

**JIOSI-GILLESPIE CTMDP ROLLOUT CONSTRUCTION
& JIOSI-GILLESPIE SSA TRAJECTORY GENERATION
FOR CONTINUOUS-TIME
REINFORCEMENT LEARNING & INFERENCE**

by
Jordan Jiosi

© 2026 Jordan Jiosi

Preprint DOI: <https://doi.org/10.5281/zenodo.18156656>

Abstract

We formalize a continuous-time reinforcement learning and inverse reinforcement learning stack that uses stochastic simulation algorithm dynamics to generate event-timed trajectories. The Jiosi-Gillespie CTMDP Rollout Construction treats Gillespie sampling as an environment stepper mapping (s, a) to (s^+, r, τ) with an explicit event label, while Jiosi-Gillespie SSA Trajectory Generation produces full jump sequences suitable for likelihood-based inference. The framework supports event-driven or piecewise-constant control, preserves standard policy evaluation and improvement logic, and admits likelihoods over timestamped demonstrations. A scheduling toy problem is reformulated from discrete to continuous time, and a concrete SSA step is shown numerically. Discounting can be applied through exponential factors or reward folding, making the method compatible with standard RL objectives while retaining exact continuous-time semantics.

1. Introduction

Discrete-state systems often evolve with irregular event timing, yet most reinforcement learning (RL) rollouts assume a fixed time step. That assumption blurs the impact of dwell times on rewards and on inverse reinforcement learning (IRL) likelihoods. We present the Jiosi-Gillespie CTMDP Rollout Construction and Jiosi-Gillespie SSA Trajectory Generation, two contracts that embed Gillespie’s stochastic simulation algorithm (SSA) into a continuous-time Markov decision process (CTMDP) compatible with modern RL and IRL. The resulting trajectories retain event times, enabling value estimation, discounting, and likelihoods without time discretization. Our contributions are: (i) a precise CTMDP stepper built on SSA rates; (ii) RL and IRL interfaces that operate on jump trajectories; and (iii) a continuous-time reformulation of a scheduling toy example inspired by [1].

2. Continuous-Time Decision Processes and Trajectories

Consider a CTMDP with state space \mathcal{S} , action set \mathcal{A} , event types Σ , and policy $\pi_\theta(a|s)$. For each event type $\sigma \in \Sigma$ and state-action pair (s, a) , the instantaneous rate is $\lambda_\sigma(s, a)$. The total rate is

$$\Lambda(s, a) \doteq \sum_{\sigma \in \Sigma} \lambda_\sigma(s, a), \quad (1)$$

and the event probability mass function is $p(\sigma | s, a) \doteq \lambda_\sigma(s, a)/\Lambda(s, a)$. Holding times satisfy $\tau_t \sim \text{Exp}(\Lambda(s_t, a_t))$. A trajectory collects jump tuples

$$\tau = \{(s_t, a_t, \tau_t, \sigma_t, s_{t+1})\}_{t=0}^{T-1}, \quad (2)$$

where s_{t+1} is drawn from the transition kernel conditioned on (s_t, a_t, σ_t) . A discounted return can be written as

$$F(\tau) = \sum_{t=0}^{T-1} \exp\left(-\beta \sum_{j=0}^t \tau_j\right) r(s_t, a_t, \sigma_t), \quad (3)$$

with rate-based discount $\beta \geq 0$. The objective is $\mathbb{E}_{\tau|\pi_\theta}[F(\tau)]$; value functions similarly use state-action marginals such as $\mathbb{E}_{s,a|\pi_\theta}[f(s, a)]$ without time discretization. This notation aligns with classical CTMDP treatments such as [2].

3. Jiosi-Gillespie CTMDP Rollout Construction

Each SSA step instantiates an environment transition. Given (s_t, a_t) :

$$\Lambda(s_t, a_t) = \sum_{\sigma \in \Sigma} \lambda_\sigma(s_t, a_t), \quad (4)$$

$$\tau_t \sim \text{Exp}(\Lambda(s_t, a_t)), \quad (5)$$

$$\sigma_t \sim p(\sigma | s_t, a_t), \quad (6)$$

$$s_{t+1} \sim P(\cdot | s_t, a_t, \sigma_t). \quad (7)$$

The stepper contract is

$$(s_{t+1}, r_t, \tau_t, \sigma_t) = \text{Step}(s_t, a_t), \quad (8)$$

where $r_t = r(s_t, a_t, \sigma_t)$ and Step is simulated via SSA sampling. This construction retains (τ_t, σ_t) alongside (s_t, a_t) , enabling downstream RL to remain agnostic to the timing model while still obtaining time-aware returns. Piecewise-constant control holds a_t fixed over the sampled holding time; event-driven control updates a_t immediately after each jump.

Algorithm 1 Jiosi-Gillespie SSA Trajectory Generation

- 1: Input: policy π_θ , horizon T , discount β , initial state s_0
 - 2: Initialize $G \leftarrow 0$, cumulative time $c \leftarrow 0$
 - 3: **for** $t = 0$ to $T - 1$ **do**
 - 4: Sample $a_t \sim \pi_\theta(a|s_t)$
 - 5: Compute $\Lambda(s_t, a_t)$ and $p(\sigma | s_t, a_t)$
 - 6: Draw $u_1, u_2 \sim \text{Unif}(0, 1)$
 - 7: Set $\tau_t = -\ln(u_1)/\Lambda(s_t, a_t)$ and update $c \leftarrow c + \tau_t$
 - 8: Sample σ_t using u_2 and the cumulative mass of $p(\sigma | s_t, a_t)$
 - 9: Evaluate $r_t = r(s_t, a_t, \sigma_t)$
 - 10: Accumulate $G \leftarrow G + \exp(-\beta c) r_t$
 - 11: Sample $s_{t+1} \sim P(\cdot | s_t, a_t, \sigma_t)$
 - 12: **end for**
 - 13: Return trajectory $\{(s_t, a_t, \tau_t, \sigma_t, s_{t+1})\}_{t=0}^{T-1}$ and return G
-

4. Jiosi-Gillespie SSA Trajectory Generation

Gillespie sampling [3, 4] draws holding times and event types from two uniform random variables $u_1, u_2 \sim \text{Unif}(0, 1)$:

$$\tau_t = -\frac{1}{\Lambda(s_t, a_t)} \ln(u_1), \quad (9)$$

$$\sigma_t = \min \left\{ \sigma \in \Sigma : \sum_{\sigma' \leq \sigma} p(\sigma' | s_t, a_t) \geq u_2 \right\}. \quad (10)$$

Algorithm Algorithm 1 generates a rollout episode. Discounting can be applied multiplicatively through $\exp(-\beta \tau_t)$ or folded into the reward as $\tilde{r}_t = \exp(-\beta \tau_t) r(s_t, a_t, \sigma_t)$; both choices keep policy evaluation unchanged while preserving continuous-time semantics.

5. Reinforcement Learning Interface

The CTMDP value of policy π_θ is $J(\theta) = \mathbb{E}_{\tau|\pi_\theta}[F(\tau)]$. Event-driven control selects a_t after each jump; piecewise-constant control commits to a_t over $(t, t + \tau_t)$. Both styles share the same return definition, so policy evaluation and improvement operators remain

structurally identical to discrete-time RL [5]. For policy gradients,

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\tau|\pi_{\theta}} \left[\sum_{t=0}^{T-1} \nabla_{\theta} \log \pi_{\theta}(a_t|s_t) \hat{A}_t \right], \quad (11)$$

where \hat{A}_t may incorporate holding-time-aware baselines such as $\hat{A}_t = \exp(-\beta \tau_t) \hat{Q}(s_t, a_t) - b(s_t)$. Standard critics and actor updates can therefore consume SSA rollouts without altering optimizer structure.

6. Inverse Reinforcement Learning Interface

Expert demonstrations are timestamped jump trajectories $\mathcal{D} = \{\tau^{(n)}\}_{n=1}^N$ with $\tau^{(n)} = \{(\tau_t^{(n)}, \sigma_t^{(n)}, s_t^{(n)}, a_t^{(n)})\}$. The joint density for one step under parameters θ is $\lambda_{\sigma_t}(s_t, a_t) \exp(-\Lambda(s_t, a_t) \tau_t) \pi_{\theta}(a_t|s_t)$. The trajectory log-likelihood is

$$\begin{aligned} \log p(\tau|\theta) = \sum_{t=0}^{T-1} & \left[\log \lambda_{\sigma_t}(s_t, a_t) - \Lambda(s_t, a_t) \tau_t \right. \\ & \left. + \log \pi_{\theta}(a_t|s_t) \right], \end{aligned} \quad (12)$$

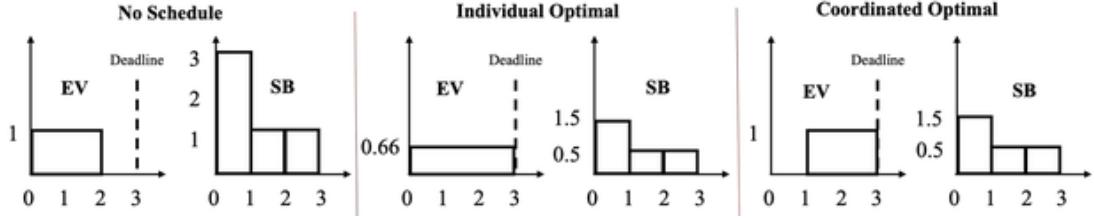
and IRL adapts either reward parameters (affecting λ_{σ} or r) or policy parameters so that $\mathbb{E}_{\tau|\pi_{\theta}}[F(\tau)]$ aligns with expert likelihood [6]. Reward recovery constrains λ_{σ} through optimality conditions; policy recovery directly maximizes the likelihood above, keeping the SSA dynamics fixed.

7. Continuous-Time Reformulation of a Scheduling Toy Example

The discrete scheduling chart in Figure Figure 1 allocates two jobs: an electric vehicle (EV) charge and a standby (SB) task, each with unit work across a three-slot horizon and a deadline at $t = 3$. In continuous time, the state records remaining work ($w^{\text{EV}}, w^{\text{SB}}$), the elapsed time c , and the deadline indicator. Event types are $\Sigma = \{\text{serve-EV}, \text{serve-SB}\}$ with rates $\lambda_{\text{EV}}(s, a)$ and $\lambda_{\text{SB}}(s, a)$ chosen by action $a \in \{\text{EV-first}, \text{SB-first}, \text{split}\}$. For action

Figure 1

Discrete scheduling illustration adapted from [1]. SSA rollouts reuse the same prioritization logic but sample holding times and event order continuously.



EV-first, set $\lambda_{\text{EV}} = 1.5$, $\lambda_{\text{SB}} = 0.5$, giving $\Lambda = 2.0$ and $p(\text{EV} \mid s, a) = 0.75$. The holding time is $\tau_t \sim \text{Exp}(2.0)$.

A worked SSA step: suppose $w^{\text{EV}} = 1$, $w^{\text{SB}} = 1$, deadline at $c = 0$, and $a_t = \text{EV-first}$.

Draw $u_1 = 0.25$, $u_2 = 0.60$. Then $\tau_t = -\ln(0.25)/2.0 \approx 0.69$, $\sigma_t = \text{serve-EV}$ because $u_2 < 0.75$, and the state updates to $w^{\text{EV}} \leftarrow 0$. The reward can encode on-time completion, e.g., $r_t = 1$ if $c + \tau_t < 3$ and w^{EV} hits zero, else -1 . Subsequent steps continue with updated rates; the SSA keeps exact timing while preserving the discrete action logic from the original schedule.

8. Discussion and Limitations

SSA-based rollouts assume memoryless holding times and well-specified rate functions; systems with strong duration dependence may violate these assumptions. Identifiability in IRL is challenging when both rates and rewards are unknown: multiple (λ_σ, r) pairs can induce the same likelihood over $\{(\tau_k, \sigma_k, s_k)\}$. Regularization, structural constraints on λ_σ , or anchoring to known physics can mitigate ambiguity. Finally, long holding times can create high-variance returns; control variates based on $\mathbb{E}_{s,a|\pi_\theta}[\Lambda(s,a)]$ help stabilize estimates.

9. Conclusion

The Jiosi-Gillespie CTMDP Rollout Construction and Jiosi-Gillespie SSA Trajectory Generation supply event-timed trajectories that keep RL and IRL objectives intact while avoiding time discretization. By treating SSA as the environment stepper, policies remain compatible with event-driven and piecewise-constant control, and likelihoods over timestamped demonstrations become straightforward. The continuous-time reformulation of the scheduling toy example illustrates how discrete action logic transfers directly to SSA rollouts, enabling precise reward attribution and inference.

References

- [1] J. Jiosi, “Empirical analysis on multi-agent reinforcement learning frameworks,” M.S. thesis, Rowan University, 2025.
- [2] X. Guo and O. Hernández-Lerma, *Continuous-Time Markov Decision Processes: Theory and Applications* (Stochastic Modelling and Applied Probability). Springer, 2009. DOI: [10.1007/978-3-540-76713-0](https://doi.org/10.1007/978-3-540-76713-0).
- [3] D. T. Gillespie, “A general method for numerically simulating the stochastic time evolution of coupled chemical reactions,” *Journal of Computational Physics*, vol. 22, no. 4, pp. 403–434, 1976. DOI: [10.1016/0021-9991\(76\)90041-3](https://doi.org/10.1016/0021-9991(76)90041-3).
- [4] D. T. Gillespie, “Exact stochastic simulation of coupled chemical reactions,” *The Journal of Physical Chemistry*, vol. 81, no. 25, pp. 2340–2361, 1977. DOI: [10.1021/j100540a008](https://doi.org/10.1021/j100540a008).
- [5] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. MIT Press, 2018.
- [6] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey, “Maximum entropy inverse reinforcement learning,” in *AAAI Conference on Artificial Intelligence*, 2008.