to map any tuple of actions taken from any combination of DPs, with any number of possible duplicates, to a primary action. MODIA is a class of problems that operates without any knowledge about which (or how many) DPs will be instantiated online.

AV Example (Continued) In Figure 1, all three DCs produce an action \((a_1, a_2, a_3) = \bar{a} \in \bar{A}\) at each time step. The example states \(a_1 = a_3 = \text{stop}\) and \(a_2 = \text{go}\). The executor \(\epsilon\) decides from \(\bar{a}\) that \(\text{stop} \in \mathcal{A}\) will be the primary action. It informs each DC \(C_i\) what the primary action means to \(C_i\) individually, simply \(\epsilon^i_{\text{stop}}(\text{stop}) = \text{stop}\), for belief updates.
3.3 The MODIA Objective

The goal of the class of problems captured by MODIA is to design the DPs, primary action set, and executor so that it solves the online real-world problem (e.g., AVs). Prior work on single-POMDP online algorithms experimentally analyze their performance with simpler metrics such as average discounted reward (ADR) or run time [Somani et al., 2013; Kurniawati and Yadav, 2016], and richer metrics such as error bound reduction (EBR) or lower bound improvement (LBI) [Ross et al., 2008]. MODIA is an online multi-POMDP model that differs from these previous online single-POMDP solvers. We instead provide a concrete objective function to enable the analysis of this complex online problem within a theoretical context. Our problem domain does not contain a model for how DPs are instantiated as DCs, nor how long DCs remain active. Thus, the objective is to minimize regret experienced at each step for any given DC instantiations.

Formally, for \((P, \mathcal{A}, \mathcal{C}, \phi, \epsilon)\), let \(h \leq \tau_{C_i}(C_n)\) be a horizon, let \(I^t = \{i \in \{1, \ldots, n\} : \tau_{C_i}(C) \leq t \leq \tau_{C_i}(C)\}\) denote the set of indexes for instantiated DPs, and let executor decision \(\epsilon(\bar{a}) = a'\) at time \(t \in \{1, \ldots, h\}\) with primary action \(a' \in \mathcal{A}\) and the tuple of all instantiated DC\'s actions \(\bar{a} \in \mathcal{A}\), so for all \(i \in I^t\), \(a_i = \pi_i(b_{i,i})\) with \(\pi_i\), \(t_i\), and \(b_{i,1}\) from instantiated DC\'s \(C_i \in \mathcal{C}\). The total regret \(R^h_t \in \mathbb{R}\) is:

\[
R^h_t = \sum_{i=1}^{I^t} \sum_{j=1}^{n} Q_i(b_{i,j}, \pi_i(b_{i,j})) - Q_i(b_{i,j}, c_{i}^{-1}(a')).
\]

We refer to the regret at time \(t\) for all instantiated DC\'s in \(I^t\) as \(r^t\). Informally, a DC\'s regret in MODIA is the expected reward following the DC\'s desired policy\’s action, minus the realized expected reward following the executor\’s action.

AV Example (Continued) Executor \(\epsilon\) selected \(\text{stop} \in \mathcal{A}\), which has \(c_1^{-1}(\text{stop}) = \text{stop}\) for all \(C_i \in \mathcal{C}\). Following each DC\'s desired action, only \(C_2\) chose \(\text{go}\) instead. This induces regret equal to \(Q_2(b_2^2, \text{go}) - Q_2(b_2^2, \text{stop}) \geq 0\); \(C_1\) and \(C_3\) have zero regret. \(R^h_t\) is updated accordingly.

3.4 LEAF for MODIA

So far we have described the general form of MODIA using a general executor. Now we examine a particular kind of executor with desirable regret-minimizing properties (shown in Section 4). Specifically, we can define a lexicographic preference over the individual actions suggested by each DC. Thus, each DC suggests an action, stored collectively as a tuple of action recommendations, and the executor only executes the best (in terms of preference) action from this set.

A lexicographic executor action function (LEAF) has two requirements regarding a MODIA\'s structure in \((P, \mathcal{A})\). First, let the primary actions \(\mathcal{A}\) be factored with the unique action sets from among the DPs; formally, \(\mathcal{A} = X_i \Lambda_i\) with \(\Lambda = \bigcup_j \{A_j\}\). Second, let \(\succ_{i}\) be a lexicographic ordering over actions in these unique action sets \(\Lambda_i \in \Lambda\). If a MODIA satisfies these two requirements, then for all \(\bar{a} = (a_1, \ldots, a_y) \in \bar{A}\) and \(a = (a_1, \ldots, a_y) \in \mathcal{A}\), LEAF \(\epsilon(\bar{a}) = a\) is defined by:

\[
a_i \succ_{i} a_i, \quad \forall a \in \{a' : \mathcal{A}_j \mid j \in A\} \text{ s.t. } a_j = a'
\]

(2)

for all \(\Lambda_i \in \Lambda\), and \(\epsilon(b) = a\) for some fixed \(a \in \mathcal{A}\). Informally, \(a\) are the current desired actions from DC\'s, \(\Lambda_i\) is the unique action set, \(\bar{a}\) are the resulting actions, and each \(a_i\) (from matching unique action set \(\Lambda_i\)) has the highest preference following \(\succ_{i}\) from the available voted-upon actions. Similarly, the inverse executor extracts the relevant action factor taken by the system and distributes it to all DC\'s who have that action set; formally, for all \(C_i \in \mathcal{C}\), with \(\phi(C_i) = P_i\), there exists an action \(a_i \in \mathcal{A}_j = A_{i}\) such that for the primary action taken \(a \in \mathcal{A}\), \(c_{i}^{-1}(a) = a_i\). In summary, LEAF simply takes the most preferred action among those available.

AV Example (Continued) In the AV example, we have action sets \(\{\text{stop}, \text{go}\} = \Lambda_1 = \Lambda_2 = \mathcal{A} = \Lambda_1\). Thus, it satisfies the first requirement: primary actions are composed of DP actions. For the second, we define a lexicographic preference \(\succ_{1}\) (encouraging safety) over \(\Lambda_1\) with \(\text{stop} \succ \text{go}\). Now \(\epsilon\) in Figure 1 is actually LEAF. Namely, the action \(\text{stop}\) is the most preferred action desired among only the actions selected by the DC\'s. Thus, \(\text{stop}\) is the result of the executor.
3.5 Risk-Sensitive MODIA

Now we also consider a specific kind of MODIA, with a form of monotonicity in an ordered relationship over actions and Q-values. Informally, we require DP’s Q-values to be monotonic over actions with a penalty for selecting policy-violating high-risk actions. Formally, a MODIA is risk-sensitive with respect to a preference \( \succ \), if for all \( b, a, a' \): (1) if \( a \succ a \geq \succ \pi_j(b) \) then \( Q_j(b, a) \leq Q_j(b, a') \), (2) if \( \pi_j(b) \succ a \) then \( Q_j(b, a) \leq Q \) for sufficient penalty \( Q \).

AV Example (Continued) Action stop makes no progress towards the goal while go does, so long as go is optimal, resulting in (1). Conversely, performing go when stop is optimal produces a severe expected cost, resulting in (2).

4 Theoretical Analysis

Given DPs and primary actions \((P, A)\), MODIA requires the selection of an executor to minimize regret accumulated over time, in addition to solving the DPs themselves. With \( n \) unknown a priori, as well as which and when DPs are instantiated as DCs, it is impossible to perform tractable planning techniques entirely offline; again, MODIA is an category of online decision-making scenarios. Assume, however, that a prescient oracle provided \((C, \phi, \tau)\) a priori. While this is an impossible scenario, it is useful to understand the worst-case complexity of exploiting this information in the underlying problem of selecting a regret-minimizing executor given this normally unobtainable information. Proposition 1 formally proves this complexity.

**Proposition 1.** If \((C, \phi, \tau)\) is known a priori, then the complexity to compute the optimal executor \( \epsilon^* \) is \( O(n^2 z mn) \) with \( z = |A|, m = \max_i |A_i| \), and \( h = \max_i \tau_i(C_i) \).

**Proof.** Must determine the worst-case complexity to fully define executor \( \epsilon^* : A \rightarrow A \) to minimize regret Equation 1. In the worst-case, we must explore all relevant executors, and compute the regret for each, resulting in the optimal solution.

By executor definition in Section 3.2, \( \bar{A} = (\bigcup_i A_i) \) and \( z = |A| \). Given \( n = |C| \), the maximum realizable set size of \( \bar{A} \) is all unique potential actions, multiplied by the maximal number of unique DCs instantiated simultaneously. In the worst-case, \( A_i \neq A_j \) for all \( i \neq j \), so all possible actions must be considered for each; this order bound is \( m = \max_i |A_i| \). Also, all combinations of instantiated DCs must be realized, so all \( \tau(C_i) \neq \tau(C_j) \) for all \( i \neq j \). In any order, \( n \) births, \( n \) deaths, and time no DCs instantiated; thus there are \( 2n+1 \) in total. Hence, the number of potential executors is \( O(zmn) \).

In the worst-case scenario, \( R_i(b_i^j, \pi_j(b_i^j)) \) differs for every time step for all \( C_i \in C \). Equation 1 requires \( O(h \pi_i(I_i)) \) operations. Given \( C, h = \max_i \tau_i(C_i) \). By definition of \( I_i \), \( \max_i I_i \leq n \). Thus, the worst-case complexity to compute an optimal \( \epsilon^* \) is \( O(zmn) \cdot O(hn) = O(n^2 z mn) \).

With Proposition 1, we know this impossible oracle scenario’s complexity is relatively high, but not exponential. This suggests a method for computing an optimal executor, under more realistic assumptions. Thus, let \( \rho \) be a given model for the hardest feature of MODIA: online instantiation. Let \( \rho : N_n \times T_n \times E_n \times N_n \times T_n \rightarrow [0,1] \) define the probability that a particular set of instantiated DCs \((n, \tau) \in N_n \times T_n \) and executor selection \( \xi \in E_n \), results in a successor DC instantiation state \((n', \tau') \in N_n \times T_n \). Here, \( N_n = \{1, \ldots, k\} \) are instantiation indexes (defining \( \phi \)), \( T_n = \{t \in \{\alpha, 1, \ldots, h\}^n \mid |i| \in N, \tau_i < \tau_e \} \) are the instantiation start and end times (defining \( \tau \)) including non-instantiated \( \alpha \) and completed \( \omega \) demarcations, and \( E_n = \{e : A \rightarrow A\} \) are valid executors (defining \( e \)). Additionally, we must assume knowledge of a maximum number of DCs \( n \) and horizon \( h \) for decidability. Given this model, Proposition 2 proves the resulting MDP’s optimal policy minimizes expected regret, and that the problem is unfortunately computationally intractable in practice.

**Proposition 2.** If \( n, h \), and model \( \rho \) are known a priori, then: (1) the resulting MDP’s optimal policy \( \pi^\star \) minimizes expected regret, and (2) its state space is exponential in \( n \) and \( k \).

**Proof.** We must show the construction of an MDP whose optimal policy minimizes expected regret and show its complexity in the necessity of an exponential state space.

Let \( (S, A, T, R) \) be a finite horizon MDP with horizon \( h = h + 1 \). States are \( S = \{s_0\} \cup E_n \times B_n \times T_n \), with \( s_0 \) denoting the initial executor selection state and \( B_n = \{B \in \bigcup B^h \mid B = n\} \) all possible reachable beliefs for \( \mathcal{P}_i \) in horizon \( h \) (denoted \( B_{\pi}^h \)) for all possible instantiations. As mentioned, we use \( s = (\hat{e}, \hat{b}, \hat{\tau}) \), each containing instantiated values \( \hat{e}_i, \hat{b}_i, \tau_{xi} \), and \( \tau_{xi} \), as well as \( \tilde{\tau} : \hat{B}_n \rightarrow N_n \) mapping beliefs to their original POMDPs’ indices. Actions are executor selection \( A = E_n \). State transitions \( T : \hat{s} \times A \rightarrow \hat{s} \rightarrow [0,1] \) have two cases. First, \( T(s^0, \hat{a}, \hat{s}') = [s' = (\hat{a}, 0, 0)] \) captures executor selection. Second, for \( \hat{s} \neq s^0 \) we have:

\[
\hat{T}(\hat{s}, \hat{a}, \hat{s}') = [(\hat{s} = s^0 \wedge \hat{e}' = \hat{a}) \lor (\hat{s} \neq s^0 \wedge \hat{e}' = \hat{e})] \cdot \hat{\rho}(\hat{\theta}(\hat{b}), \hat{\tau}, \hat{e}, \hat{\theta}(\hat{b}'), \hat{\tau}') \prod_{i=1}^n [\hat{b}'_i = \hat{b}_i^j \wedge \hat{\tau}_i = \hat{\tau}_i + 1] \prod_{i=1}^n Pr(\hat{b}_i | \hat{b}_i, \pi_j(\hat{b}_i)) [\hat{\tau}_i \in N \wedge \hat{\tau}_i + 1] \prod_{i=1}^n [\hat{b}'_i = \omega \wedge \hat{b}_i = \hat{b}_i]
\]

with \( j = \theta_i(\hat{b}) \). This captures executor state assignment, the instantiation model \( \hat{\rho} \), the proper initialization of belief, the belief update for active DCs, and the termination of a DC. Rewards \( \hat{R} : \hat{S} \times A \rightarrow \mathbb{R} \) describe the negative regret, \( \hat{R}(\hat{s}, \hat{a}) = \sum_i Q_i(\hat{b}_i, \pi_i(\hat{b}_i)) - Q_i(\hat{b}_i, \pi_j(\hat{b}_i)) \hat{\tau}_i \in N \) with \( \hat{R}(s^0, \hat{a}) = 0 \). By construction, this is MODIA, assuming \( \hat{a}, n \), and \( h \) were provided. By assigning \( \epsilon = \pi^\star(s^0) \), we minimize expected regret. In the worst-case, it necessitates modeling all \( n \) DC instantiation permutations (with replacement) of the \( k \) DPs, which is \( O(k^n) \).

This illustrates the importance of the original MODIA formulation. Even with the instantiation model of Proposition 2, the problem is still unscalable. And the knowledge needed to bound the number of active DCs (e.g., \( n \) and \( h \)) is generally unavailable a priori. This intrinsic lack of information motivated our formulation that minimizes the regret at each time step. Hence, the agent is guided by the optimal
DC policies from each instantiated DP, selecting the regret-minimizing action at each time step. Proposition 3 proves that LEAF minimizes the regret in risk-sensitive MODIA at each time step, enabling a tractable solution to MODIA.

**Proposition 3.** If a MODIA is risk-sensitive, then LEAF minimizes regret $r_t^\epsilon$ for all $t$.

**Proof.** By definition of regret $r_t^\epsilon$ for LEAF $\epsilon$ at time step $t$: 
$$r_t^\epsilon = \sum_{i \in I_t} Q_i (b_t, \pi_i (b_t^i)) - Q_j (b_t, \pi_j (a_t^j))$$
with $\phi(C_i) = P_j$.
We must show for all $\epsilon$, $r_t^\epsilon \leq r_t^\epsilon$. For readability, hereafter, let $a_i = e_i^1 (a_t^i)$, $a_i = e_i^1 (a_t^i)$, $a_j = \pi_j (b_t^j)$, and $b_i = b_t^i$. By definition of risk-sensitive, there always exists action $a_j^i$ such that $Q_j (b_t, a_j^i) \geq Q$. Thus, it is sufficient to show that for all $i \in I_t$, $Q_j (b_t, a_j) \geq Q_j (b_t, a_i)$, or there exists $C_i \epsilon C$ such that $Q_j (b_t, a_i) \leq Q$. By risk-sensitivity and LEAF, consider 3 cases for $\epsilon$ and $\epsilon$.

**Case 1:** $a_i = a_j$, for $a_i, a_i \in A_x = A_j$. Trivially, we have $Q_j (b_t, a_j) = Q_j (b_t, a_i)$.

**Case 2:** $a_i \not\sim a_j$ has two cases. 

**Case 2.1:** If $a_i = a_j$, then by definition $\pi_j$’s optimality, for any $a_i \in A_j$, $Q_j (b_t, a_j) = Q_j (b_t, a_j) \geq Q_j (b_t, a_j)$. 

**Case 2.2:** If $a_i \not= a_j$, then by LEAF Equation 2, $a_i \in A \epsilon A \epsilon a_i = a_i$. Thus, by definition of $a_i \epsilon A$, this $a_i \epsilon a_i$ such that $a_i = a_j$, or $Q_j (b_t, a_j)$. By risk-sensitivity, $a_i = a_i = a_i \not\sim a_j$ that implies $Q_j (b_t, a_j) \leq Q$.

**Case 3:** $a_i \not\sim a_j$. By definition of risk-sensitivity, we have $a_i \not\sim a_i, a_i \not\sim a_j$ and consequently $Q_j (b_t, a_i) \leq Q_j (b_t, a_j)$.

All cases proven. LEAF minimizes regret $r_t^\epsilon$ for any $t$. □

5 **Application to Autonomous Vehicles**

We apply MODIA and LEAF to this concrete problem of AV decision-making at intersections. The formulation expands on the numerous AV examples described in Section 3. Due to space considerations, we focus our attention strictly on defining vehicle-related DP (POMDP); however, pedestrian and other DPs follow in a similar manner. Overall, this AV robotic application serves to both ground our theoretical work and simultaneously present an actual solution to intersection decision-making in the real world.

The MODIA AV ($P, A$) defines $P$ by converting intersection types (and pedestrian types) into POMDP DP. These types capture the static abstracted information. For example, intersection types contain features such as the number of road segments, lane information (incoming and outgoing), crosswalk locations, and traffic controller information. A DP is created for all lanes within all intersection types (and pedestrian types). Formally, for each such vehicle and intersection type, we define the DP POMDP ($S_i, A_i, \Omega_i, T_i, O_i, R_i$) = $P_i \in P$. $S_i$ = $S_{av} \times S_{av} \times S_{av} \times S_{av} \times S_{av} \times S_{av}$ describes the AV’s location (approaching/at/edged/inside/goal) and time spent at location (short/long), as well as the other vehicle’s location (approaching/at/edged/inside/empty), time spent at location (short/long), blocking (yes/no), and priority at intersection in relation to AV (ahead/behind), respectively. Actions are simply $A_i = \{stop, edge, go\}$, and encode movement by assigning desired velocity and goal points along the AV’s trajectory within the intersection. Lower-level nuances in path planning [Wray et al., 2016b] are optimized by other methods. $\Omega_i$ = $\Omega_{av} \times \Omega_{av} \times \Omega_{av} \times \Omega_{av} \times \Omega_{av} \times \Omega_{av}$, primarily encode the noisy sensor updates in blocking detection (yes/no) but also if the time spent was updated (yes/no) for both the AV and other vehicle. $T_i : S_i \times A_i \times S_i \rightarrow [0, 1]$ multiply the probabilities of a wide range of situations quantifiable and definable in the state-action space described. This includes multiplying probabilities for: (1) vehicle kindly lets AV have priority, (2) vehicle cuts AV off, (3) AV’s success or failure of motion to an abstracted state based on its physical size, (4) a new vehicle arrives at an intersection lane, (6) time increments, (7) vehicle actually stops at stop sign or does a rolling stop, (8) vehicle is blocking the AV’s path following the static intersection type’s road structure, etc. Additionally, a dead end state (an absorbing non-goal self-loop) is reached when the AV and other vehicle both have state factor “inside” while also “blocking” each other. $O_i : S_i \times A_i \rightarrow O_i \rightarrow [0, 1]$ is defined as unit cost for all states, except the goal state.

The primary actions are $A = \{stop, edge, go\}$ and simply describe the AV’s movement along the desired trajectory. We define a lexicographic preference > over this action set stop > edge > goal. This preference formalizes the notion that if even one DC said to stop, then the AV should stop. Similarly, if at least one DC said to edge but none said stop, then the AV should cautiously edge forward. Otherwise, the AV should go. This enables us to apply LEAF because $A_i = A$ for all $A_i$ (even the pedestrian DPs) and we have lexicographic preference >. Lastly, the defined MODIA produces $Q$-values that satisfy risk-sensitivity.

6 **Experimentation**

We begin with experiments on six different intersections in an industry-standard vehicle simulation developed by Realtime Technologies, Inc. that accurately simulates vehicle dynamics with support for ambient traffic and pedestrians. We evaluated MODIA on real map data at six different intersections, each highlighting a commonly encountered real-world scenario. Table 1 describes each scenario by name and provides details regarding the road segments, vehicles, and pedestrians that exist. The number of potential incidents describes how many risks exist, which MODIA perfectly obviates. We compare a MODIA AV with ignorant and naive AV baseline algorithms. The ignorant AV follows the law but ignores the existence of all vehicles and pedestrians, acting as if the intersections are empty. The naive AV follows the law and cautiously waits until all others have cleared the intersection beyond 15 meters before attempting to go. These two baselines implement extremes of rule-based AVs [Jo et al., 2015] and serve as a form of bound for AV behavior to understand MODIA AV’s performance. We evaluate each by their time to complete an intersection, which includes the observations while approaching, decisions at the intersection, and travel within the intersection. In Table 1, we observe the MODIA AV successfully completes intersections faster than the cau-
tious naive AV. While the MODIA AV takes longer than the ignorant AV, the ignorant AV encounters each potential incident and the MODIA AV safely avoids them.

Figure 2 depicts a common 4-way intersection with our fully-operational AV prototype, which operates on real public roads and contains an implementation of MODIA and LEAF.

7 Related Work

Previous work on an general models related to MODIA include architectures for mobile robots [Brooks, 1986; Rosenblatt, 1997] or other systems [Decker, 1996], and contain decision-components that produce actions, aggregated to a system action. They do not, however, naturally model uncertainty or have a general theoretical grounding. Forms of hierarchies include action-based execution of child problems with multi-options [Barto and Mahadevan, 2005] and abstract machines [Parr and Russell, 1998]. Action-space partitioning that execute smaller MDPs [Hauskrecht et al., 1998] and POMDPs [Pineau et al., 2001] also exists. These do not model the online execution of an unknown number of decision-components for use in robotics. More application-focused work on action voting for simple POMDPs to solve intractable POMDPs have been used successfully [Yadav et al., 2015]. Robotic applications of hierarchical POMDPs for an intelligent wheelchair decompose the problem into components [Tao et al., 2009], or with two POMDP levels for vision-based robots [Sridharan et al., 2010]. These practical methods work well but lack generalized mathematical foundations. Also, none of these present AV-specific solutions.

Previous work specific to AV decision-making includes simple rule-based or finite-state controller systems [Jo et al., 2015], which are simple to implement but are brittle, difficult to maintain, and were unable to handle the abundant uncertainty in AV decision-making. Initial attempts using deep neural networks map raw images to control [Chen et al., 2015] are slow to train and tend to fail rapidly when presented with novel situations. Mixed-observability MDPs for pedestrian avoidance also successfully use a decision-component approach (AV-pedestrian pairs) but provide limited theoretical work and do not extend to intersections [Bandopadhyay et al., 2013]. Using a single POMDP for all decision-making has been explored, including continuous POMDPs using raw spacial coordinates for mid-level decision-making [Brechtel et al., 2014], online intention-aware POMDPs for pedestrian navigation [Bai et al., 2015], and POMDPs for lane changes that use online approximate lookahead algorithms [Ulbrich and Maurer, 2013]. These approaches do not address the exponential complexity concerns (scalability), provide generalizable theoretical foundations, or enable simultaneous seamless integration of multiple different decision-making scenarios on a real AV, all of which are provided by MODIA.

8 Conclusion

MODIA is a principled theoretical model designed for direct practical use in online decision-making for autonomous robots. It has a number advantages over the direct use of a massive monolithic POMDP for planning and learning. Namely, it remains tractable by growing linearly in the number of decision-making problems encountered. Its component-based form simplifies the design and analysis, and enables provable theoretical results for this class of problems. MODIA is shown to successfully solve a challenging AV interaction problem. Future work will explore more executors and models beyond LEAF and risk-sensitive MODIA, develop additional AV-related DPs, and tackle other intractable robotic domains such as humanoid service robots using MODIA as a scalable online decision-making solution.

Acknowledgments

We thank the reviewers for their helpful comments, Liam Pedersen for valuable discussions, and Nissan Motor Co., Ltd. for supporting this work.
References


