

POMDP: Acting on a Belief, Not a Guess

Amp

An MDP hands the agent the state; a POMDP hides it and hands back an observation. This note follows one thread through that change: the agent's knowledge becomes a probability distribution over states, Bayes's rule moves the distribution, and every decision is read off it. The first half builds the belief, shows why acting on the most likely state can be far worse than paying to observe, and writes the belief-space Bellman equation whose search tree explodes. The second half is the part an operational planner actually keeps: the factored transition that turns one joint model into an ego model times a driver model per other agent – which makes prediction the belief-update step inside planning – and the guided observation branching that shrinks the belief tree enough to search in real time. The application is the POMDP inside the EPSILON driving planner.

October 08, 2026

Contents

1. The state is gone	3
2. A belief, and the rule that moves it	4
3. Why a guess is not enough	5
4. The tree the update grows	6
5. Many agents, one factored transition	7
6. From a policy tree to one decision	9
7. Making it real-time: guided observation branching	10
8. What is exact, and what was traded away	11
9. Sources and further reading	12
Bibliography	13

1. The state is gone

A Markov decision process gives the agent a state s , a transition model $T(s'|s, a)$, a reward $R(s, a)$, and the assumption that the agent **sees** s at every step. Optimal behavior is then a function of the state: a policy $\pi(s)$, and a value $V(s)$ that the policy tries to maximize. A POMDP deletes exactly one thing – the sight of the state – and keeps everything else.

The smallest problem in which that deletion matters is the **tiger problem**. Two doors stand in front of the agent; behind one is a tiger, behind the other a reward. The agent cannot see behind the doors. It can open a door, or it can listen for a growl. Listening is cheap and informative but imperfect.

ACTION	REWARD	OBSERVATION
open left	+10 if the tiger is right, −100 if it is left	none
open right	+10 if the tiger is left, −100 if it is right	none
listen	−1	growl from the left with 0.85 if the tiger is left, else 0.15

Listening leaves the world alone. Opening a door ends the episode and the tiger is placed again at random, so after opening a door the agent knows nothing about the next one.

Hook: At the moment the agent must act, what does it actually know? It has never seen the state. Every observation is in the past, and the state itself is unobservable. So the agent's knowledge is **not** a state – it is a set of probabilities over states. What happens to a policy when the object it reads is a distribution rather than a point?

2. A belief, and the rule that moves it

Formally a POMDP is a tuple $\langle X, A, Z, T, O, R \rangle$: states X , actions A , observations Z , a transition model $T(x_{t-1}, a_t, x_t) = p(x_t | x_{t-1}, a_t)$, an observation model $O(x_t, z_t) = p(z_t | x_t)$, and a reward $R : X \times A \rightarrow \mathbb{R}$. Because the state is not seen, the agent carries a **belief** b , a probability distribution over X , and moves it after acting and observing. Bayes's rule gives the move:

$$b_{t(x_t)} = \eta O(x_t, z_t) \int_{x_{t-1} \in X} T(x_{t-1}, a_t, x_t) b_{t-1}(x_{t-1}) dx_{t-1}. \quad (1)$$

Read the three pieces in order. The prior b_{t-1} is pushed forward by the motion model T to say where the state probably went; the observation model O scores each candidate state by how well it explains what was actually seen; the normalizer η makes the result a distribution again. Every POMDP belief update is this one equation [1].

With two states, one number is enough. Write $b = P(\text{tiger is left})$, so $1 - b$ is the probability of the other case and the possible beliefs form the interval $[0, 1]$ – the **belief simplex** for two states. Listening does not move the tiger, so the observation model alone updates the belief. A growl from the left arrives with probability $P(\text{growl left}) = 0.85b + 0.15(1 - b) = 0.15 + 0.7b$ and leaves

$$b' = \frac{0.85b}{0.15 + 0.7b}, \quad b'' = \frac{0.15b}{0.85 - 0.7b}. \quad (2)$$

Start from $b = 0.6$: a growl on the left arrives with probability 0.57 and moves the belief to $b' = 0.8947$; a growl on the right arrives with probability 0.43 and moves it to $b'' = 0.2093$. The two numbers 0.8947 and 0.2093 – not the growl, and not the old 0.6 – are the agent's new knowledge. That is the object a policy has to read.

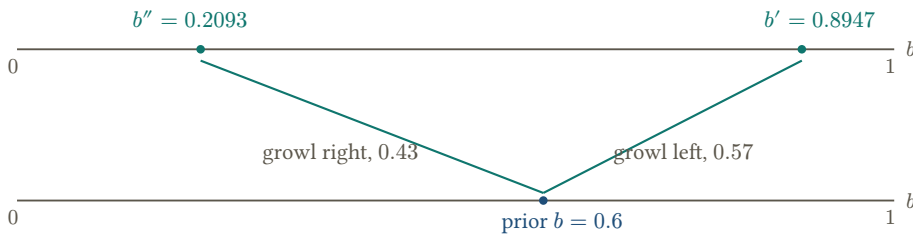


Figure 1: The belief update of Equation 2 as a map. The prior sits on the lower axis; each possible growl moves it to one point on the upper axis. The state never appears, only these two candidate posteriors and the probabilities that choose between them.

Two facts about b carry the rest of this note. First, b_t is a **sufficient statistic** for the whole history: given b_t , no earlier action or observation changes the best thing to do, which is why a policy may be written $\pi(b)$ alone, and why the belief follows a Markov process of its own. Second, the integral in Equation 1 runs over the whole state space: only in small models is it a hand computation, and at scale it is approximated – by a particle filter, by an assumed family, or, as the second half of this note shows, by freezing the expensive part. The sidecar `pomdp-demo.html` in this folder lets you drag b and watch the update move it.

3. Why a guess is not enough

The tempting shortcut is to act on the larger of b and $1 - b$. Here is what it costs. Opening the left door yields +10 only when the tiger is right, so its expected value is $10 - 110b$; opening the right yields $110b - 100$. The best immediate action is therefore

$$A(b) = \max(10 - 110b, 110b - 100) = -45 + 110 \left| b - \frac{1}{2} \right|. \quad (3)$$

At $b = 0.6$ the most likely state is “tiger left,” the greedy action is to open the right door, and it expects $A(0.6) = -34$: acting on the guess loses 34.

Now value the alternative: pay 1 to listen, then open the better door. Its value is the listening cost plus the growl-weighted average of acting on the two updated beliefs,

$$L(b) = -1 + P(\text{growl left})A(b') + P(\text{growl right})A(b''). \quad (4)$$

Substituting Equation 2 collapses this to

$$L(b) = -46 + 110(|0.5b - 0.075| + |0.5b - 0.425|), \quad (5)$$

which is **flat** at $L(b) = -7.5$ for every b between 0.159 and 0.841. At $b = 0.6$ one growl moves the agent from -34 to -7.5 , so the guess is beaten by 26.5: the growl’s information is worth 27.5, and observing it costs 1.

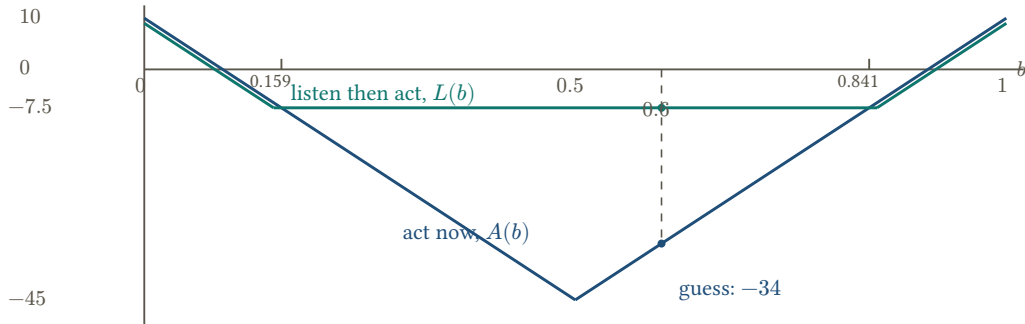


Figure 2: Two-step value over the belief interval. Navy: act now, $A(b)$, the best door opened on the current belief. Teal: listen once then act, $L(b)$, flat at -7.5 across the middle. Where teal lies above navy – about $0.159 < b < 0.841$ – the best two-step policy pays 1 to observe instead of acting on the better guess. At $b = 0.6$: guess -34 , observe -7.5 .

Two things follow. First, the greedy rule is not slightly off; it is off by the whole value of the observation: over the broad interval $0.159 < b < 0.841$ no policy that reads only the better guess can act correctly, because reading the guess throws away exactly the information that says whether a growl is worth its price. Second, the value of a belief is not the value of its most likely state but the upper envelope of the lines the alternative policies draw over the simplex – two for the doors, three for listen-then-act – so it is **piecewise-linear and convex** in b . That is the general shape of a POMDP value function, a maximum over one linear piece per **policy tree**, and what the exact algorithms manipulate [2].

4. The tree the update grows

Because the belief is a sufficient statistic, a POMDP is an MDP whose state space is the belief simplex. Its optimal value obeys a Bellman equation, but with a second expectation inside: after choosing an action the agent must average over the observations it might receive, each leading to a different belief.

$$V^*(b) = \max_{a \in A} \left\{ \int_X b(x) R(x, a) dx + \gamma \int_Z p(z | b, a) V^*(\tau(b, a, z)) dz \right\}. \quad (6)$$

Here $\tau(b, a, z)$ is the update Equation 1 written as a function of the belief, the action, and the observation. Unrolling Equation 6 does not produce a sequence of actions; it produces a **tree**. Each node is a belief. Its children are reached by an action edge, then an observation edge, and the tree alternates between the two. A policy is a whole tree: at every belief node, which action to take.

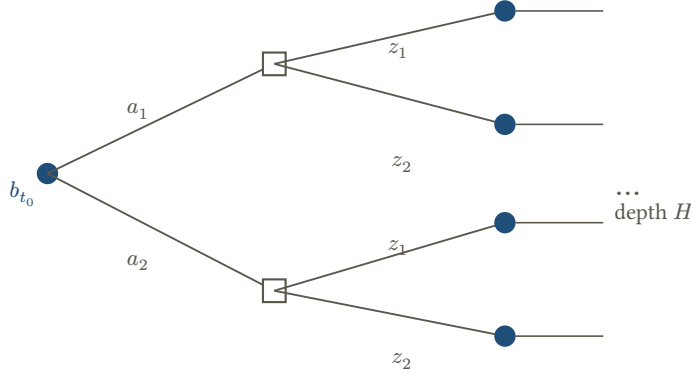


Figure 3: The belief tree. Square nodes are the agent’s action choices, round nodes are the beliefs after an observation. Each level doubles the branches, once over actions and once over observations, so a depth- H search has $|A|^H |Z|^H$ leaves – before the agents multiply anything.

The count is the problem. Forward-simulating 5 seconds at 0.2 seconds per step is 25 decision points; nine primitive controls per step give $9^{25} \approx 7 \times 10^{23}$ action sequences before a single observation is branched. With other agents in the scene, each carrying its own hidden state, the product only grows. The formalism is right and the search is infeasible, so a usable planner keeps the formalism and attacks the branching. The rest of this note is how.

5. Many agents, one factored transition

Driving adds the other cars. The full state is $x_t = \{x_t^0, x_t^1, \dots, x_t^N\}$, with $i = 0$ the ego vehicle and each x_t^i holding position, velocity, acceleration, heading, and steering. For the surrounding vehicles x_t^i also hides what cannot be observed – their **intention** and their **aggressiveness**. Only the ego can be commanded, so the action of the whole problem is just the ego action, $a_t = a_t^0$.

The joint transition $p(x_t \mid x_{t-1}, a_t)$ is not something anyone can write down directly. The step that makes it usable is to give every other agent its own action and assume the instantaneous transitions are independent [1]:

$$p(x_t \mid x_{t-1}, a_t) \approx \underbrace{p(x_t^0 \mid x_{t-1}^0, a_t^0)}_{\text{ego}} \prod_{i=1}^N \int_{A^i} \underbrace{p(x_t^i \mid x_{t-1}^i, a_t^i)}_{\text{kinematics}} \underbrace{p(a_t^i \mid x_{t-1}^i)}_{\text{driver model}} da_t^i. \quad (7)$$

Read the factors. The ego factor is ordinary motion under the chosen action. Each other agent factors into a low-level **kinematic** model of how its state evolves under an action, and a high-level **driver model** $p(a_t^i \mid x_{t-1}^i)$ that says what action it chooses. The driver model is the whole interaction physics: it takes the **joint** state as input, so what agent i does depends on where everyone – including the ego – is. The kinematic factors are independent; the driver models are not.

Assume also that observations are independent across agents, $p(z_t \mid x_t) = \prod_{i=0}^N p(z_t^i \mid x_t^i)$. Then the belief update Equation 1 factors into one term per agent:

$$\begin{aligned} b_t(x_t) &= \eta \cdot p(z_t^0 \mid x_t^0) \int_{X^0} p(x_t^0 \mid x_{t-1}^0, a_t^0) b_{t-1}^0(x_{t-1}^0) dx_{t-1}^0 \\ &\times \prod_{i=1}^N p(z_t^i \mid x_t^i) \int_{X^i A^i} p(x_t^i \mid x_{t-1}^i, a_t^i) p(a_t^i \mid x_{t-1}^i) b_{t-1}^i(x_{t-1}^i) da_t^i dx_{t-1}^i. \end{aligned} \quad (8)$$

The product looks like a set of independent updates, but it is not: the driver models $p(a_t^i \mid x_{t-1}^i)$ read the joint state, so each agent's belief is updated against what everyone else is believed to be doing. That coupling is the interaction, and it is why this factorization is not the same as predicting each car alone.

The consequence is the sentence that ties this note to the planner. The hidden states are estimated from observations during the belief update, and estimating a hidden state from observations **is** behavior prediction. Prediction is not a module that runs before planning; it is the belief-update step inside planning, and a pipeline that predicts first and plans afterward has thrown away the conditioning on the ego's own future actions [1].

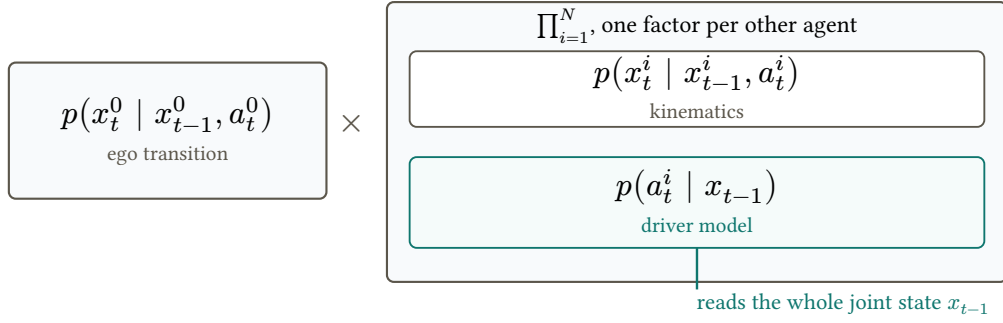


Figure 4: The factored transition of Equation 7. One joint model becomes the ego transition times, for each other agent, a kinematic model times a driver model. The driver model reads the whole joint state x_{t-1} , so it is where agents feel each other; the kinematics alone would leave them independent.

Try it: In Equation 8, which factor is the only place where one agent's state can change another agent's motion?

The driver model $p(a_t^i \mid x_{t-1})$. The kinematic factor $p(x_t^i \mid x_{t-1}^i, a_t^i)$ involves only agent i , so all coupling between agents enters through the action each one chooses, and the driver model chooses it from the joint state.

6. From a policy tree to one decision

A solved belief tree is still a tree; the motion planner downstream wants one decision. The bridge is to walk the tree down its most likely path. At each belief node the planner takes the action with the largest action value, then the observation that action is most likely to produce, and repeats:

$$a_t^* = \operatorname{argmax}_{a \in A} Q^*(b_{t-1}, a), \quad z_t^* = \operatorname{argmax}_{z \in Z} p(z \mid b_{t-1}, a_t^*). \quad (9)$$

Following those two rules from the root yields a trace of beliefs, actions, and observations, $\mathcal{S}_t = [b_t^*, a_{t+1}^*, z_{t+1}^*, b_{t+1}^*, \dots]$. Each belief in the trace is then collapsed to its most likely state,

$$x_t = \operatorname{argmax}_x b_t^*(x), \quad (10)$$

and the sequence of collapsed states for every agent becomes the decision $\mathcal{D}_t = [x_{t+1}, \dots, x_{t+H}]$ handed to the motion planner [1].

Two things are worth naming. The output is a **trace**, not a single action, because the motion planner needs to know not only where the ego should go but what everyone else is expected to do over the horizon. And following the most likely observation is a choice: it commits to the nominal branch of the tree rather than keeping the whole contingent policy. For a planner that will re-plan in 50 milliseconds, the nominal branch is the part worth passing on.

7. Making it real-time: guided observation branching

After all this machinery the only uncertainty left is the other agents' hidden states – their intentions, which the planner cannot observe. Even so, sampling intention combinations is exponential: with N agents and a few intentions each, a naive POMDP solver branches over $O(|Z|^d)$ observation sequences. EPSILON's answer is to shrink that branching with domain knowledge [1], and it starts by making the belief cheap to represent.

One number per agent, over geometric candidates. A driver's **semantic action** φ_t^i is the center-line it is pursuing, so its hidden state is a categorical belief over the candidate center-lines near it. A deliberately small network – a 1-D convolution over sampled center-line points, an LSTM over the vehicle's recent trajectory, and a softmax head – outputs that distribution. Because the classes are geometric candidates rather than fixed labels, a new road just means new candidates, with no retraining.

Then branch only where it is dangerous. This is conditional focused branching. Condition on the ego policy under evaluation, and forward-simulate once with every other agent assigned its most-likely (MAP) intention. Most cars never come near the ego's path, and for them the MAP guess is fine. The few whose simulated future **does** threaten the maneuver are the **critical** agents; those are re-simulated with their intentions swapped, one alternative at a time. A policy's score is the belief-weighted average over that handful of scenarios, so the observation space that was $O(|Z|^d)$ becomes a few evaluations.

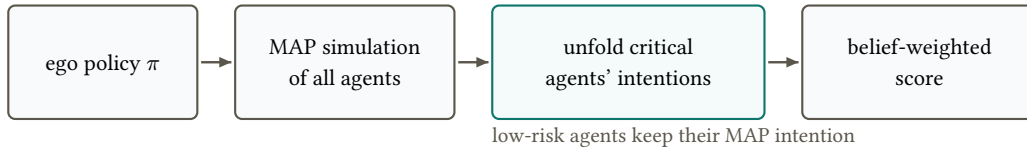


Figure 5: Conditional focused branching. The ego policy is fixed, every agent is simulated once at its most likely intention, the agents that threaten the maneuver are identified, and only those have their intentions unfolded. The policy score is the belief-weighted average over the resulting scenarios.

One assumption inside the simulation is doing a lot of work and should be said plainly. During the forward simulation, the other agents' hidden states are **not** updated: each keeps the intention it started with. The reason is the exponential blow-up above – updating them inside the simulation costs $O(|Z|^d)$ – and in practice fixing them anticipates the future well. What is **not** frozen is the interaction: every agent's action at each step depends on the simulated state of the previous step, so the rollout is still closed-loop and every car still reacts to what the others just did. What is given up is a car changing its mind mid-horizon; its average reaction is instead baked into the driver model. The same paper adds a safety mechanism precisely because that prior model can be wrong about an over-aggressive driver [1].

8. What is exact, and what was traded away

It is worth separating the two layers of this note, because they have different statuses.

The belief update Equation 1 is exact – it is Bayes’s rule, and Equation 8 is what it becomes under independent kinematics and a joint driver model. Nothing is approximated in writing it down. What is approximated is the **search**: a depth- H belief tree is replaced by a handful of closed-loop rollouts, the hidden states are frozen during a rollout, semantic actions are treated as deterministic given the hidden state, and the ego’s observations are assumed noise-free. Each choice buys real-time behavior and costs something identifiable – no mid-horizon mind-changing, a belief that is a categorical guess over candidate lanes rather than a full distribution, and a nominal trace instead of a contingent policy.

That is the honest summary of how a POMDP reaches a car. The first half of this note is the theory that makes the belief the right state; the second half is the set of approximations that makes searching it affordable. The full system around these pieces – the action-space branching, the spatio-temporal corridor, and the safety mechanism – is the subject of [its own note in this repository](#)^o.

9. Sources and further reading

The POMDP tuple, the belief update Equation 1, the factored transition Equation 7, the factored belief update Equation 8, the trace extraction Equation 9 and mode collapse Equation 10, and the guided observation branching of section 7 are from W. Ding, L. Zhang, J. Chen, and S. Shen, “EPSILON: An Efficient Planning System for Automated Vehicles in Highly Interactive Environments,” **IEEE Transactions on Robotics** 38(2):1118–1138, 2022 [1]. The tiger problem, its reward and observation models, and the observation that listening beats acting under uncertainty are from L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, “Planning and acting in partially observable stochastic domains,” **Artificial Intelligence** 101(1–2):99–134, 1998, section 5.1 [2]. Every number on this page is recomputed from the tiger model rather than quoted, and `scripts/tiger_check.py` reproduces them.

Unconsulted suggestions for the reader, not used to support any claim here: the point-based POMDP solvers (PBVI, POMCP) for the exact and approximate branches of the value function, and the earlier guided-branching paper by the same group for the origin of conditional focused branching.

Bibliography

- [1] W. Ding, L. Zhang, J. Chen, and S. Shen, “EPSILON: An Efficient Planning System for Automated Vehicles in Highly Interactive Environments,” *IEEE Transactions on Robotics*, vol. 38, no. 2, pp. 1118–1138, 2022, doi: 10.1109/TRO.2021.3104254^o.
- [2] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, “Planning and acting in partially observable stochastic domains,” *Artificial Intelligence*, vol. 101, no. 1–2, pp. 99–134, 1998, doi: 10.1016/S0004-3702(98)00023-X^o.