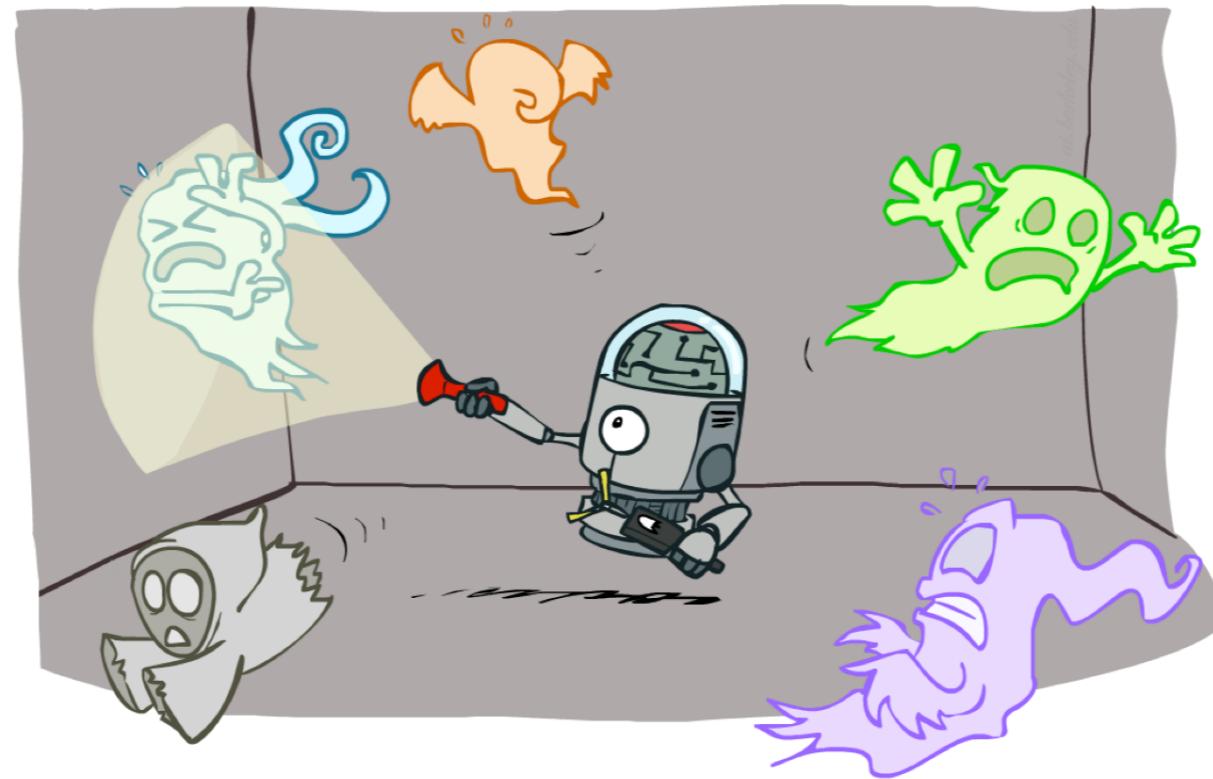


# Ve492: Introduction to Artificial Intelligence

## Hidden Markov Models II



Paul Weng

UM-SJTU Joint Institute

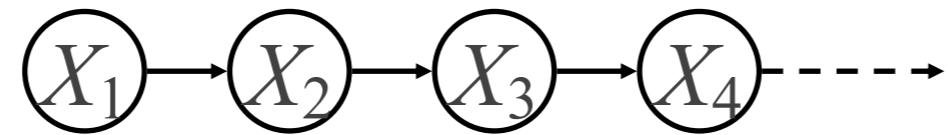
Slides adapted from <http://ai.berkeley.edu>, AIMA, UM, CMU

# Today

---

- ❖ Quick Review of (H)MM
- ❖ Particle Filtering
- ❖ Viterbi Algorithm for Most Likely Explanation
- ❖ Dynamic Bayesian Network

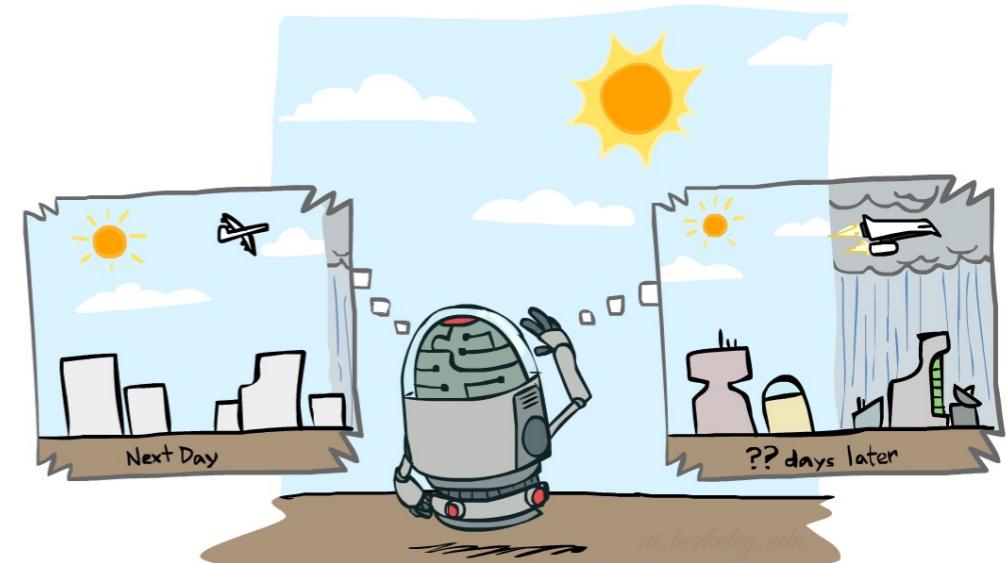
# Markov Chain



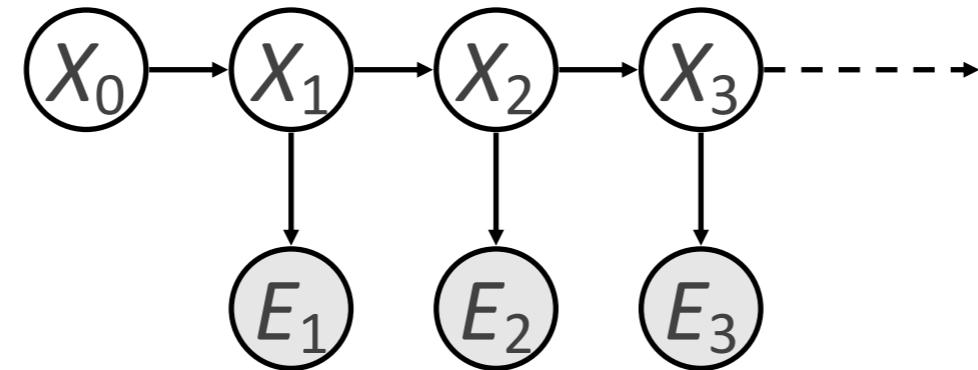
- ❖ You know transition probabilities,  $P(X_t \mid X_{t-1})$ , and  $P(X_4)$ .
- ❖ Write an equation to compute  $P(X_5)$ .

$$\begin{aligned} P(X_5) &= \sum_{x_4} P(x_4, X_5) \\ &= \sum_{x_4} P(X_5 \mid x_4)P(x_4) \end{aligned}$$

- ❖ More generally:  $P_{t+1} = T^\top P_t$  where  $P_t = P(X_t)$
- ❖ Stationary distribution:  $P_\infty = T^\top P_\infty$



# HMM, Filtering, Forward Algorithm



Filtering: What is the current state, given all evidence?

$$P(X_{t+1} | e_{1:t+1}) = \alpha P(e_{t+1} | X_{t+1}) \sum_{x_t} P(X_{t+1} | x_t) P(x_t | e_{1:t})$$

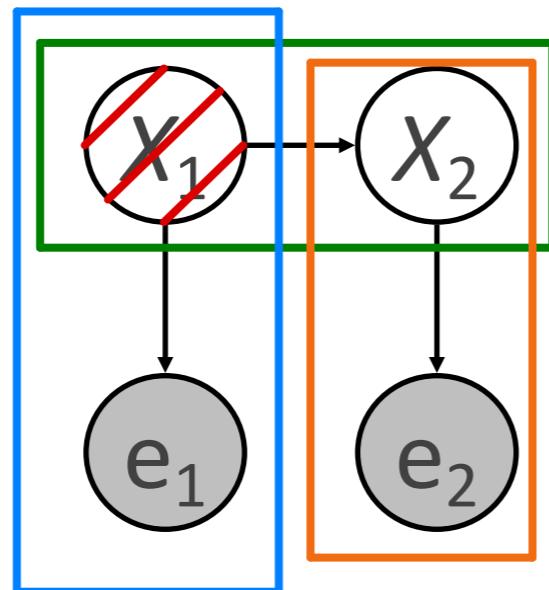


$$\mathbf{f}_{1:t+1} = \text{FORWARD}(\mathbf{f}_{1:t}, e_{t+1})$$

# Forward Algorithm

Query: What is the current state, given all of the current and past evidence?

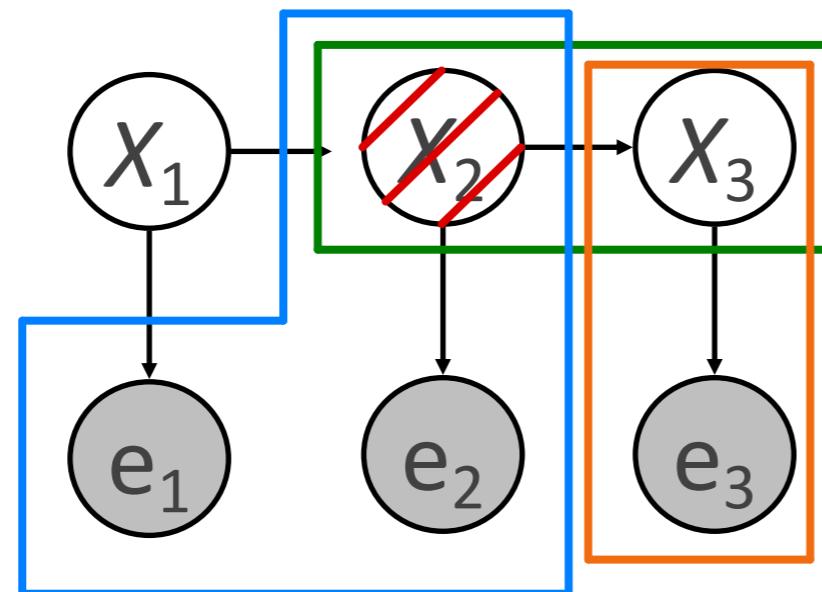
Marching **forward** through the HMM network



# Forward Algorithm

Query: What is the current state, given all of the current and past evidence?

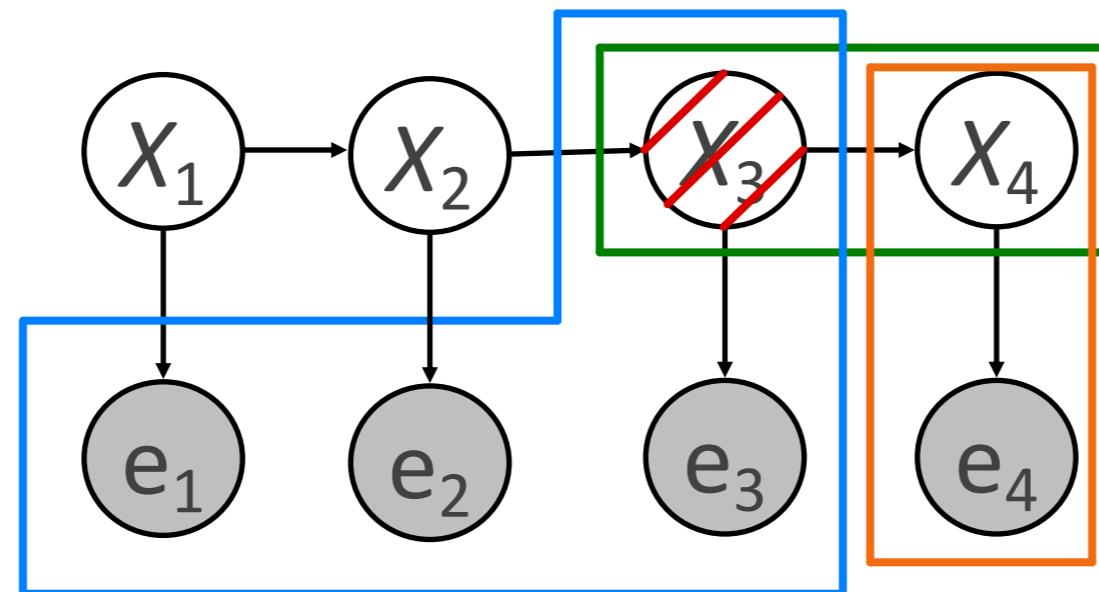
Marching **forward** through the HMM network



# Forward Algorithm

Query: What is the current state, given all of the current and past evidence?

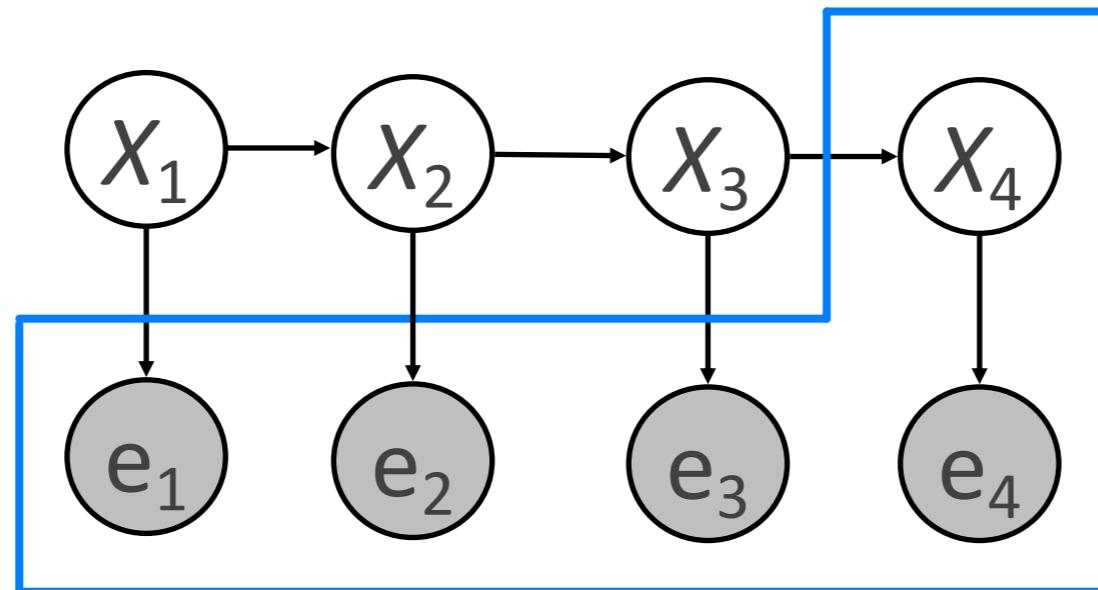
Marching **forward** through the HMM network



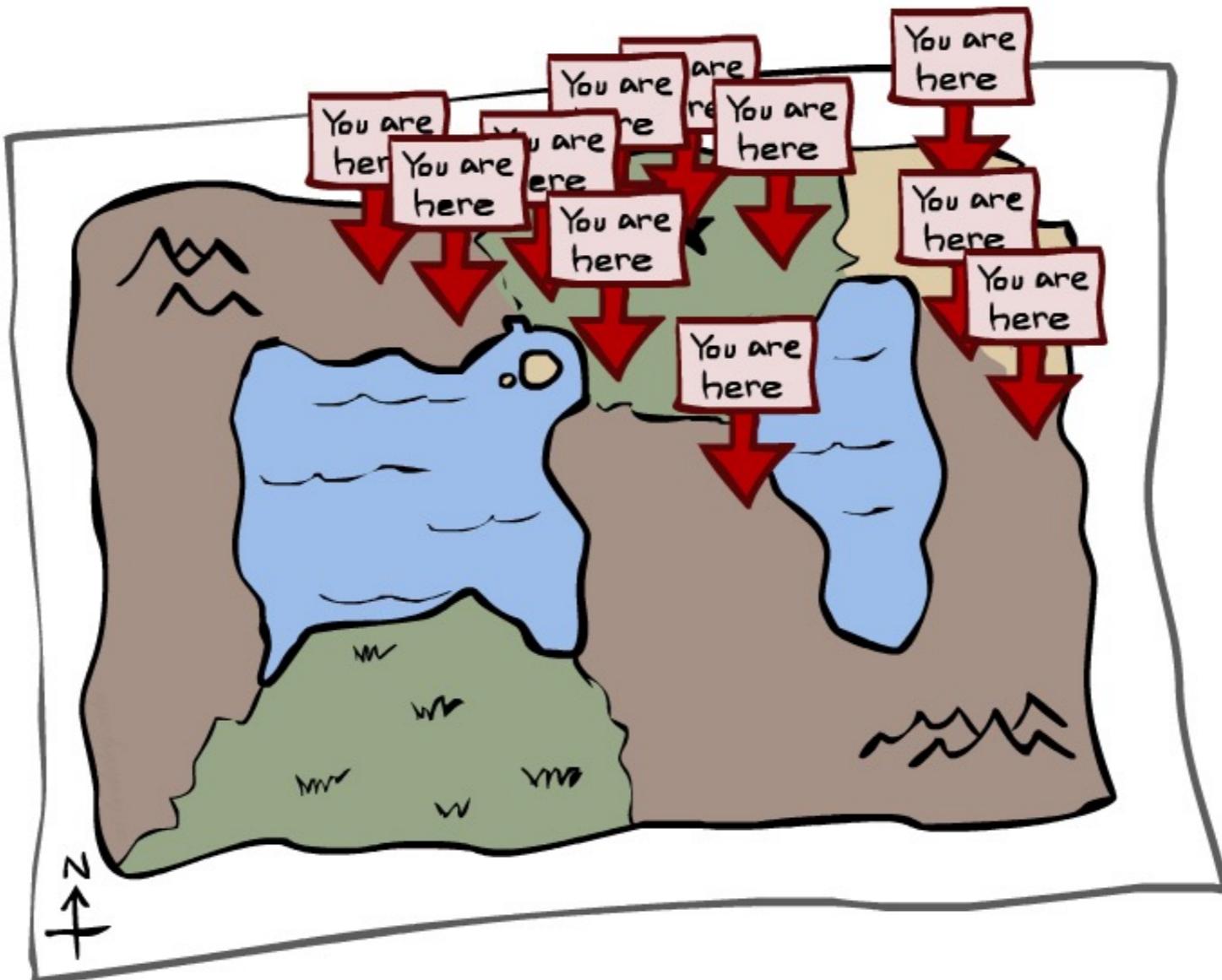
# Forward Algorithm

Query: What is the current state, given all of the current and past evidence?

Marching **forward** through the HMM network



# Particle Filtering



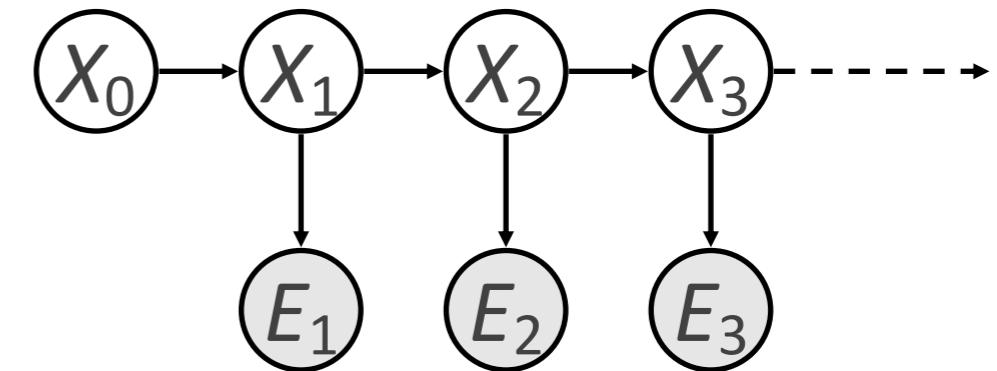
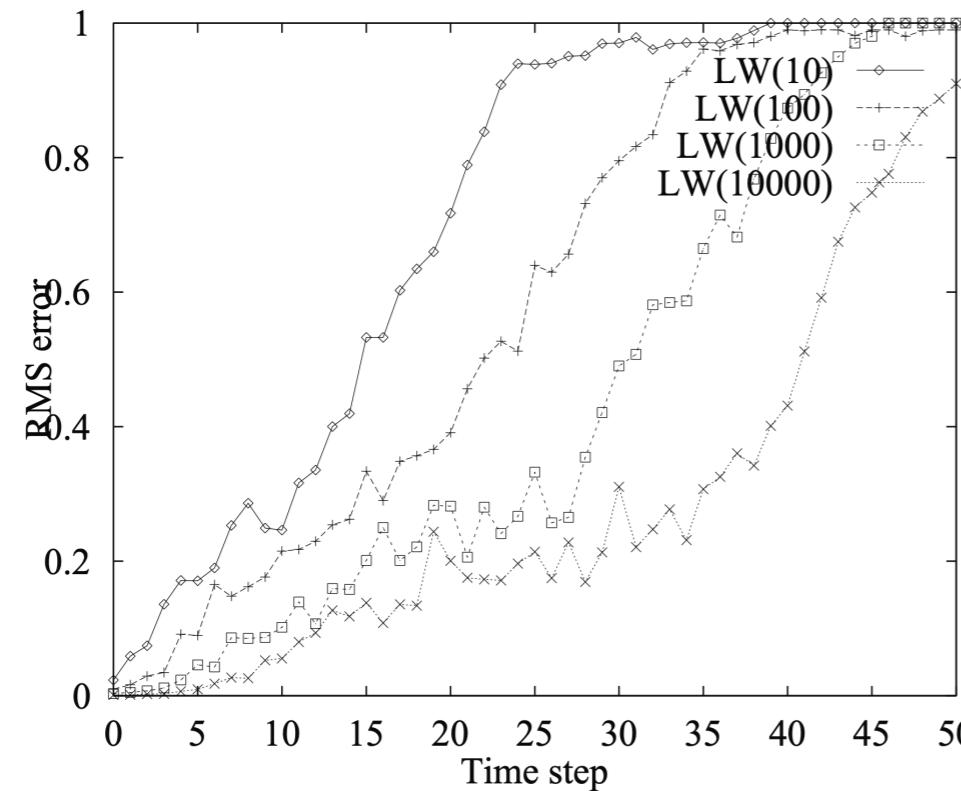
# Quiz: Algorithms for Filtering

---

- ❖ Which algorithms can be applied to the filtering task?
  - ❖ Variable elimination
  - ❖ Prior sampling
  - ❖ Rejection sampling
  - ❖ Likelihood weighting
  - ❖ Gibbs sampling

# Limitations of Current Algorithms

- ❖ Exact inference infeasible if state space is large or continuous
  - ❖ e.g., when  $|X|$  is more than  $10^6$  or so (tracking 3 ghosts in a  $10 \times 20$  world)
  - ❖ e.g., robot localization



- ❖ Previous sampling methods do not exploit the temporal structure of an HMM

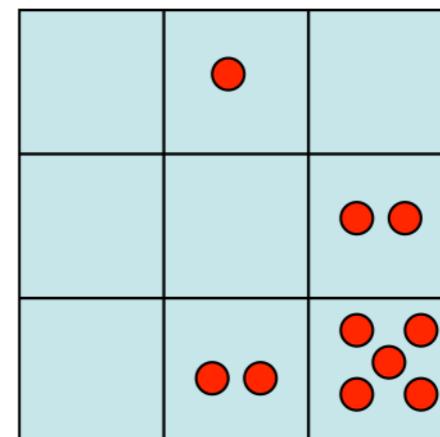
# Particle Filtering

- ❖ PF = approximate inference for filtering task

- ❖ Principle of Particle Filtering

- ❖ Track samples of  $X_t$ , not all values
- ❖ Samples are called particles
- ❖ Time per step is linear in the number of samples
- ❖ But: number needed may be large
- ❖ In memory: list of particles, not states

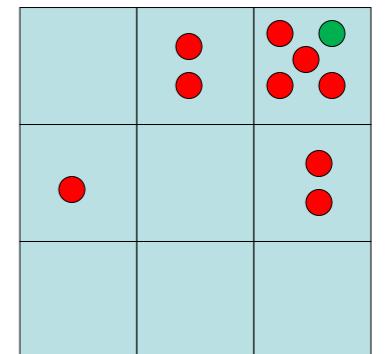
0.0	0.1	0.0
0.0	0.0	0.2
0.0	0.2	0.5



# Representation: Particles

- ❖ Our representation of  $P(X)$  is now a list of  $N$  particles (samples)

- ❖ Generally,  $N \ll |X|$
- ❖ Storing map from  $X$  to counts would defeat the point



- ❖  $P(x)$  approximated by number of particles with value  $x$

- ❖ So, many  $x$  may have  $P(x) = 0$ !
- ❖ More particles, more accuracy

Particles:

(3,3)

(2,3)

(3,3)

(3,2)

(3,3)

(3,2)

(1,2)

(3,3)

(3,3)

(2,3)

- ❖ For now, all particles have a weight of 1

- ❖ In PF, a set of samples approximates  $f_{1:t}(X_t) = P(X_t | e_{1:t})$

# Particle Filtering: Elapse Time

- ❖ This first step captures the passage of time
- ❖ Each particle is moved by sampling its next position from the transition model

$$x' = \text{sample}(P(X'|x))$$

- ❖ This is like prior sampling – samples' frequencies reflect the transition probabilities
- ❖ Here, most samples move clockwise, but some move in another direction or stay in place

- ❖ New particles approximates:

$$\sum_{x_t} P(X_{t+1}|x_t) f_{1:t}(x_t)$$

- ❖ Consistency: If enough samples, close to exact values before and after

Particles:

(3,3)

(2,3)

(3,3)

(3,2)

(3,3)

(3,2)

(1,2)

(3,3)

(3,3)

(2,3)

Particles

:

(3,2)

(2,3)

(3,2)

(3,1)

(3,3)

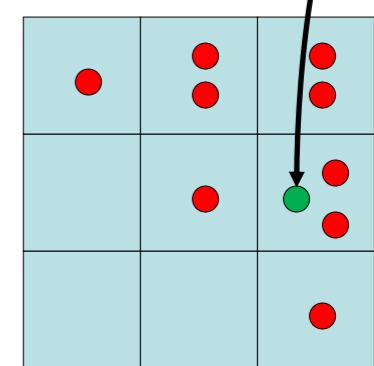
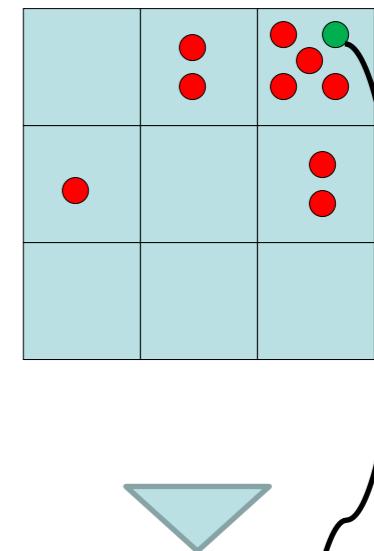
(3,2)

(1,3)

(2,3)

(3,2)

(2,2)

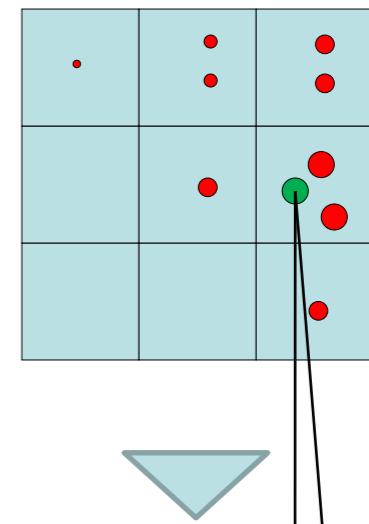


# Particle Filtering: Resample

- ❖ Particles receive weights corresponding to  $P(e' | X')$
- ❖ Rather than tracking weighted samples, we resample
- ❖ N times, we choose from our weighted sample distribution (i.e. draw with replacement)
- ❖ This is equivalent to renormalizing the distribution
- ❖ Now the update is complete for this time step, continue with the next one

Particles:

(3,2) w=.9  
(2,3) w=.2  
(3,2) w=.9  
(3,1) w=.4  
(3,3) w=.4  
(3,2) w=.9  
(1,3) w=.1  
(2,3) w=.2  
(3,2) w=.9  
(2,2) w=.4

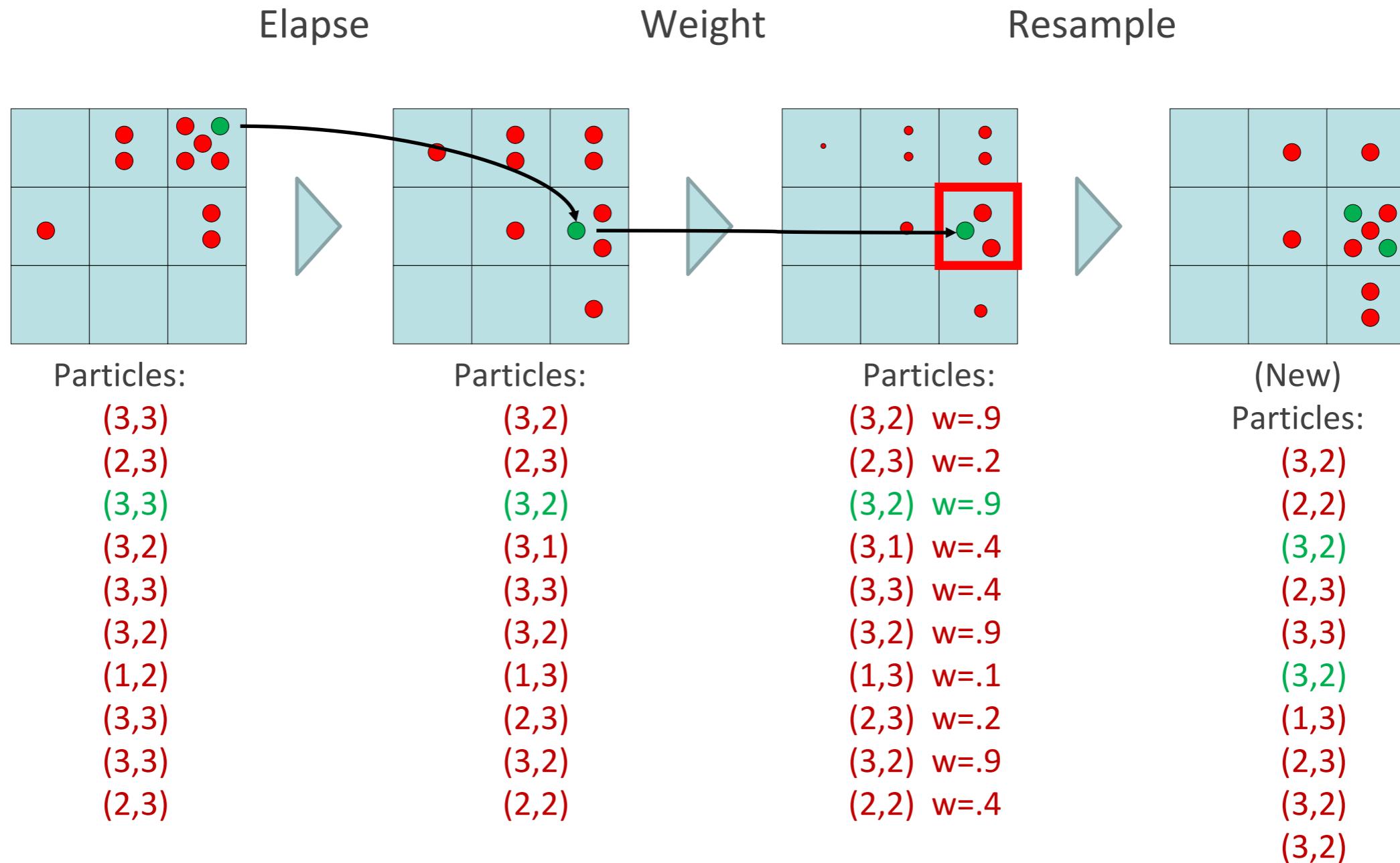


(New)  
Particles:

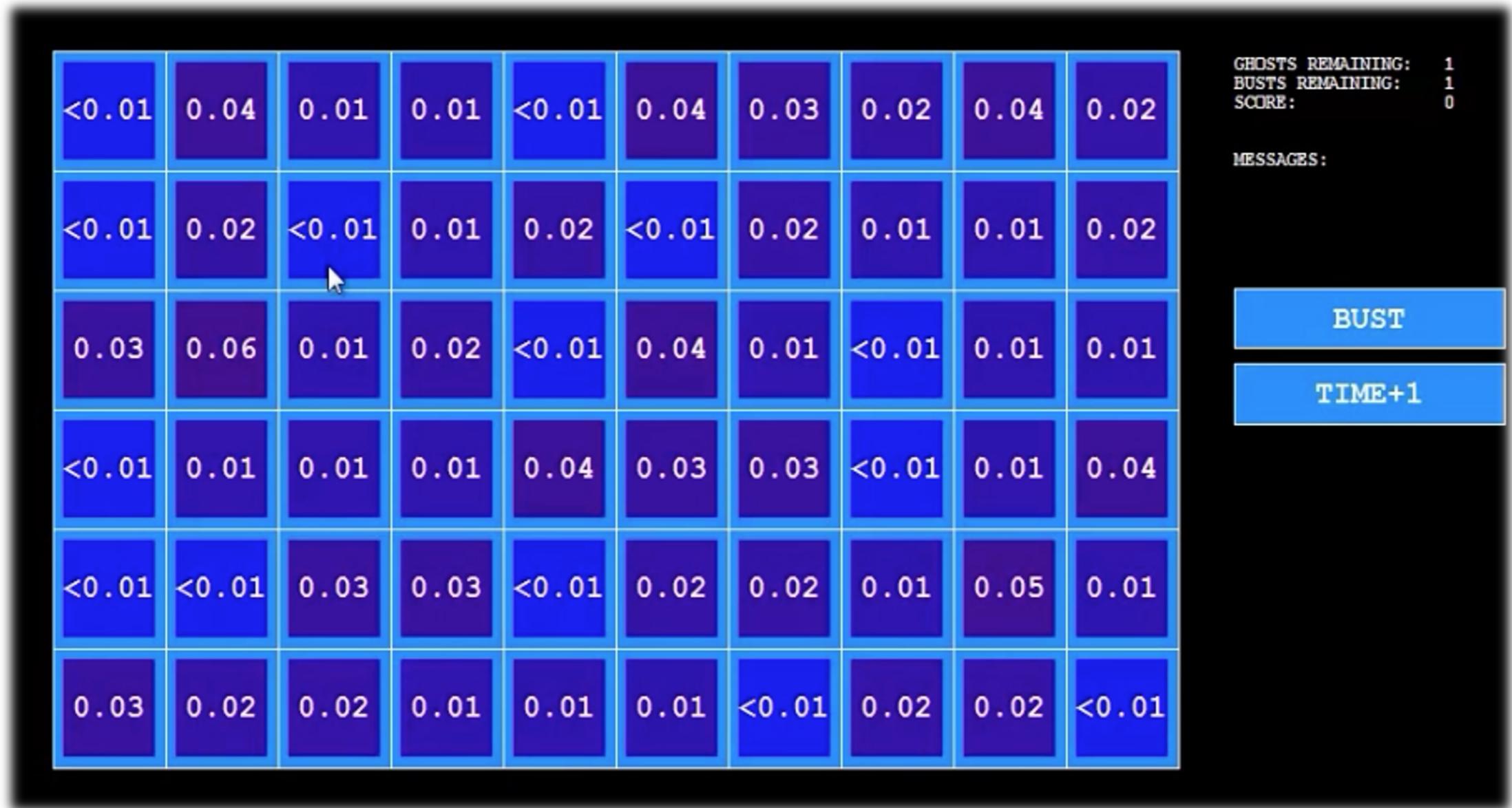
(3,2)  
(2,2)  
(3,2)  
(2,3)  
(3,3)  
(3,2)  
(1,3)  
(2,3)  
(3,2)  
(3,2)

# Summary: Particle Filtering

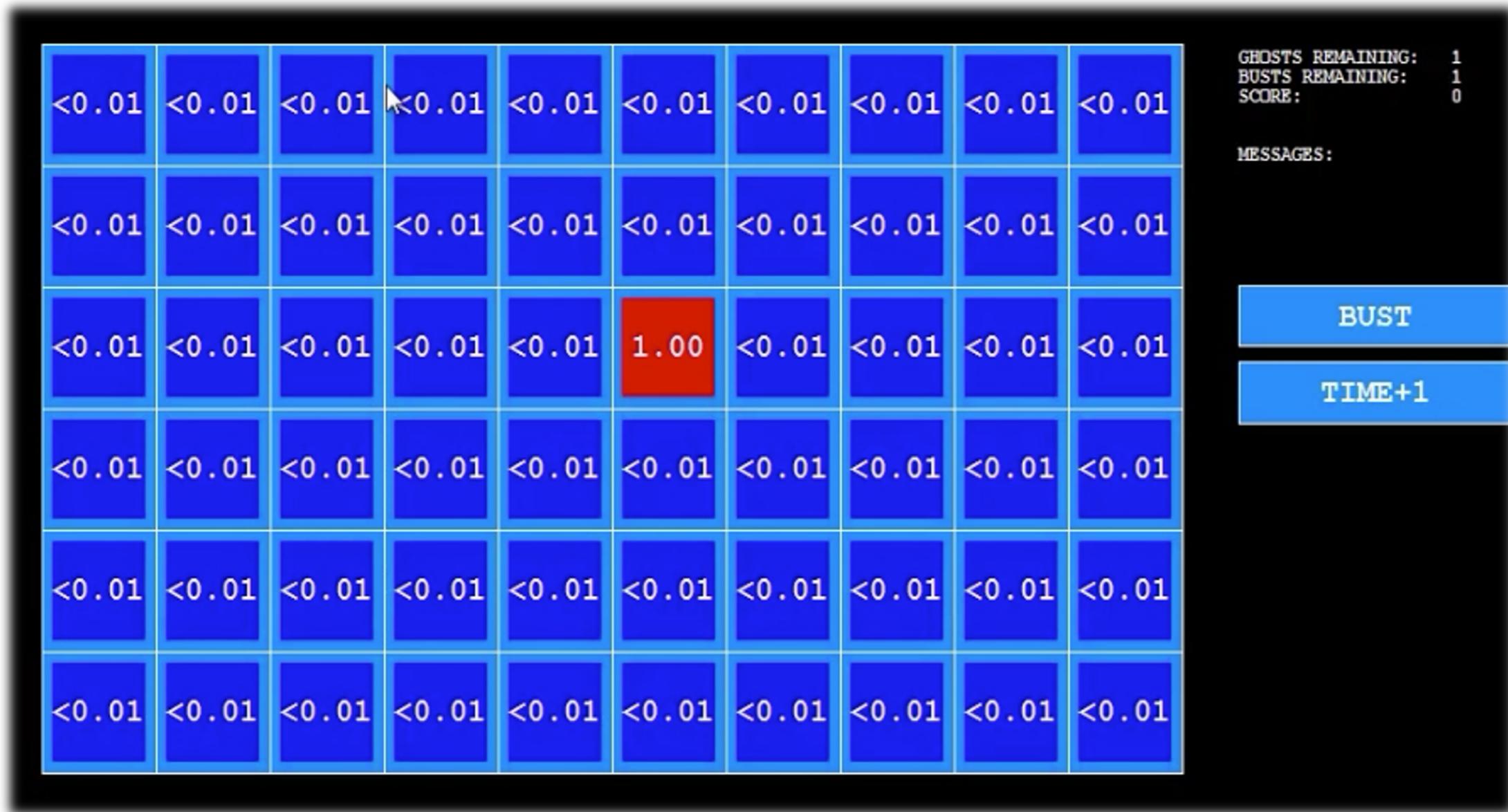
- ❖ Particles: track samples of states rather than an explicit distribution



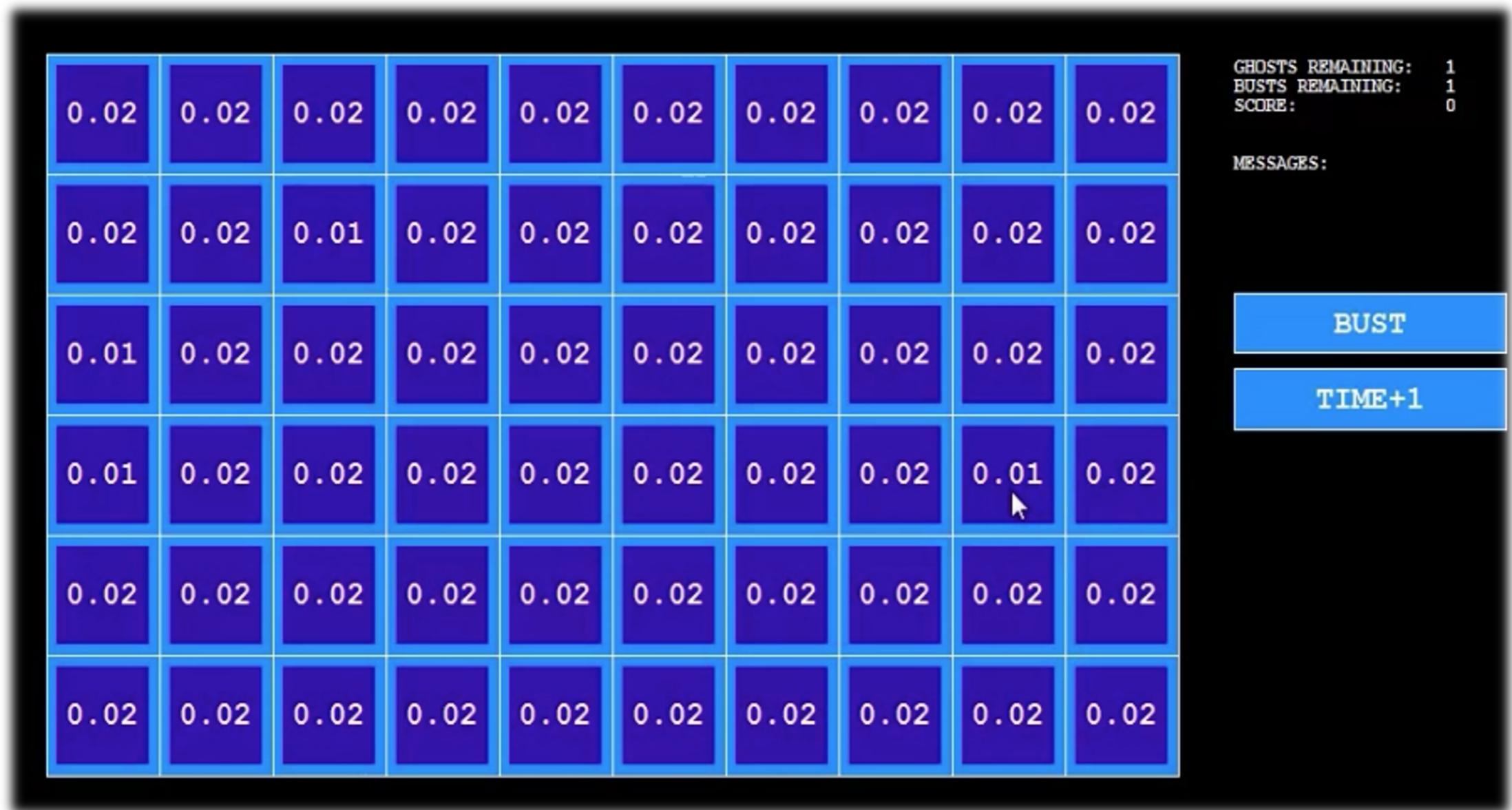
# Video of Demo – Moderate Number of Particles



# Video of Demo – One Particle



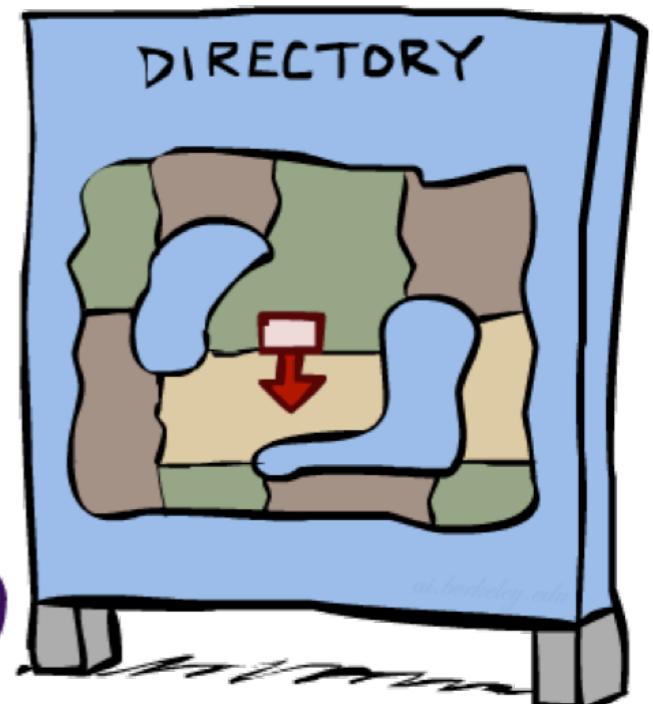
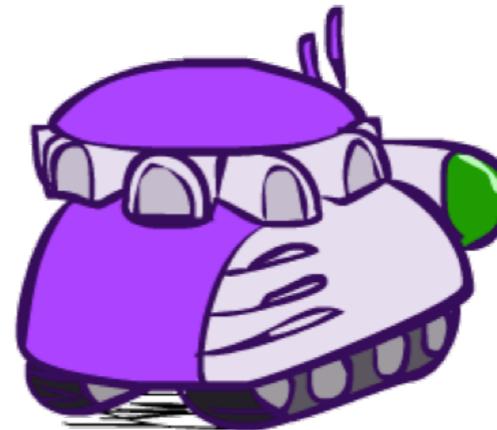
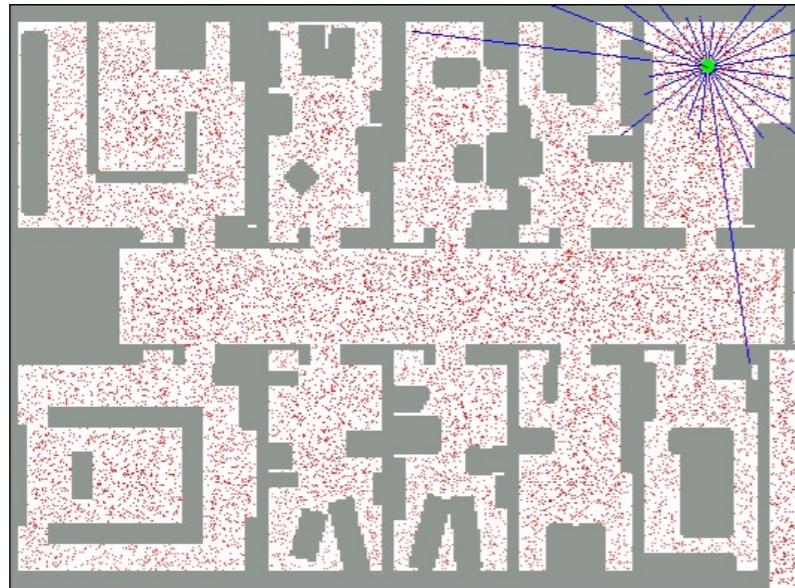
# Video of Demo – Huge Number of Particles



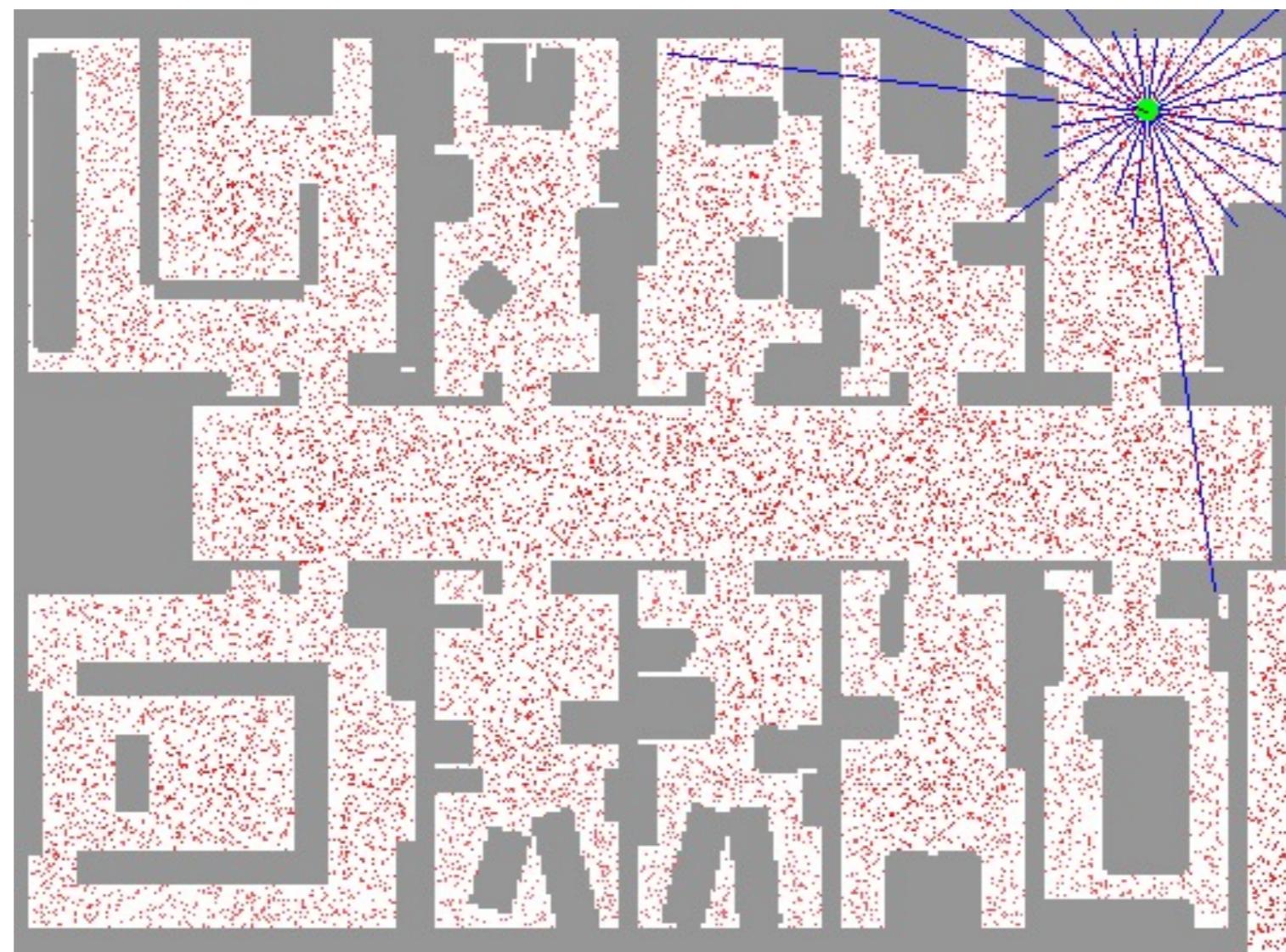
# Robot Localization

In robot localization:

- ❖ We know the map, but not the robot's position
- ❖ Observations may be vectors of range finder readings
- ❖ State space and readings are typically continuous
- ❖ Particle filtering is a main technique



# Particle Filter Localization (Laser)



[Dieter Fox, et al.]

# Most Likely Explanation

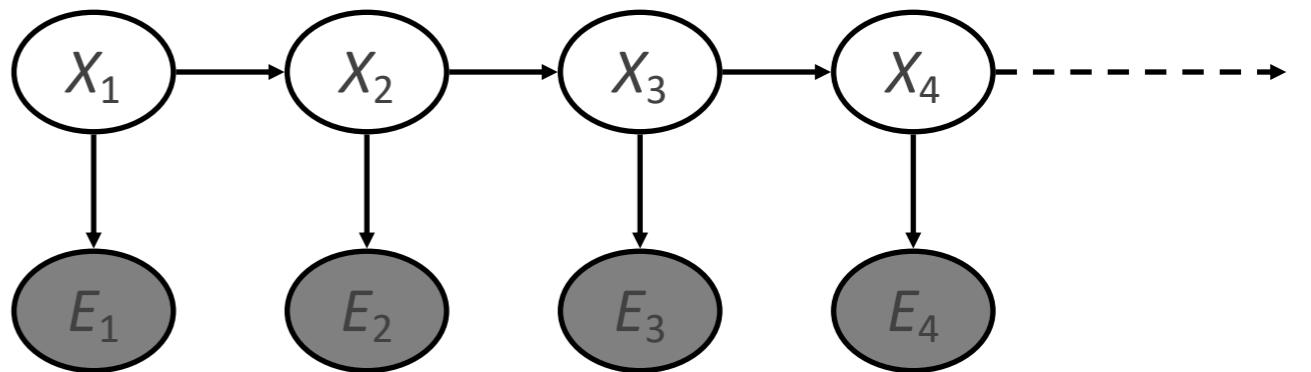


# Inference tasks

- ❖ **Filtering:**  $P(X_t | e_{1:t})$ 
  - ❖ **belief state**—input to the decision process of a rational agent
- ❖ **Prediction:**  $P(X_{t+k} | e_{1:t})$  for  $k > 0$ 
  - ❖ evaluation of possible action sequences; like filtering without the evidence
- ❖ **Smoothing:**  $P(X_k | e_{1:t})$  for  $0 \leq k < t$ 
  - ❖ better estimate of past states, essential for learning
- ❖ **Most likely explanation:**  $\arg \max_{x_{1:t}} P(x_{1:t} | e_{1:t})$ 
  - ❖ speech recognition, decoding with a noisy channel

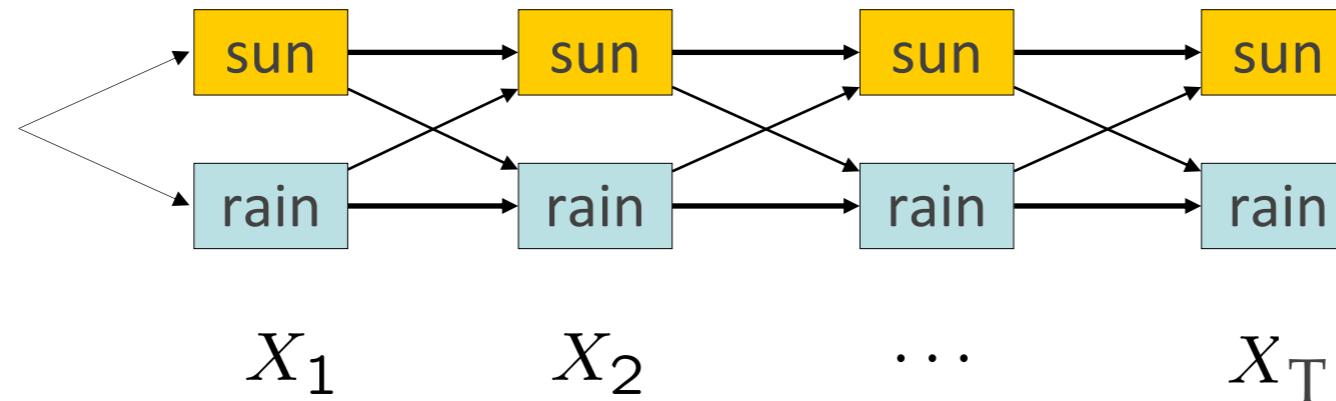
# HMMs: MLE Queries

- ❖ HMMs defined by
  - ❖ States  $X$
  - ❖ Observations  $E$
  - ❖ Initial distribution:  $P(X_1)$
  - ❖ Transitions:  $P(X_t|X_{t-1})$
  - ❖ Emissions:  $P(E|X)$
- ❖ New query: most likely explanation:  $\text{argmax}_{x_{1:t}} P(x_{1:t}|e_{1:t})$
- ❖ New method: the Viterbi algorithm



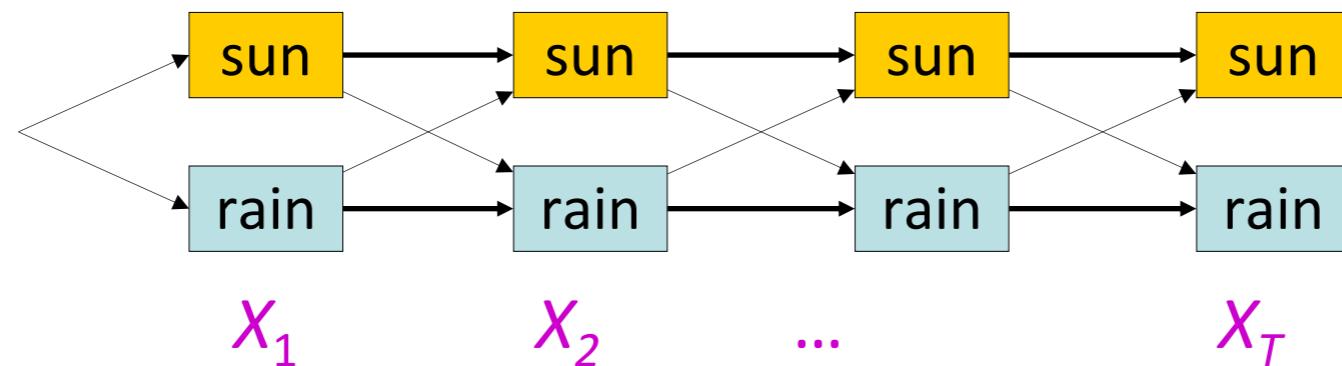
# State Trellis

- ❖ State trellis: graph of states and transitions over time



- ❖ Each arc represents some transition  $x_{t-1} \rightarrow x_t$
- ❖ Each arc has weight  $P(x_t|x_{t-1})P(e_t|x_t)$
- ❖ Each path is a sequence of states
- ❖ Product of weights on a path = sequence's probability along with the evidence
- ❖ Forward algorithm computes sums of paths, Viterbi computes best paths

# Forward / Viterbi algorithms



## Forward Algorithm (sum)

For each state at time  $t$ , keep track of the ***total probability of all paths*** to it

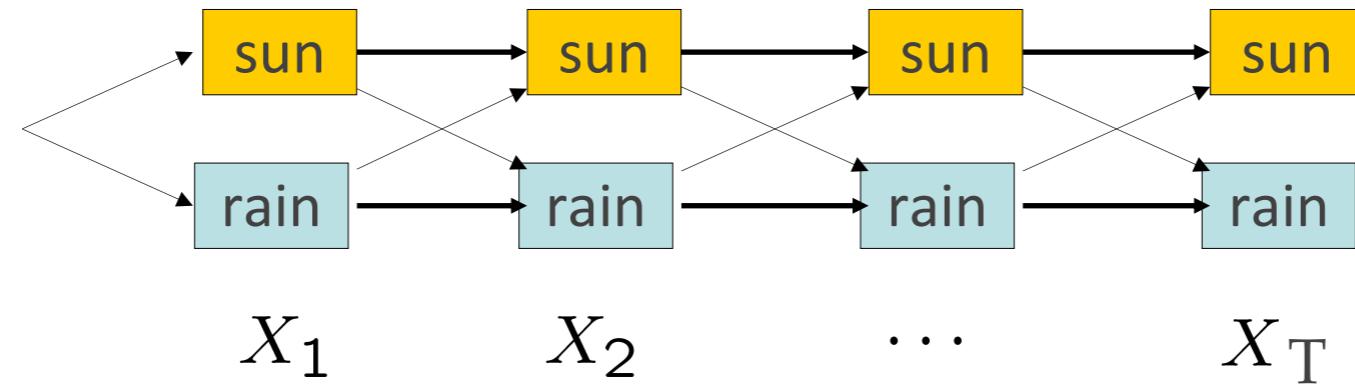
$$\begin{aligned}\mathbf{f}_{1:t+1} &= \text{FORWARD}(\mathbf{f}_{1:t}, e_{t+1}) \\ &= \alpha P(e_{t+1}|X_{t+1}) \sum_{x_t} P(X_{t+1}|x_t) \mathbf{f}_{1:t}\end{aligned}$$

## Viterbi Algorithm (max)

For each state at time  $t$ , keep track of the ***maximum probability of any path*** to it

$$\begin{aligned}\mathbf{m}_{1:t+1} &= \text{VITERBI}(\mathbf{m}_{1:t}, e_{t+1}) \\ &= P(e_{t+1}|X_{t+1}) \max_{x_t} P(X_{t+1}|x_t) \mathbf{m}_{1:t}\end{aligned}$$

# Why is This True?



$$m_1[x_1] = P(e_1|x_1)P(x_1)$$

probability of best path  $1 : t$  that ends at  $x_t$

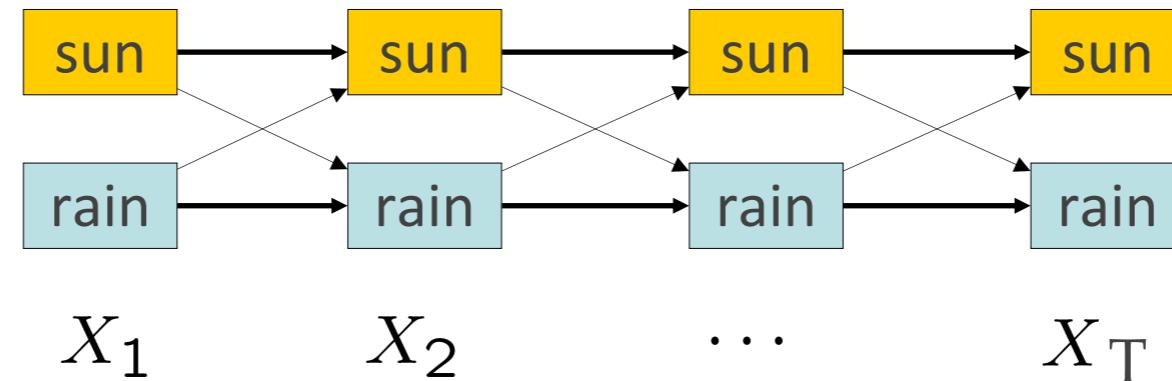
$$m_2[x_2] = \max\{P(e_2|x_2)P(x_2|x_1 = \text{rain})m_1[x_1 = \text{rain}], P(e_2|x_2)P(x_2|x_1 = \text{sun})m_1[x_1 = \text{sun}]\}$$

$$= \max_{x_1} P(e_2|x_2)P(x_2|x_1)m_1[x_1]$$

$$m_t[x_t] = \max_{x_{1:t-1}} P(x_{1:t-1}, x_t, e_{1:t})$$

$$= P(e_t|x_t) \max_{x_{t-1}} P(x_t|x_{t-1})m_{t-1}[x_{t-1}]$$

# How About the MLE Path?



$$m_N[x_N]$$

probability of best path  $1 : N$  that ends at  $x_N$

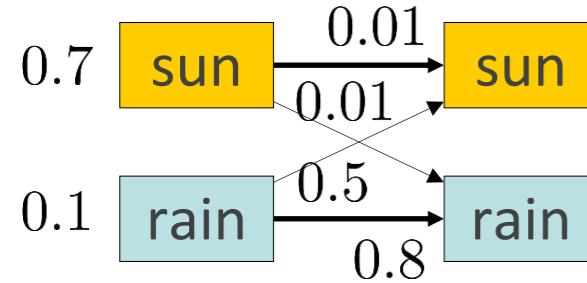
what is the last state on the most likely path?

$$\arg \max_{x_N} m_N[x_N]$$

what is the *second to last* state on the most likely path?

# A Tricky Counter-Example

$$P(e_1|x_1)P(x_1) \quad P(e_2|x_2)P(x_2|x_1)$$



$X_1$

$X_2$

$$\arg \max_{x_1} m_1[x_1] = ?$$

sun!

$$m_2[x_2] = \max_{x_1} P(e_2|x_2)P(x_2|x_1)m_1[x_1]$$

$$m_2[x_2 = \text{sun}] = \max\{0.7 \times 0.01, 0.1 \times 0.5\} = 0.05$$

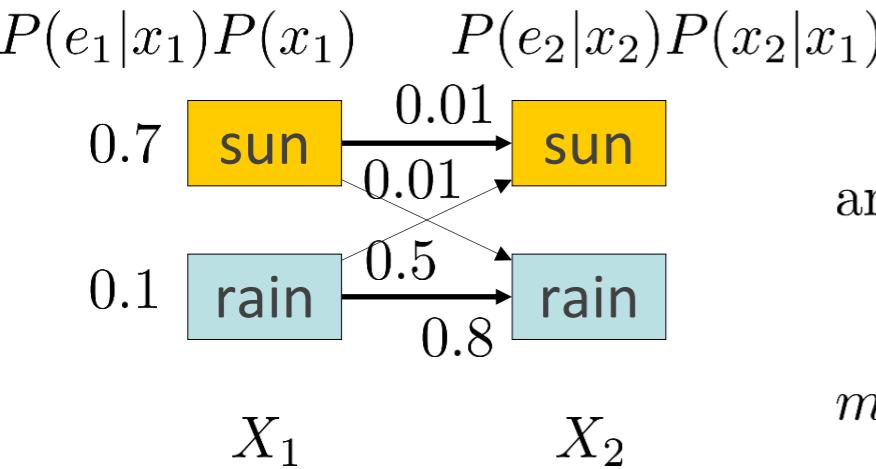
$$m_2[x_2 = \text{rain}] = \max\{0.7 \times 0.01, 0.1 \times 0.8\} = 0.08$$

$$\arg \max_{x_2} m_2[x_2] = \text{rain}$$

$$P(x_1 = \text{sun}, x_2 = \text{rain}, e_1, e_2) = 0.7 \times 0.01 = 0.007$$

best path  $1:t$  that ends at  $x_t \neq$  best path  $1:N$  that goes through  $x_t$

# How to Recover the MLE Path?



$$\arg \max_{x_1} m_1[x_1] = ?$$

sun!

$$m_2[x_2] = \max_{x_1} P(e_2|x_2)P(x_2|x_1)m_1[x_1]$$

this path starts at rain

$$m_2[x_2 = \text{sun}] = \max\{0.7 \times 0.01, 0.1 \times 0.5\} = 0.05$$

$$m_2[x_2 = \text{rain}] = \max\{0.7 \times 0.01, 0.1 \times 0.8\} = 0.08$$

$$\arg \max_{x_2} m_2[x_2] = \text{rain}$$

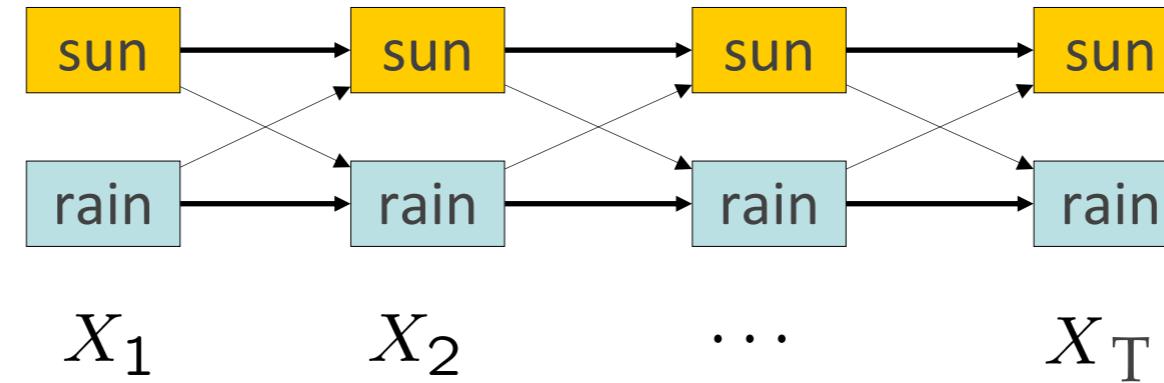
this path starts at rain

idea: what if we also save where the best path came from?

$$m_t[x_t] = \max_{x_{t-1}} P(e_t|x_t)P(x_t|x_{t-1})m_{t-1}[x_{t-1}]$$

$$a_t[x_t] = \arg \max_{x_{t-1}} P(e_t|x_t)P(x_t|x_{t-1})m_{t-1}[x_{t-1}]$$

# Follow the Breadcrumbs...



for  $t = 1$  to  $N$ :

$$m_t[x_t] = \max_{x_{t-1}} P(e_t|x_t)P(x_t|x_{t-1})m_{t-1}[x_{t-1}]$$

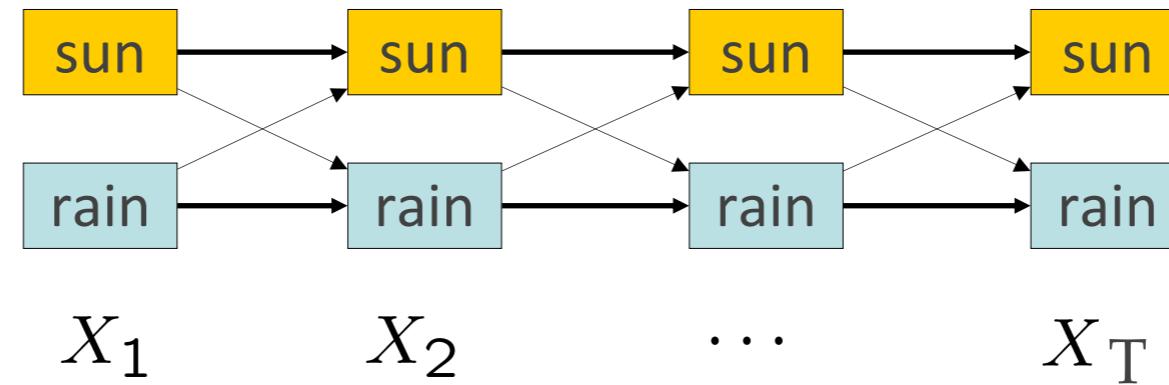
$$a_t[x_t] = \arg \max_{x_{t-1}} P(e_t|x_t)P(x_t|x_{t-1})m_{t-1}[x_{t-1}]$$

last state on most likely path:  $x_N^* = \arg \max_{x_N} m_N[x_N]$

second to last state on most likely path:  $x_{N-1}^* = a_N[x_N^*]$

third to last state on most likely path:  $x_{N-2}^* = a_{N-1}[x_{N-1}^*]$

# Follow the Breadcrumbs...



for  $t = 1$  to  $N$ :

$$m_t[x_t] = \max_{x_{t-1}} P(e_t|x_t)P(x_t|x_{t-1})m_{t-1}[x_{t-1}]$$

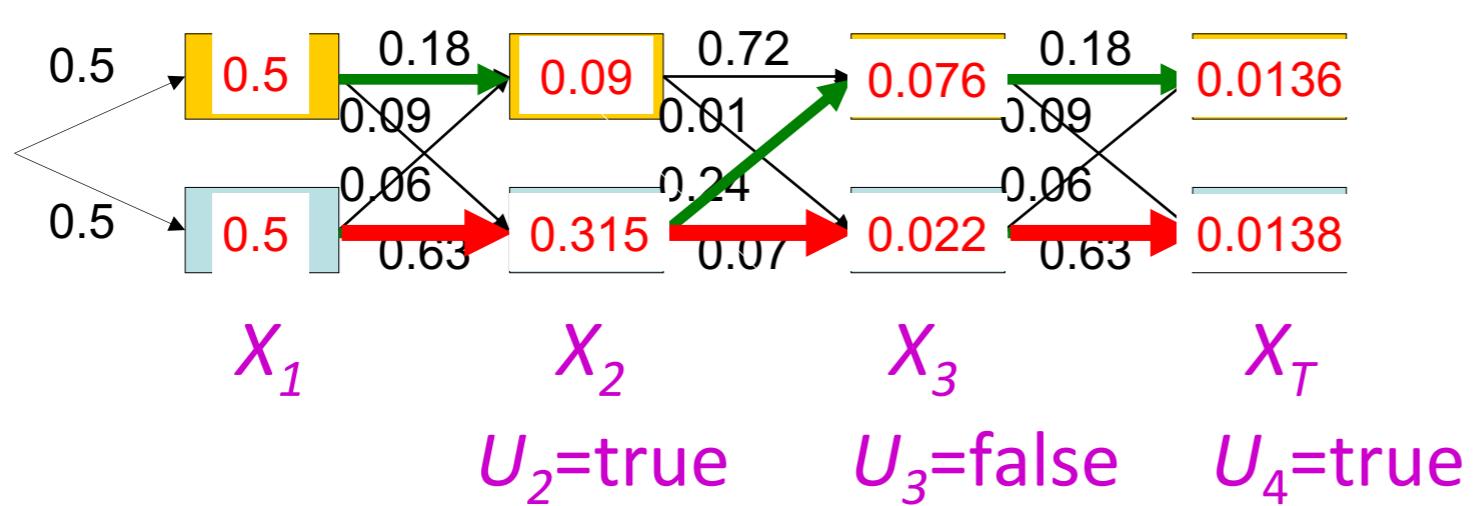
$$a_t[x_t] = \arg \max_{x_{t-1}} P(e_t|x_t)P(x_t|x_{t-1})m_{t-1}[x_{t-1}]$$

$$x_N^* = \arg \max_{x_N} m_N[x_N]$$

for  $t = N$  to 2:

$$x_{t-1}^* = a_t[x_t^*]$$

# Viterbi Algorithm



$W_{t-1}$	$P(W_t   W_{t-1})$	
	sun	rain
sun	0.9	0.1
rain	0.3	0.7

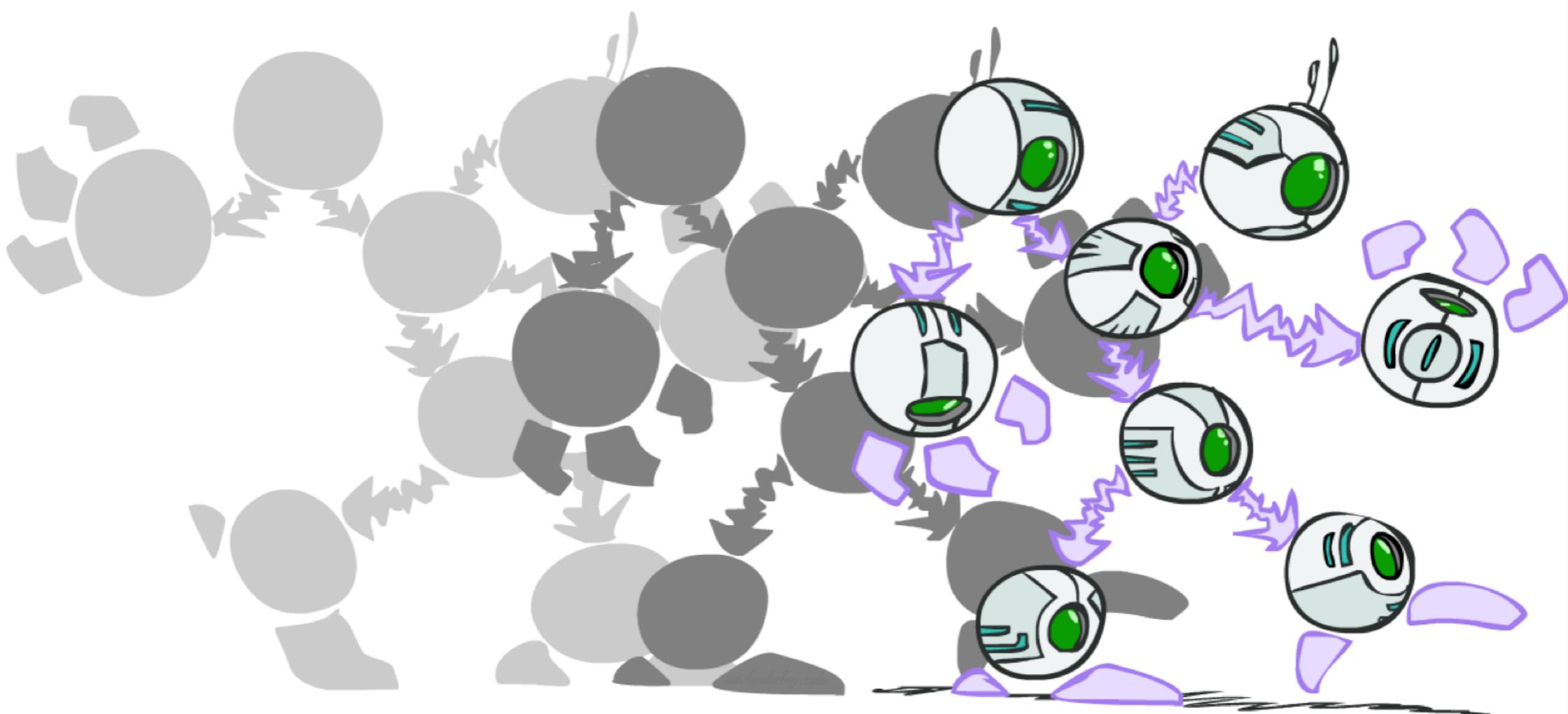
$W_t$	$P(U_t   W_t)$	
	true	false
sun	0.2	0.8
rain	0.9	0.1

Time complexity?  
 **$O(|X|^2 T)$**

Space complexity?  
 **$O(|X| T)$**

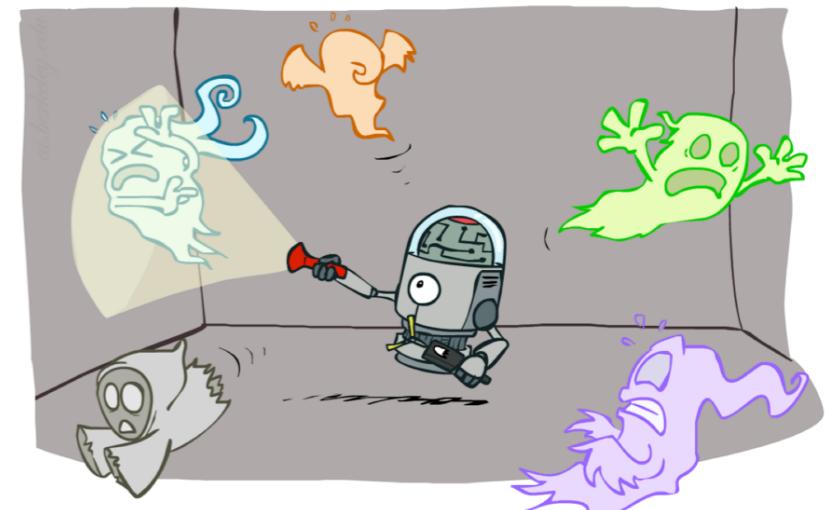
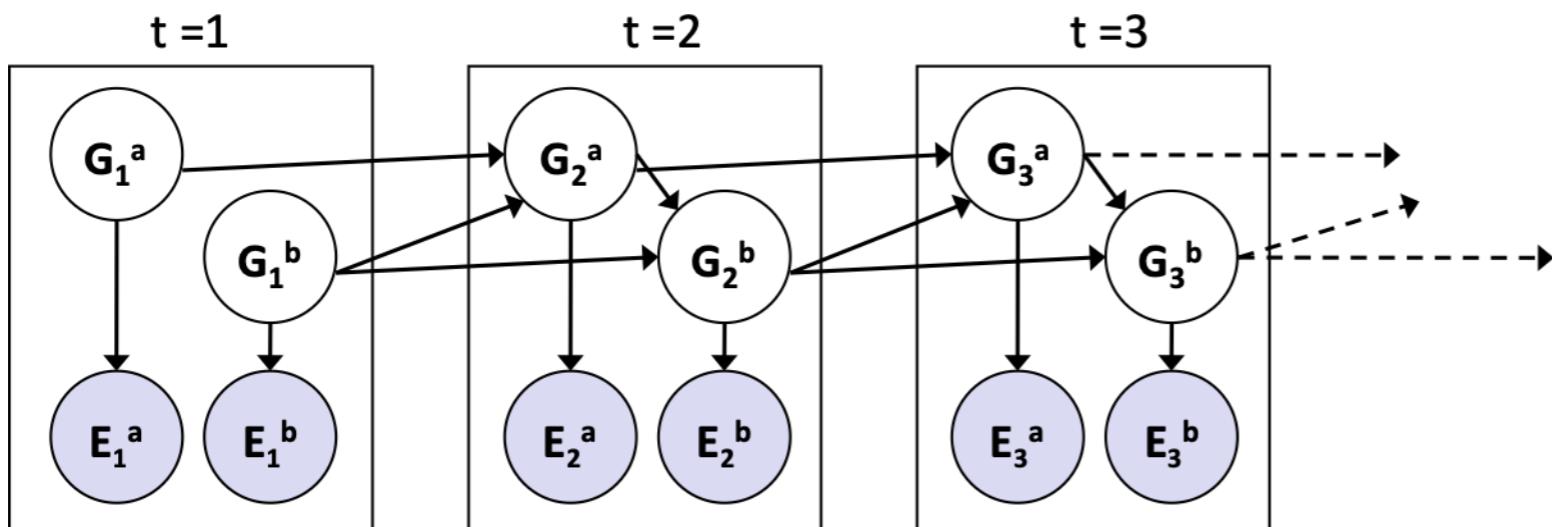
Number of paths?  
 **$O(|X|^T)$**

# Dynamic Bayes Nets



# Dynamic Bayes Nets (DBNs)

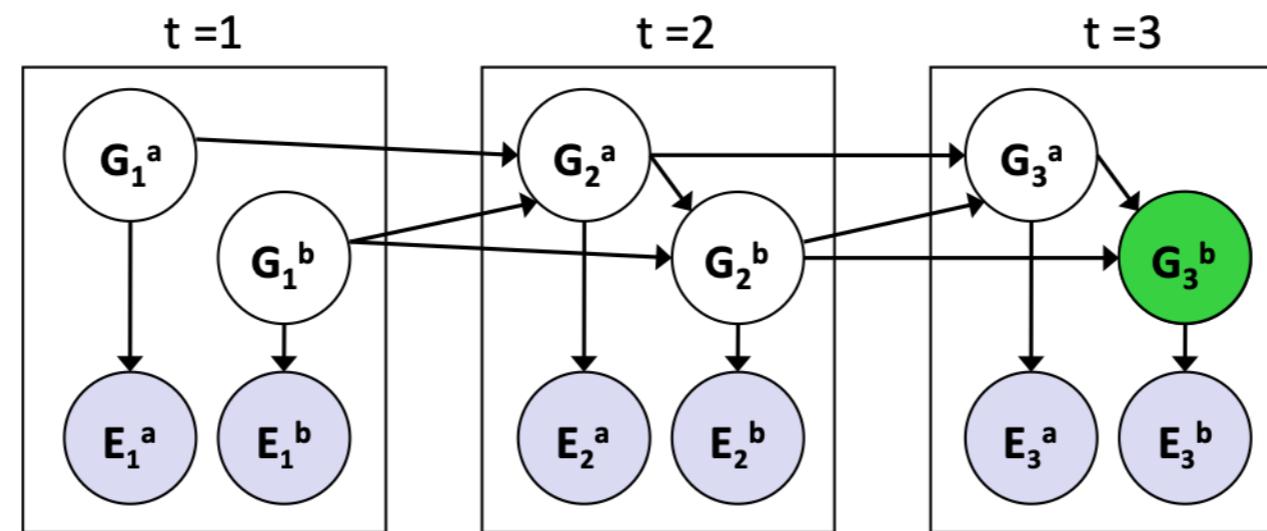
- ❖ We want to track multiple variables over time, using multiple sources of evidence
- ❖ Idea: Repeat a fixed Bayes net structure at each time
- ❖ Variables from time  $t$  can condition on those from  $t-1$



- ❖ Dynamic Bayes nets are a generalization of HMMs

# Exact Inference in DBNs

- ❖ Variable elimination applies to dynamic Bayes nets
- ❖ Procedure: “unroll” the network for  $T$  time steps, then eliminate variables until  $P(X_T | e_{1:T})$  is computed



- ❖ Online belief updates: Eliminate all variables from the previous time step; store factors for current time only

# DBN Particle Filters

- ❖ A particle is a complete sample for a time step
- ❖ **Initialize:** Generate prior samples for the t=1 Bayes net
  - ❖ Example particle:  $\mathbf{G}_1^a = (3,3)$   $\mathbf{G}_1^b = (5,3)$
- ❖ **Elapse time:** Sample a successor for each particle
  - ❖ Example successor:  $\mathbf{G}_2^a = (2,3)$   $\mathbf{G}_2^b = (6,3)$
- ❖ **Observe:** Weight each entire sample by the likelihood of the evidence conditioned on the sample
  - ❖ Likelihood:  $P(E_1^a | \mathbf{G}_1^a) * P(E_1^b | \mathbf{G}_1^b)$
- ❖ **Resample:** Select prior samples (tuples of values) in proportion to their likelihood