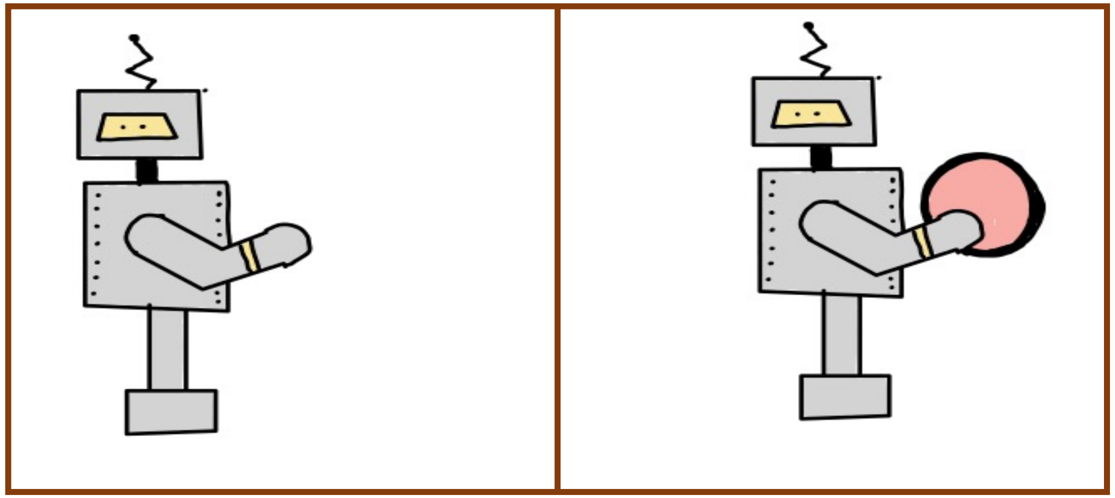
Opportunistic Learning for Markov Decision Systems with Application to Smart Robots



Michael J. Neely
University of Southern California
Allerton Conf. on Communication, Control, Computing, Sept. 2024

Markov Decision Model

- Time slots t = {0, 1, 2, …}
- State pair (S(t), W(t)):

  S(t) in {1, …, n} (*basic state*)

  W(t) iid vector (*arbitrary dimension, unknown distribution*)
- Every slot t:

  Observe: (S(t), W(t))

  Choose: A(t) in $\mathcal{A}$ (*action set has arbitrary cardinality*)
- Transition prob for S(t+1) and cost vector depend on (S(t),W(t),A(t)).

Time Average Goal

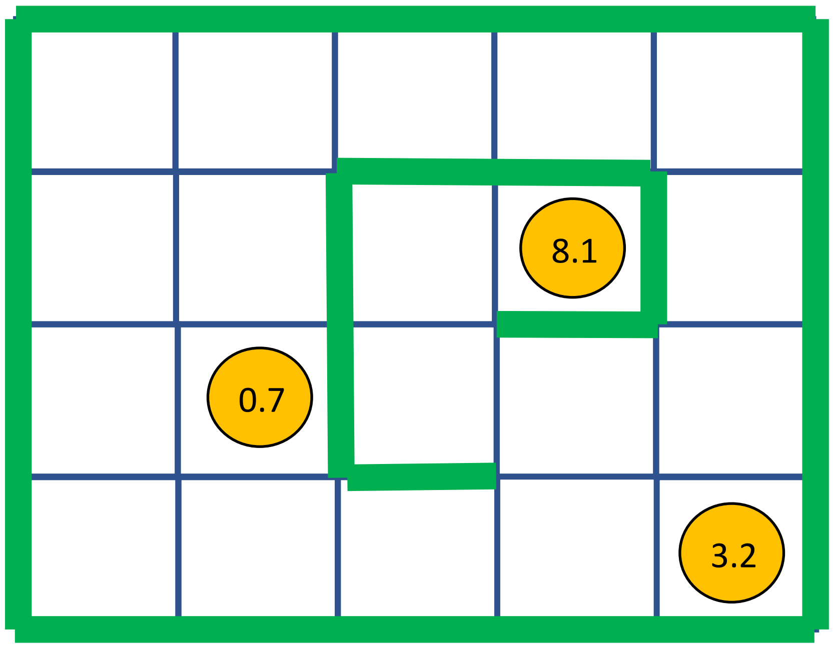
Vector of costs $(C_0(t), C_1(t), \ldots, C_k(t))$.

Minimize: $\overline{C_0}$

Subject to: $\overline{C_i} \leq 0$ for i in $\{1, \ldots, k\}$

(Infinite horizon time averages)

Toy Example: Roving Robot

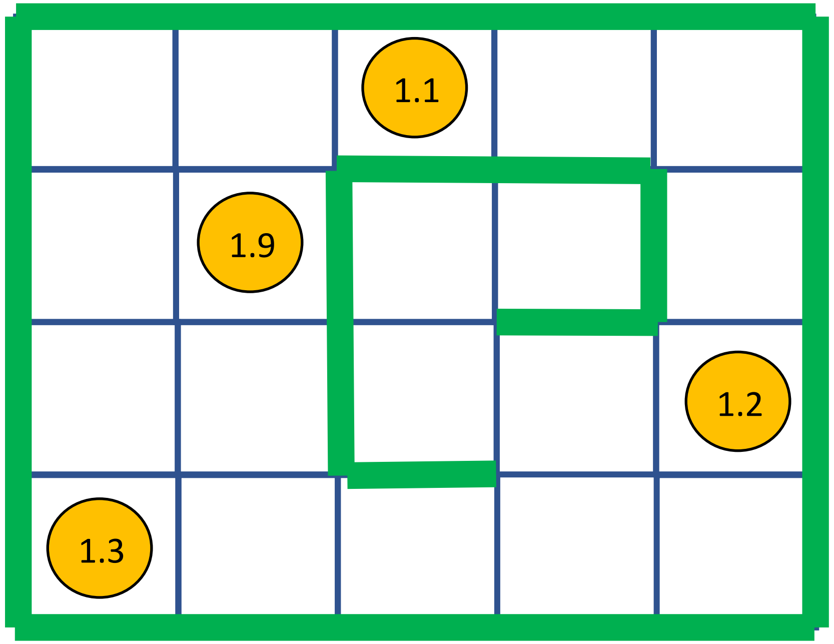


$= W(t)$

- Robot moves over grid of 20 locations
- Valuable objects randomly appear and disappear in each location

  (*iid over slots , unknown joint distribution*)
- Robot has global view of current rewards $W(t) = (W_1(t), \ldots, W_{20}(t))$

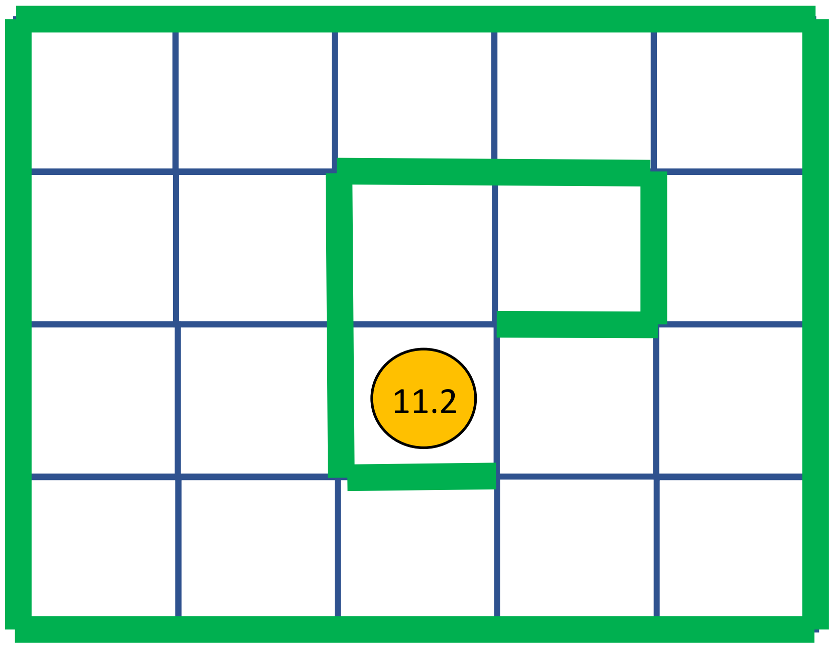
Toy Example: Roving Robot



= W(t)

- Robot moves over grid of 20 locations
- Valuable objects randomly appear and disappear in each location
  (*iid over slots , unknown joint distribution*)
- Robot has global view of current rewards $W(t) = (W_1(t), \ldots, W_{20}(t))$

Toy Example: Roving Robot

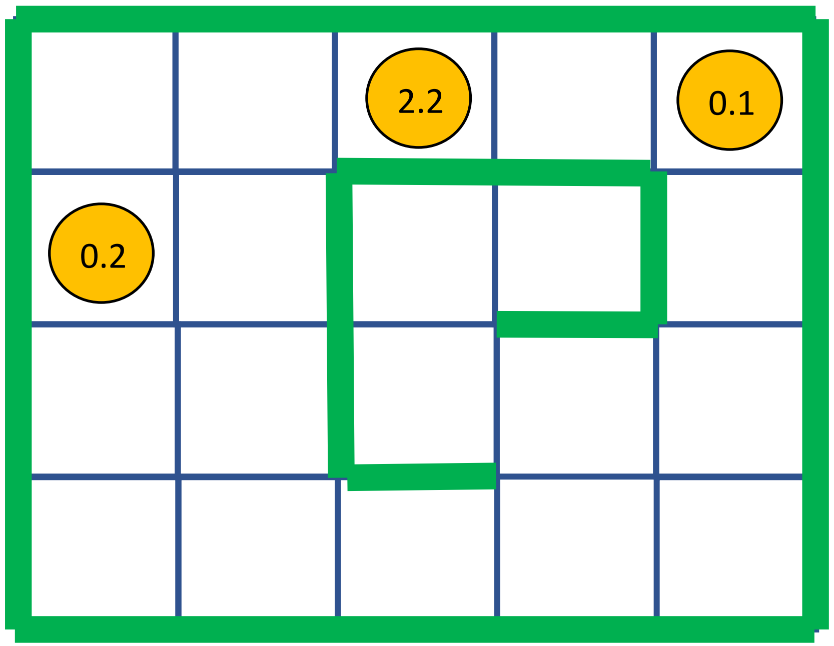


= W(t)

- Robot moves over grid of 20 locations
- Valuable objects randomly appear and disappear in each location

  (*iid over slots , unknown joint distribution*)
- Robot has global view of current rewards $W(t) = (W_1(t), \ldots, W_{20}(t))$

Toy Example: Roving Robot

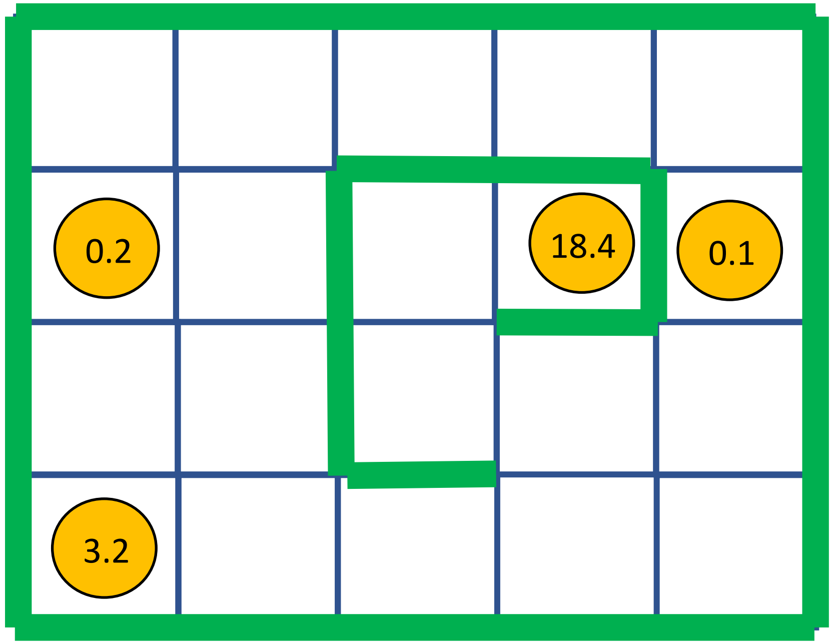


= W(t)

- Robot moves over grid of 20 locations
- Valuable objects randomly appear and disappear in each location

    (*iid over slots , unknown joint distribution*)
- Robot has global view of current rewards $W(t) = (W_1(t), \ldots, W_{20}(t))$

Toy Example: Roving Robot



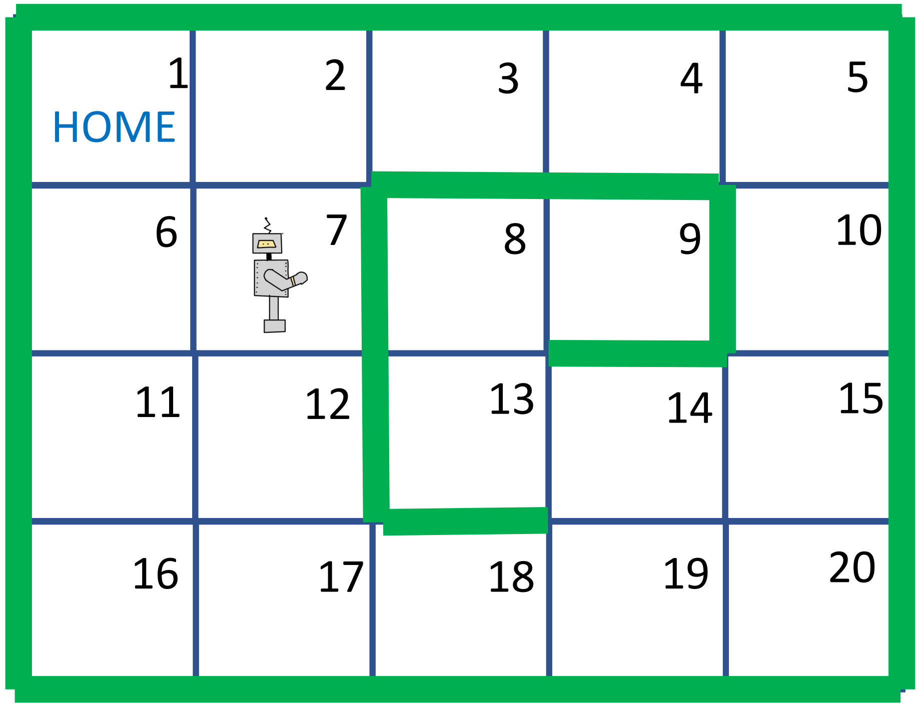
$= W(t)$

- Robot moves over grid of 20 locations
- Valuable objects randomly appear and disappear in each location

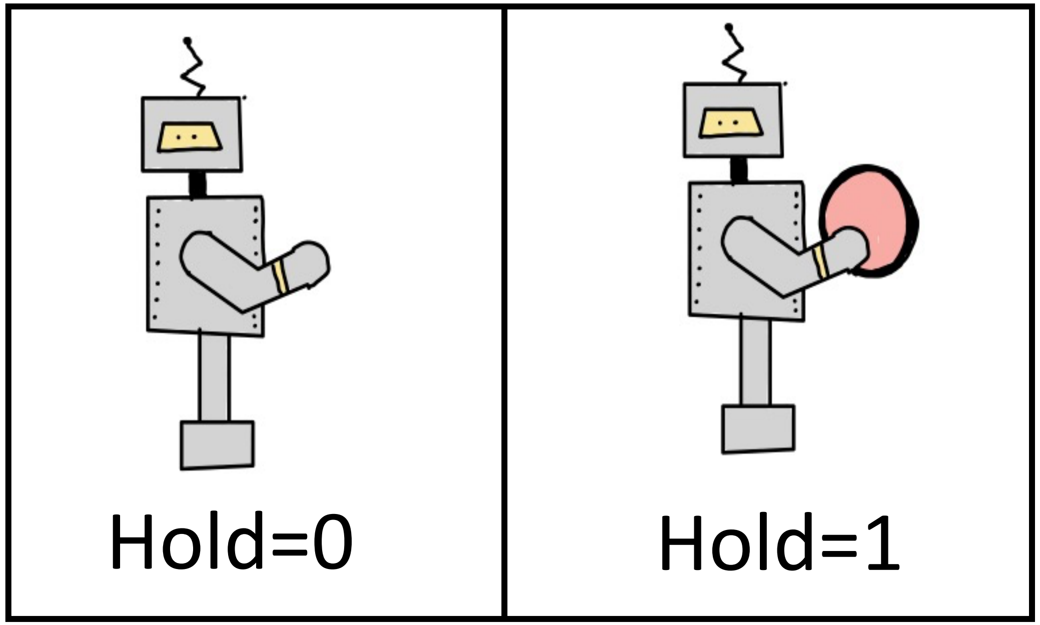
  (*iid over slots , unknown joint distribution*)
- Robot has global view of current rewards $W(t) = (W_1(t), \ldots, W_{20}(t))$

Basic State: S(t) = (Location(t), Hold(t))

Location(t) in {1, ..., 20}



Hold(t) in {0,1}



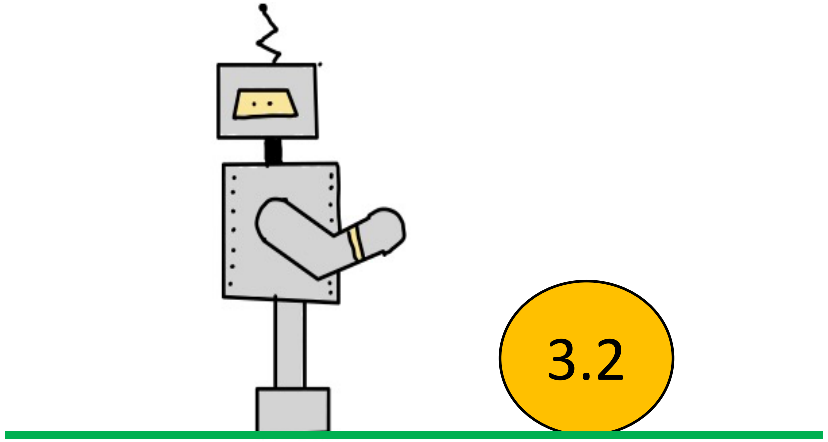
20 x 2 = 40 basic states

Rules

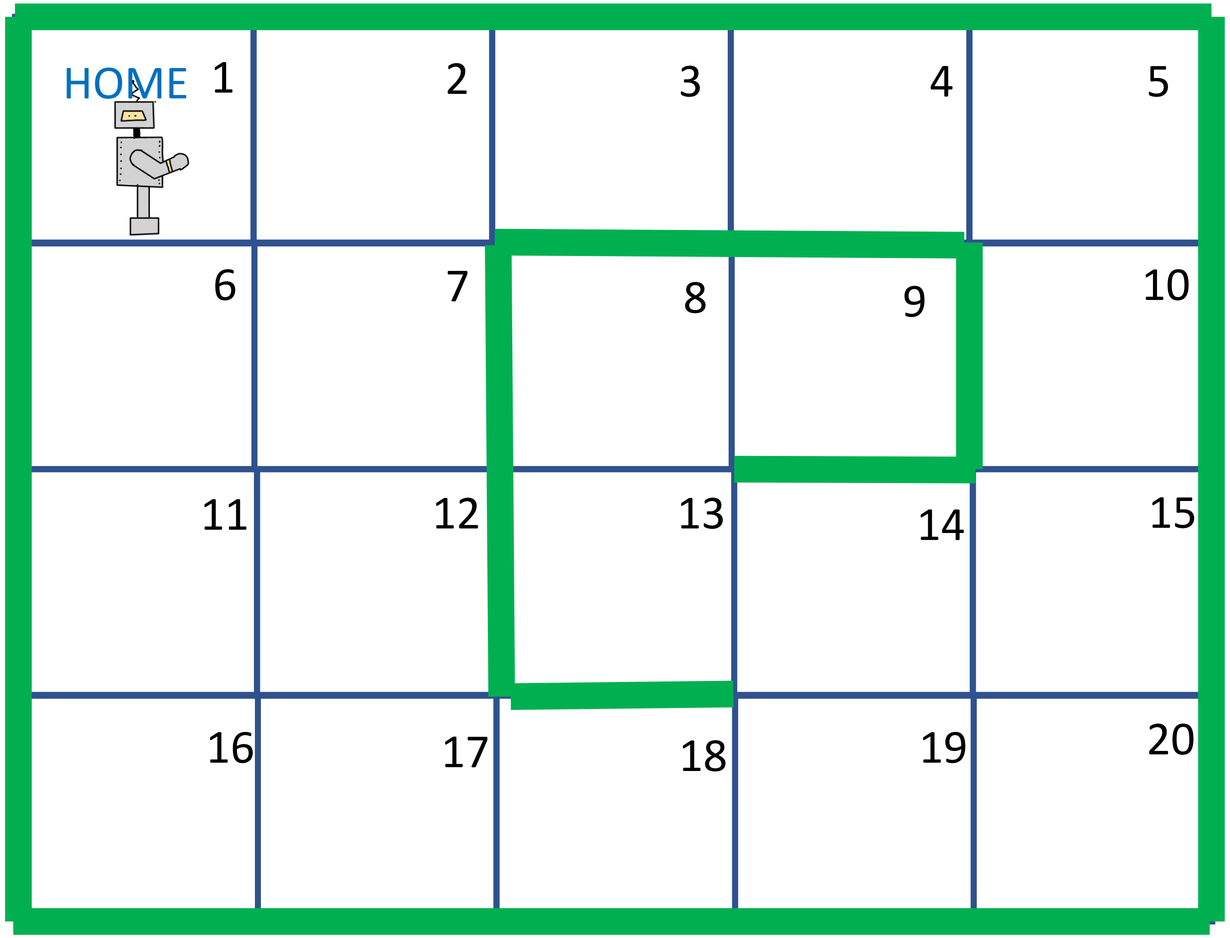
- Robot can hold at most one object at a time
- Can only drop an object at home base (deposit there for points)
- Every slot t:
    1. Robot decides whether or not to pick up object (if any) at current location
    2. Robot then decides to either stay in current location, or move one step in any *feasible direction*: {Stay, N, S, W, E}.

Action: (Pickup(t), Move(t))

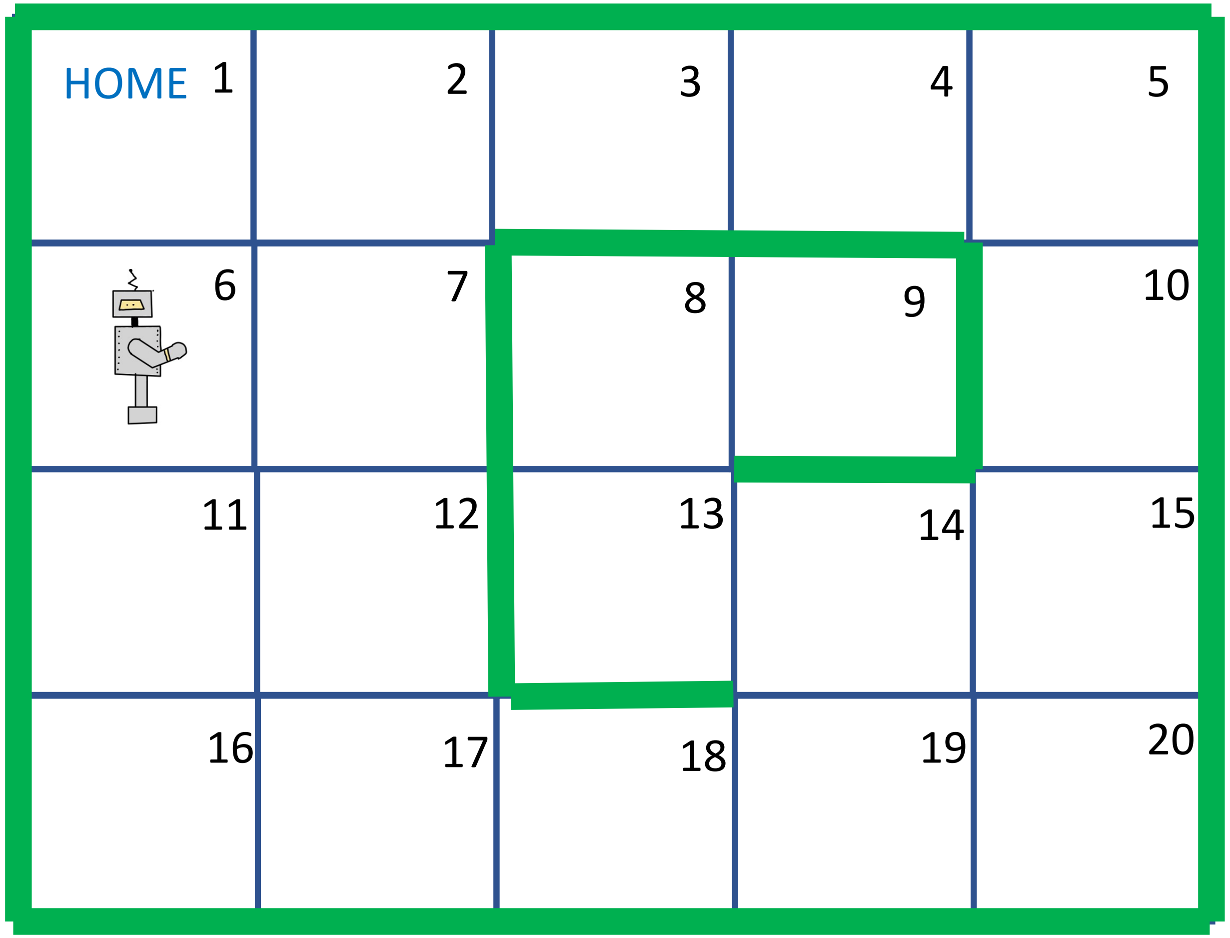
Should I pick this up?



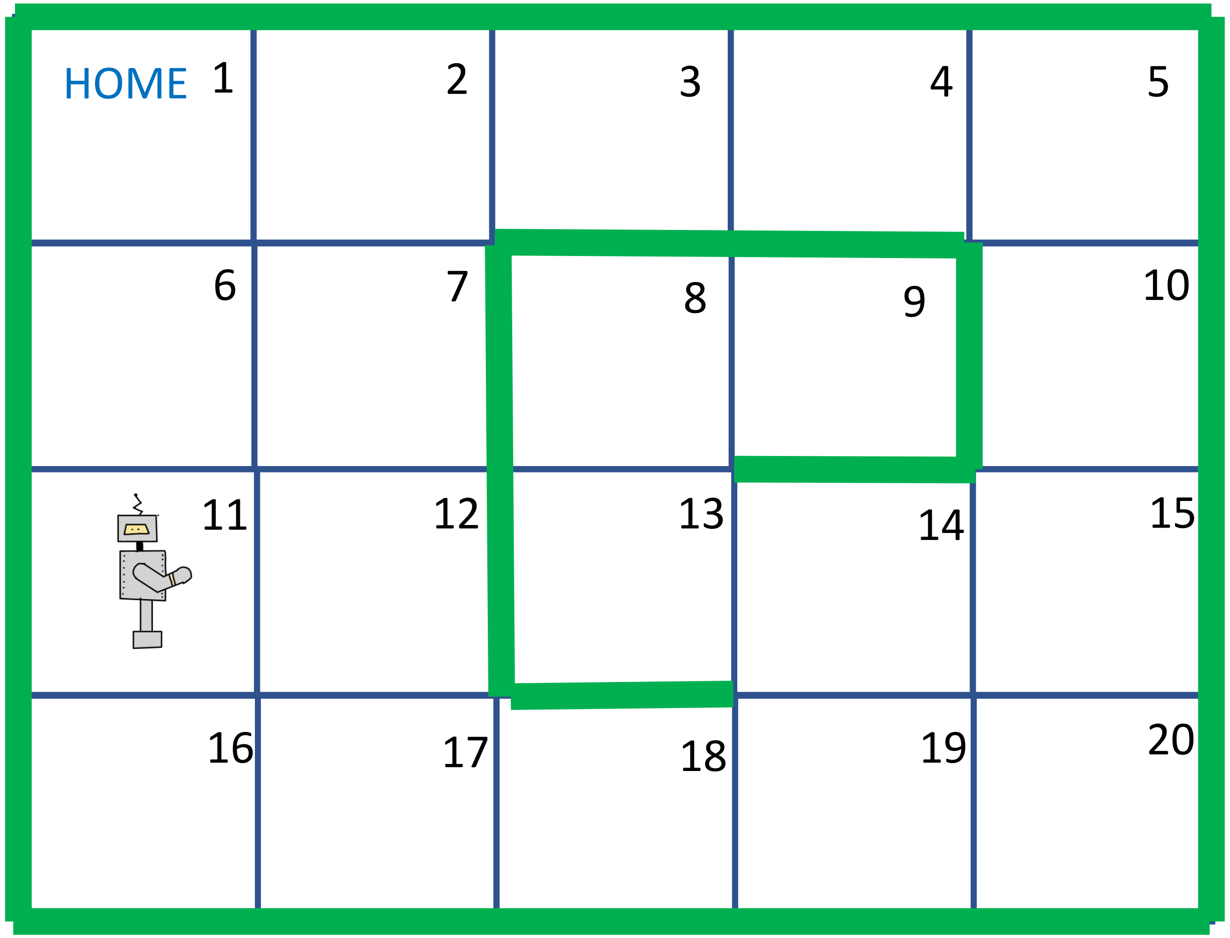
Where should I move next?



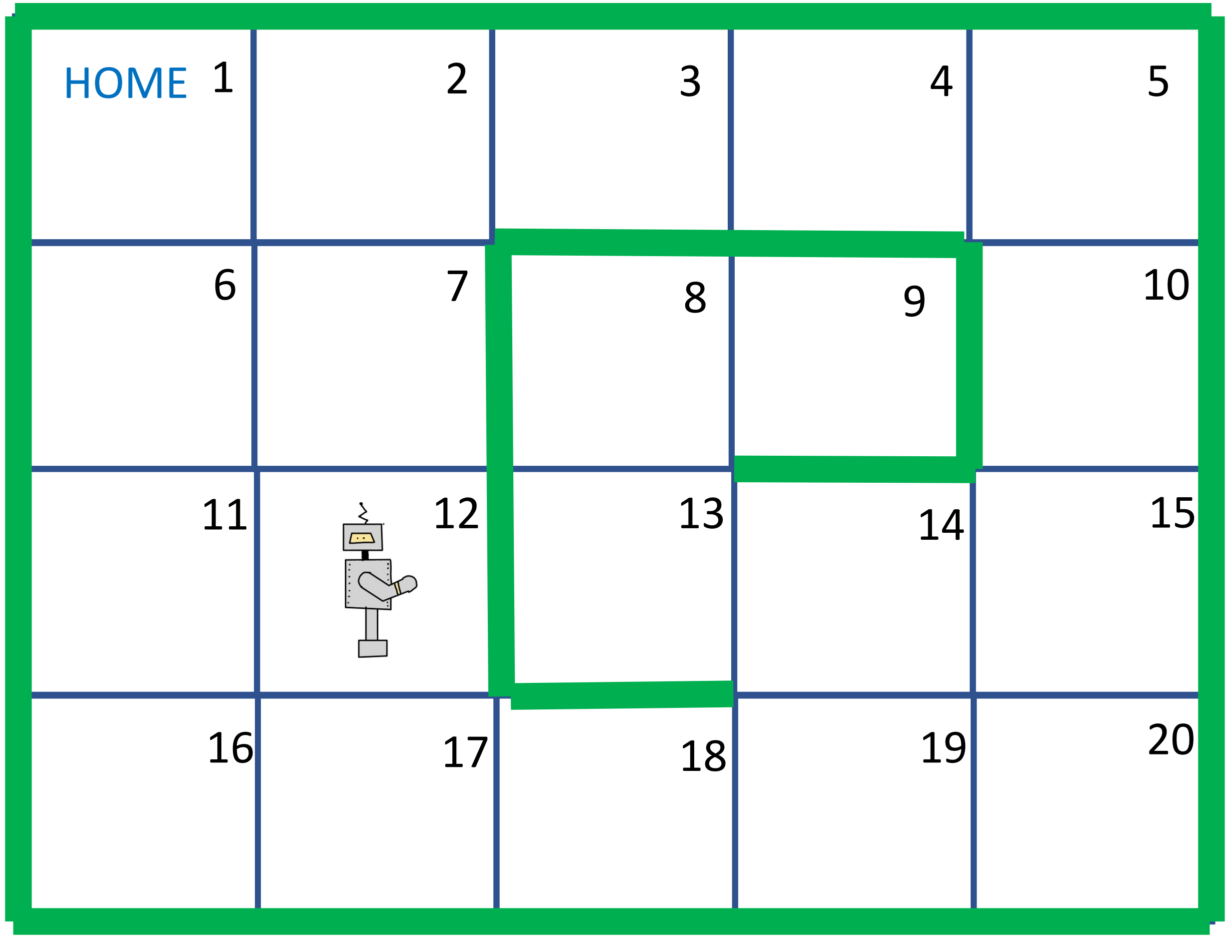
$$S(t) = (1,0)$$



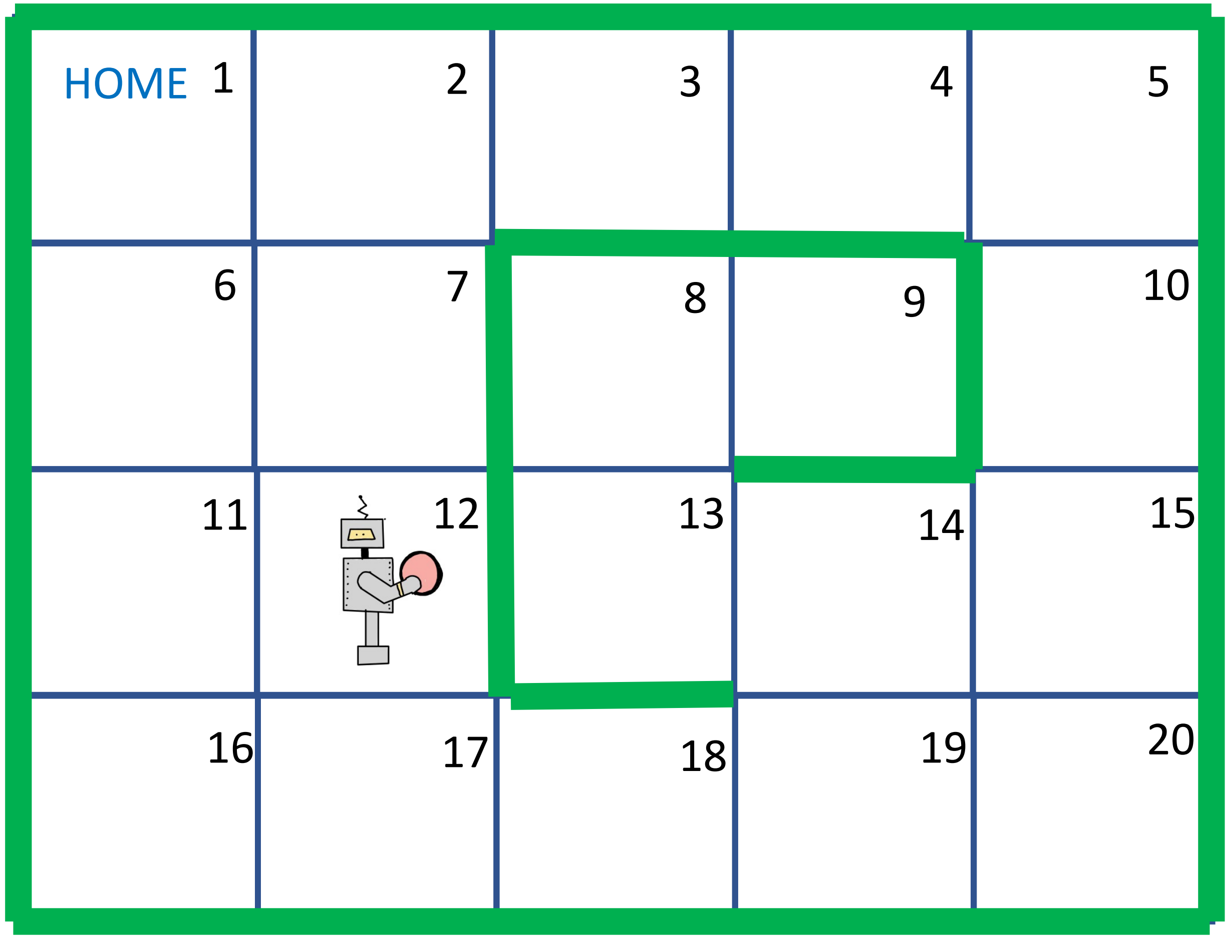
$S(t) = (6,0)$



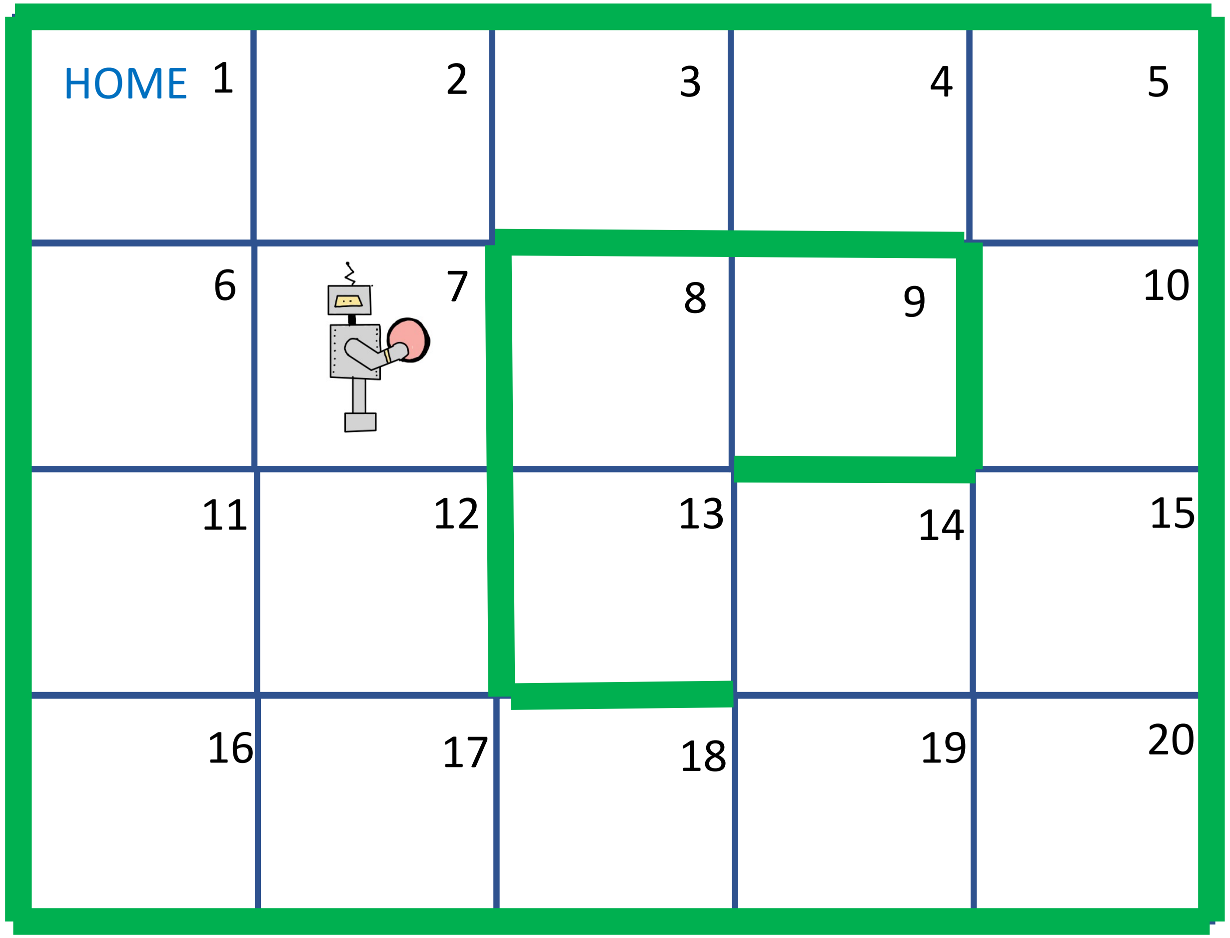
$$S(t) = (11,0)$$



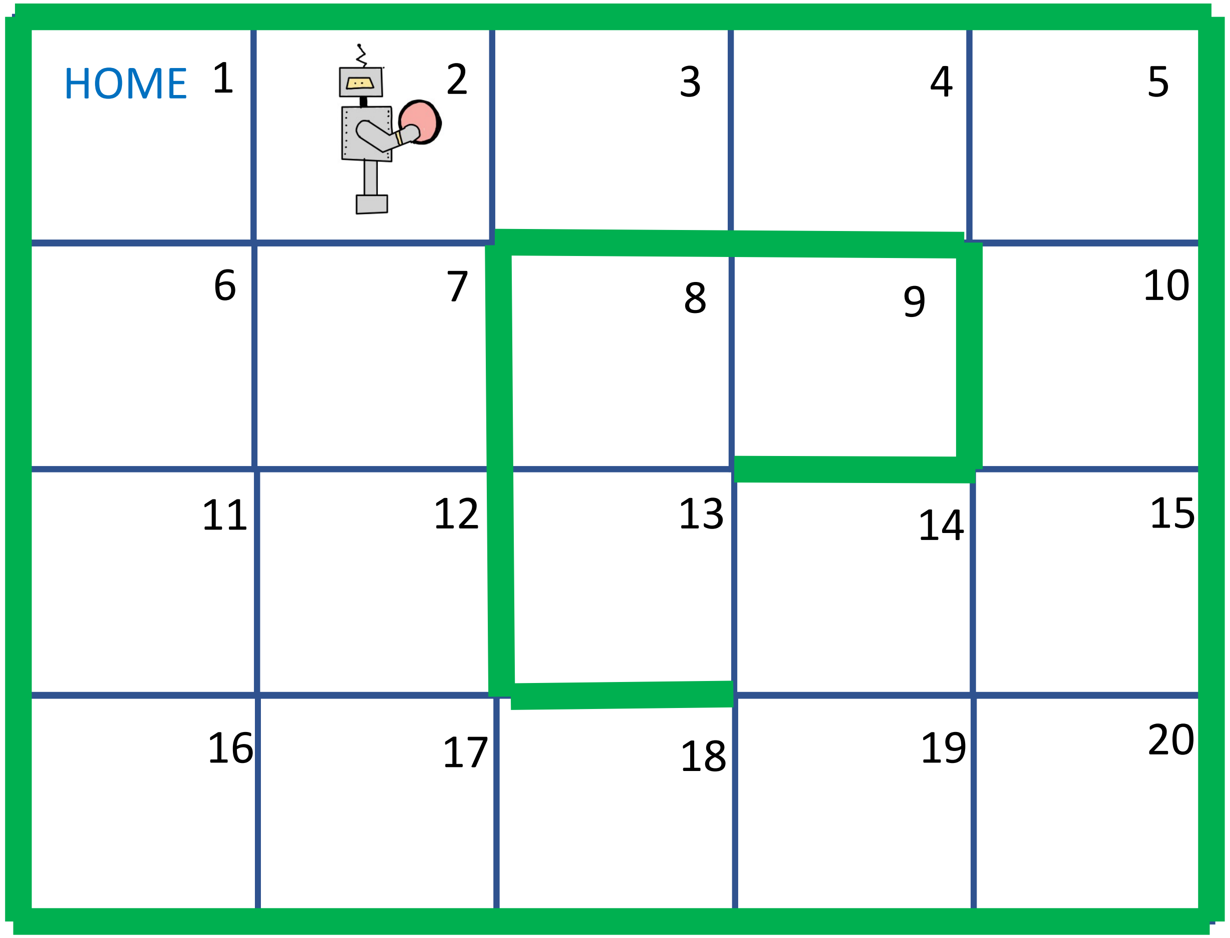
$S(t) = (12,0)$



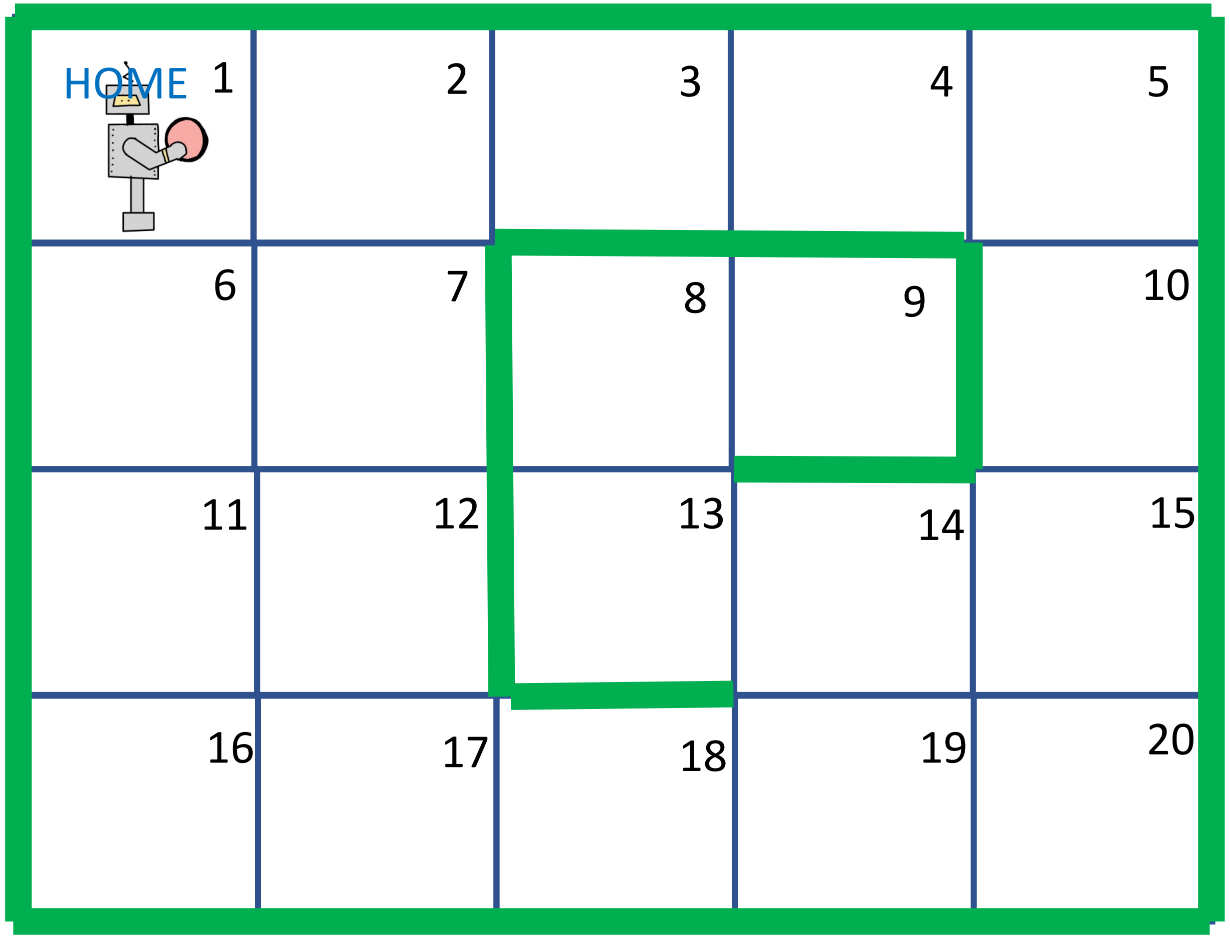
$$S(t) = (12,1)$$



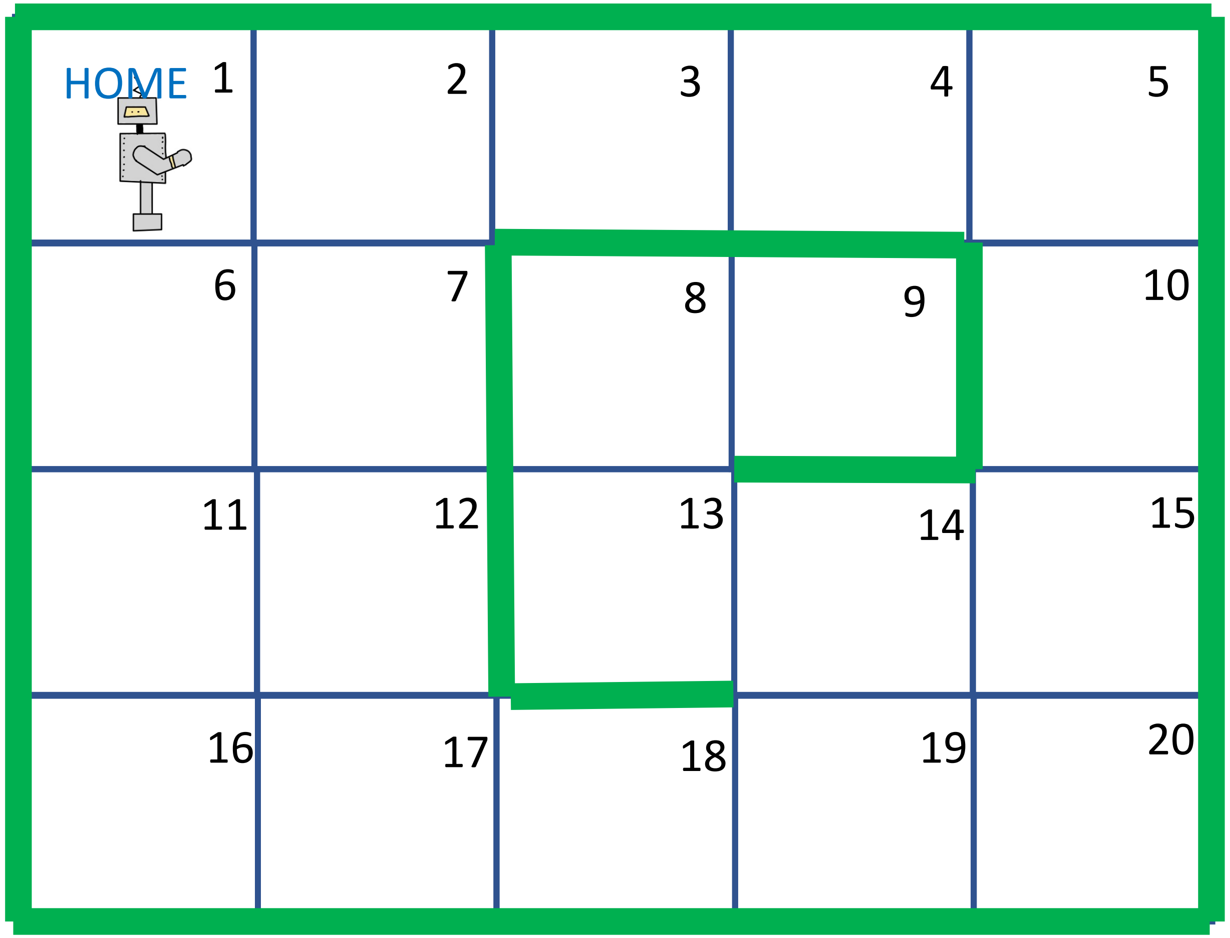
$S(t) = (7,1)$



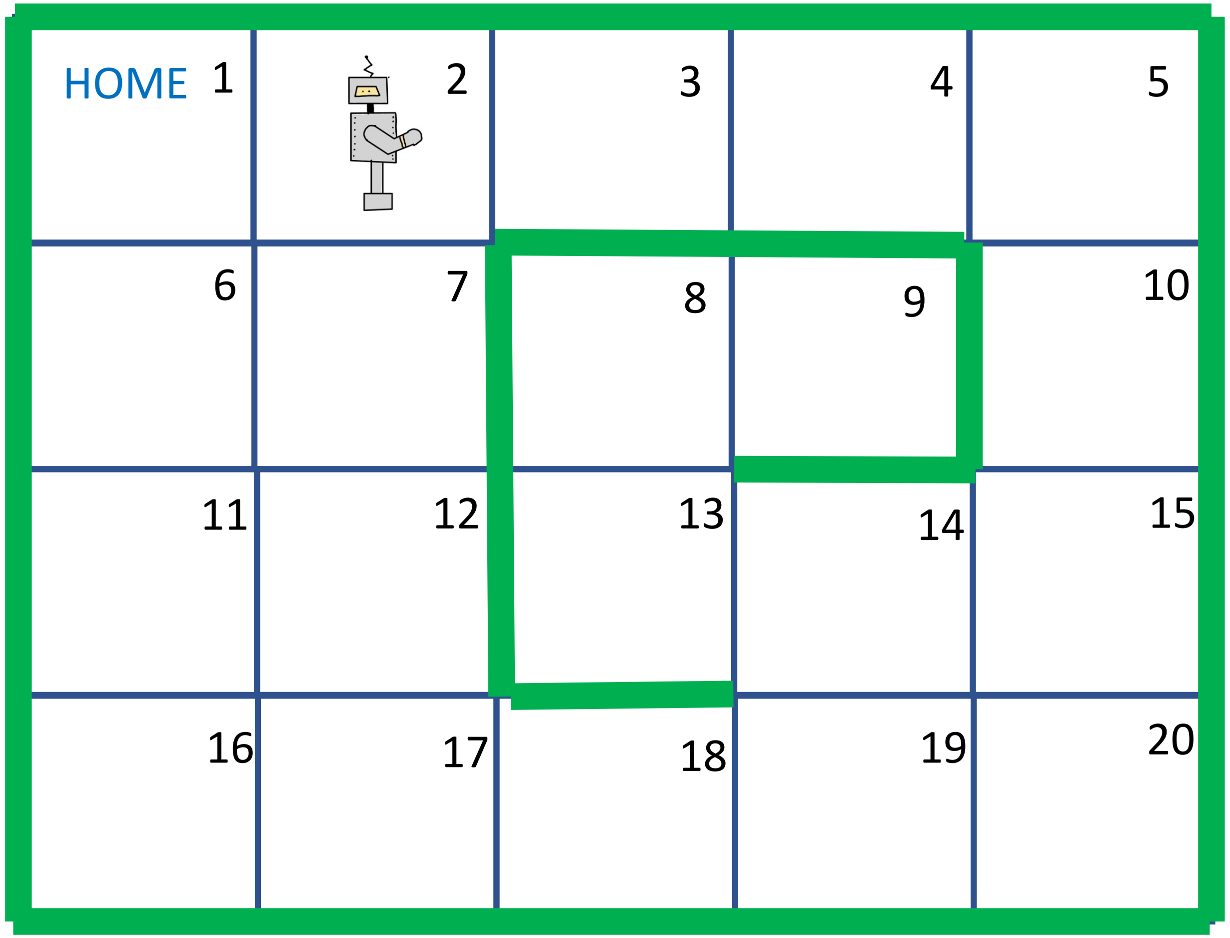
$S(t) = (2,1)$



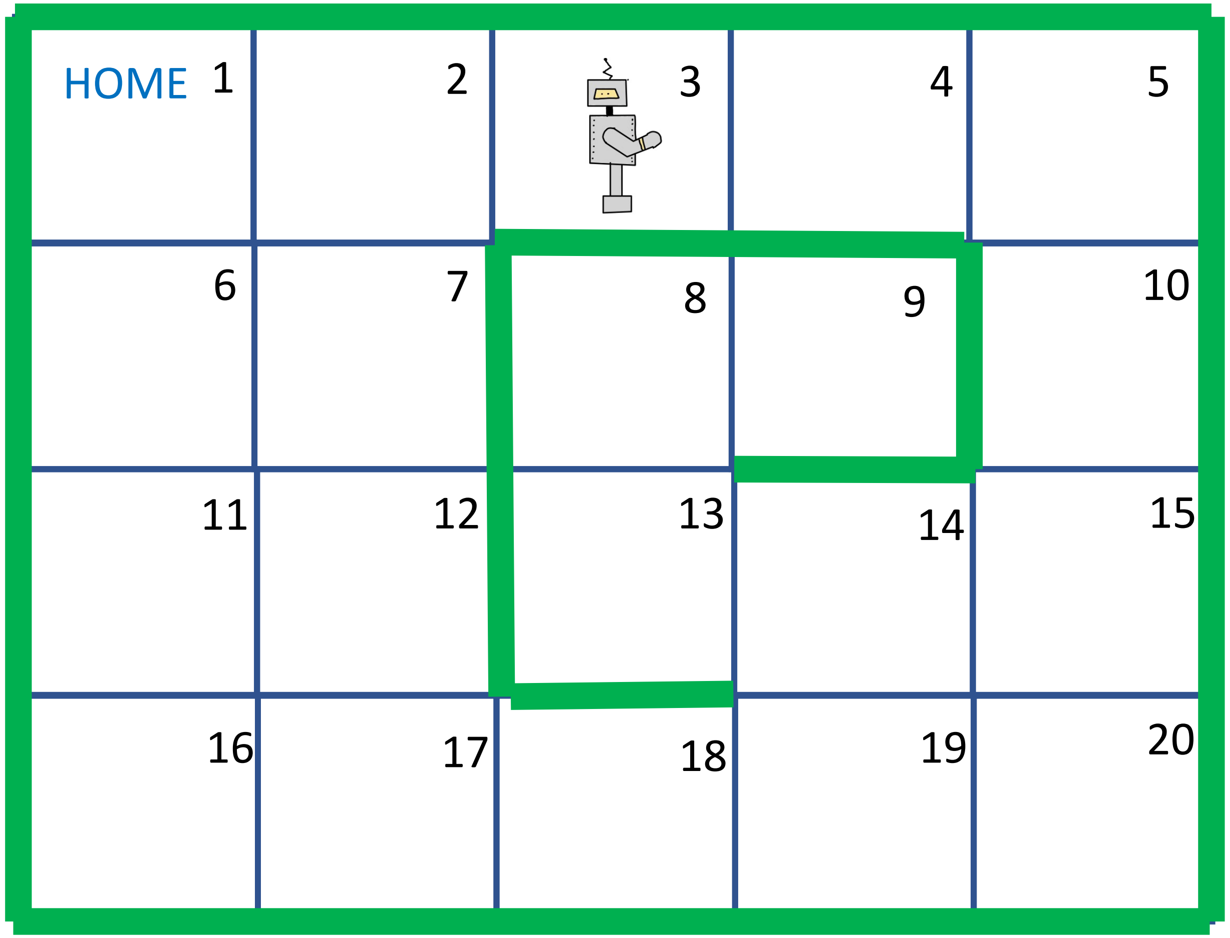
$$S(t) = (1,1)$$



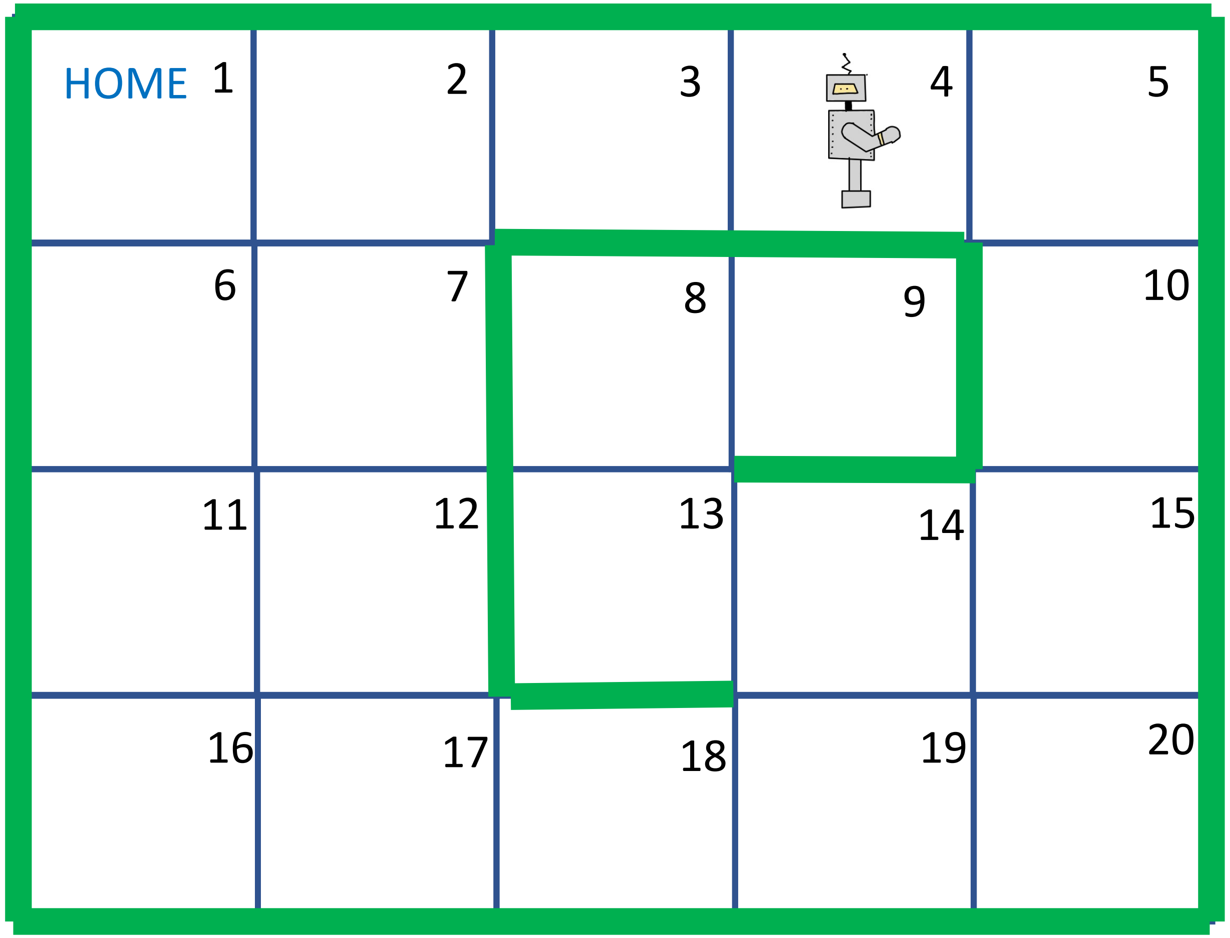
$$S(t) = (1,0)$$



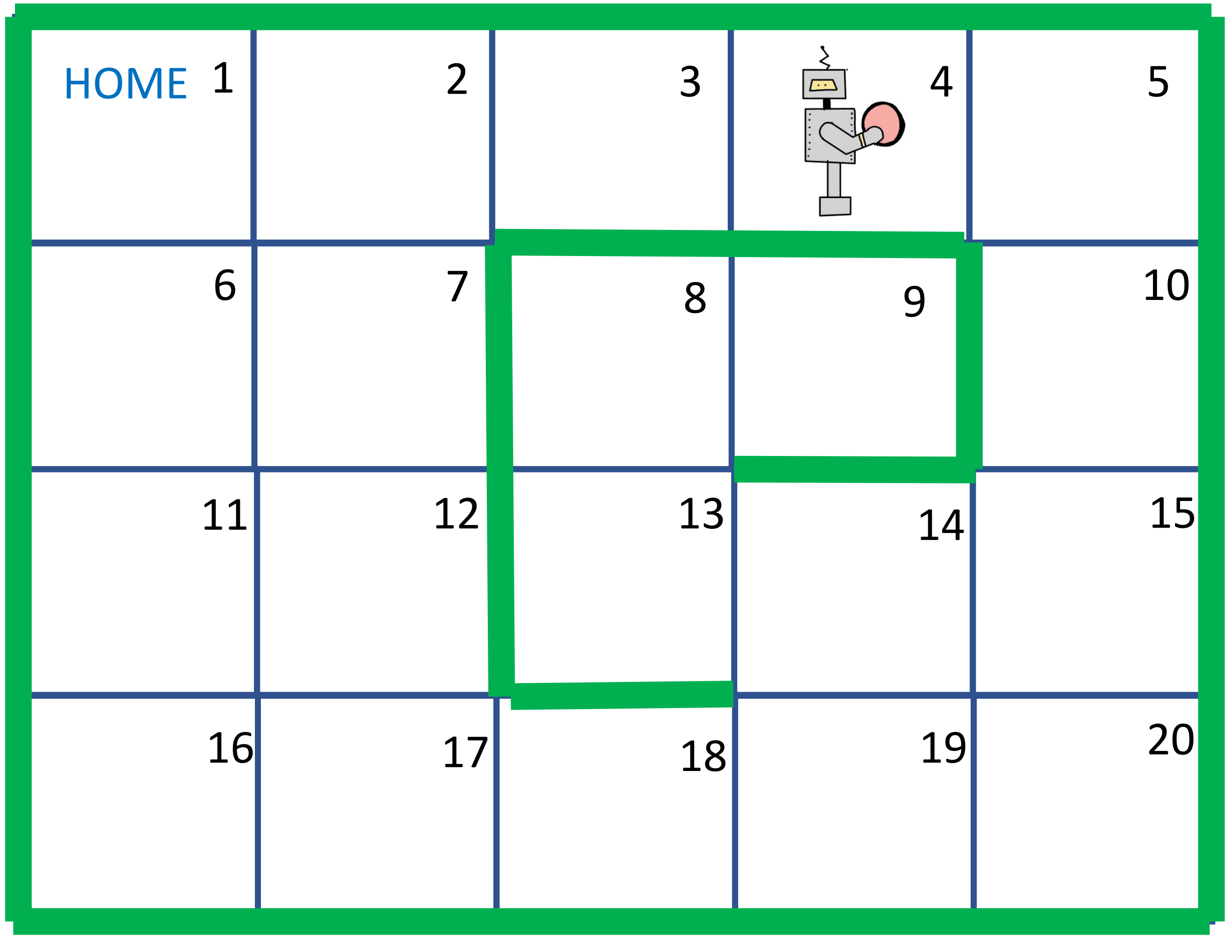
$$S(t) = (2,0)$$



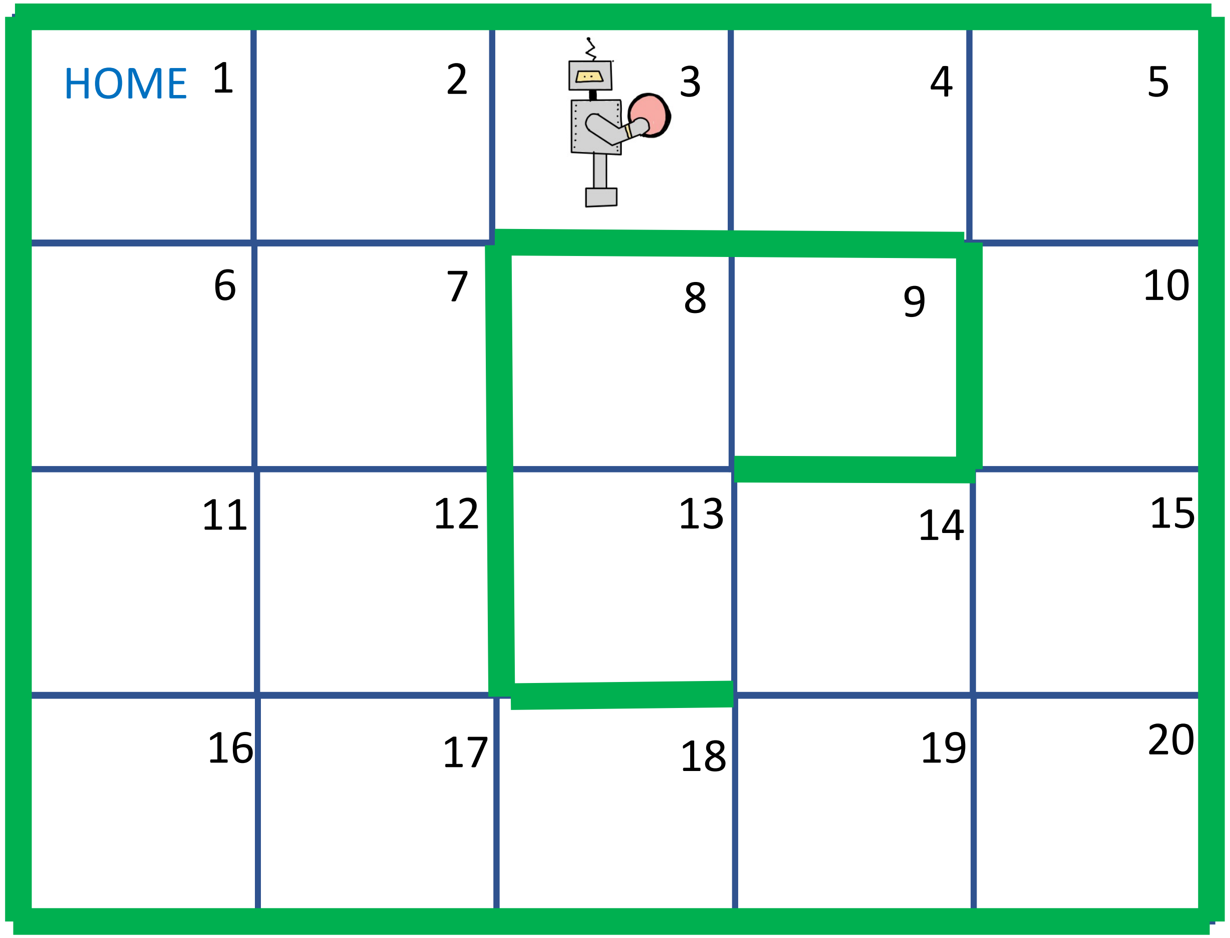
$S(t) = (3,0)$



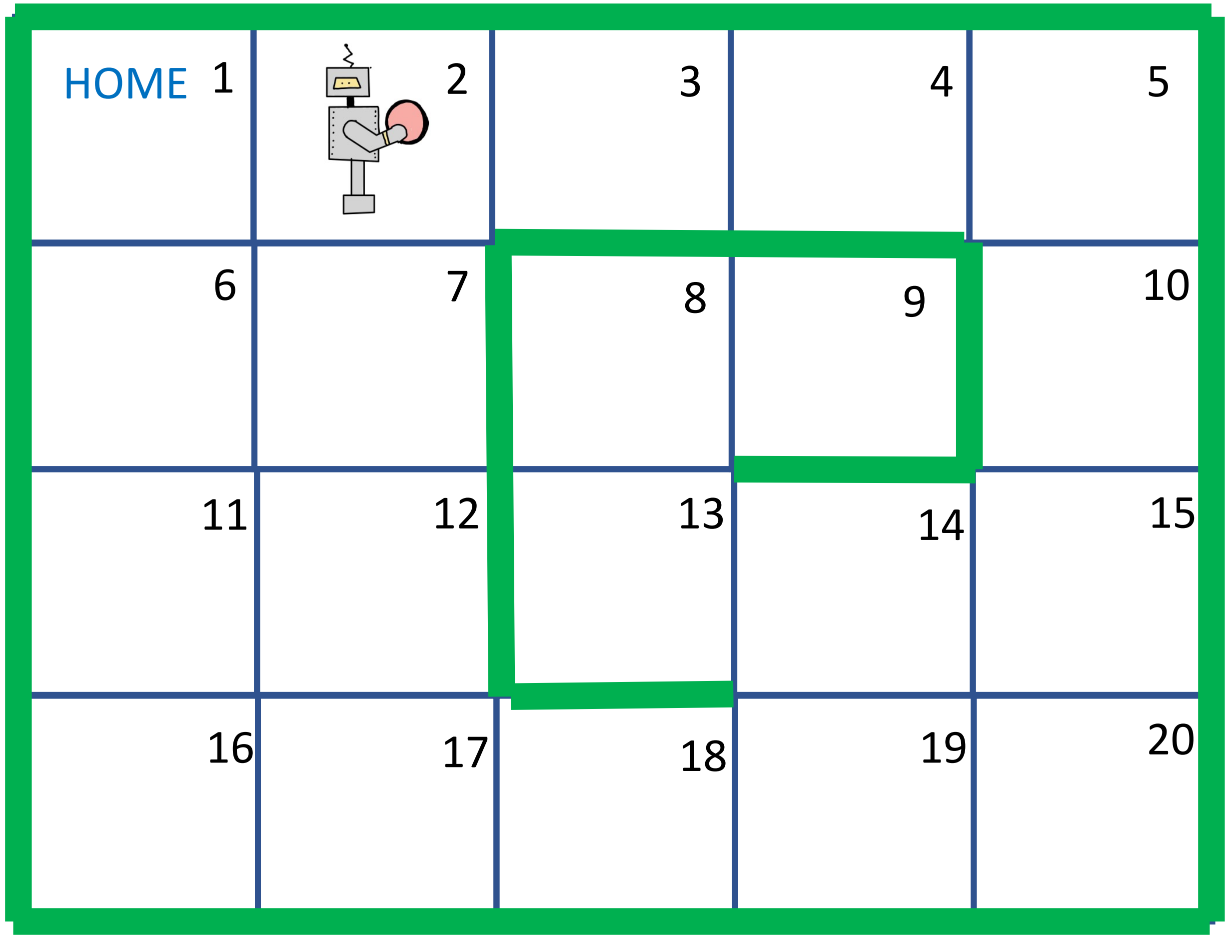
$$S(t) = (4,0)$$



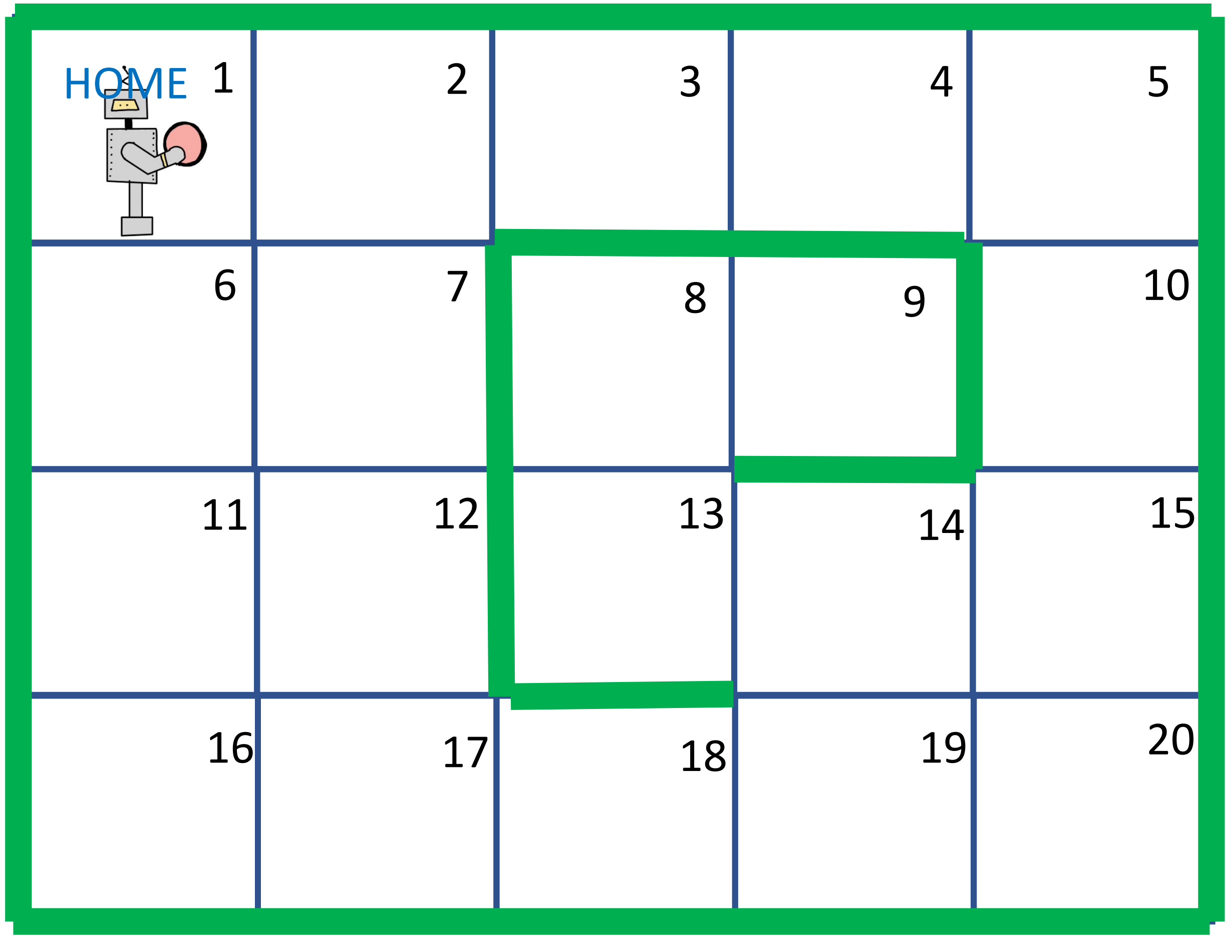
$$S(t) = (4,1)$$



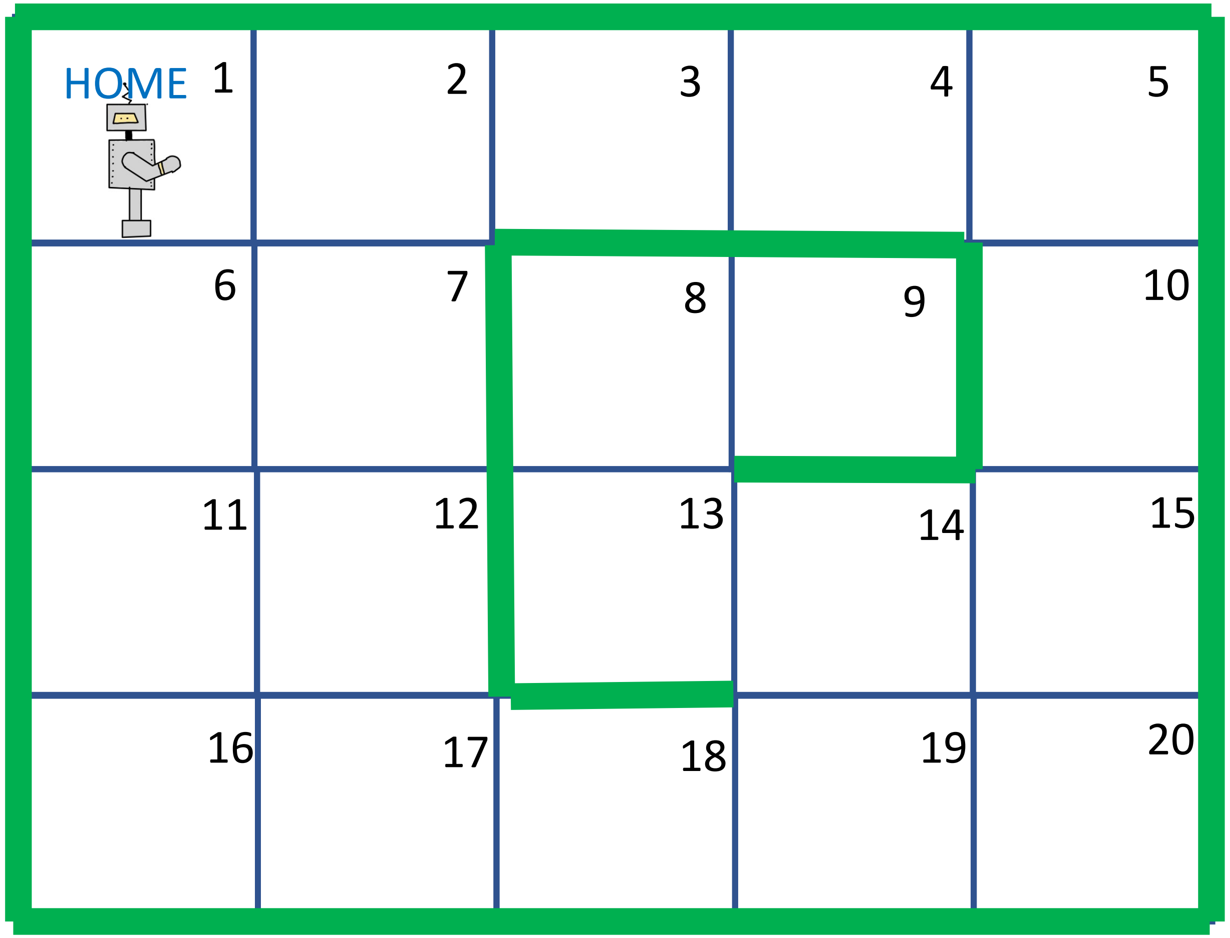
$S(t) = (3,1)$



$$S(t) = (2,1)$$



$S(t) = (1,1)$



$S(t) = (1,0)$

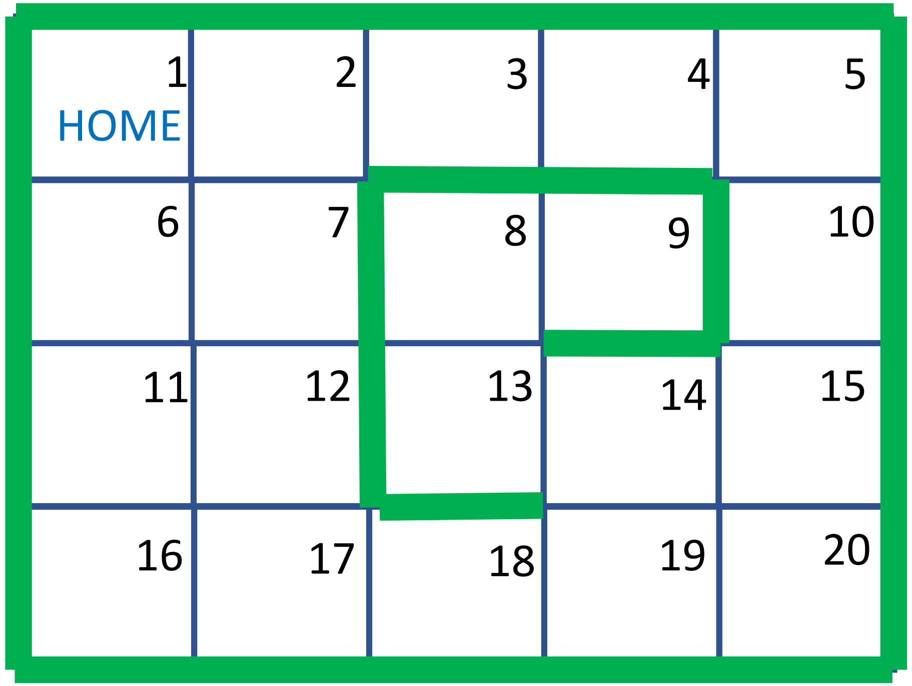
This paper…

- Develops Lyapunov drift based online method for general opportunistic MDPs

- Applies general method to the robot problem

Before describing the proposed algorithm (which does not know the distribution of W(t)), lets consider its performance for the robot problem in comparison to a well reasoned heuristic ***optimized with knowledge of the distribution***.

Compare to Fine-Tuned Heuristic



Rewards appear iid Bern(1/2), then:

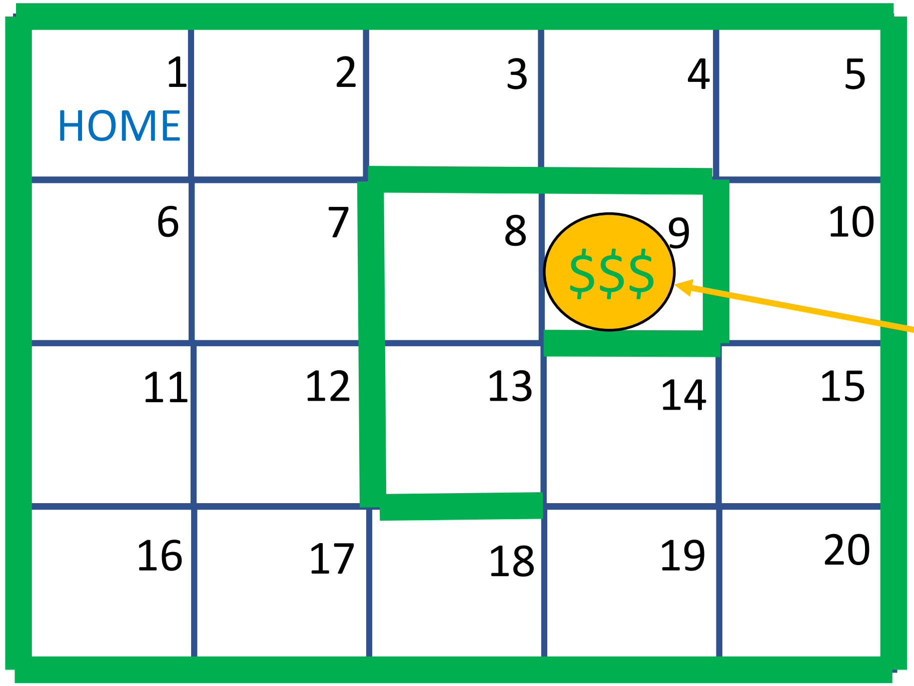
$R_9(t) \sim U[0,20]$

$R_{16}(t) \sim U[0,4]$

$R_i(t) \sim U[0,1]$

Distribution for rewards

Compare to Fine-Tuned Heuristic



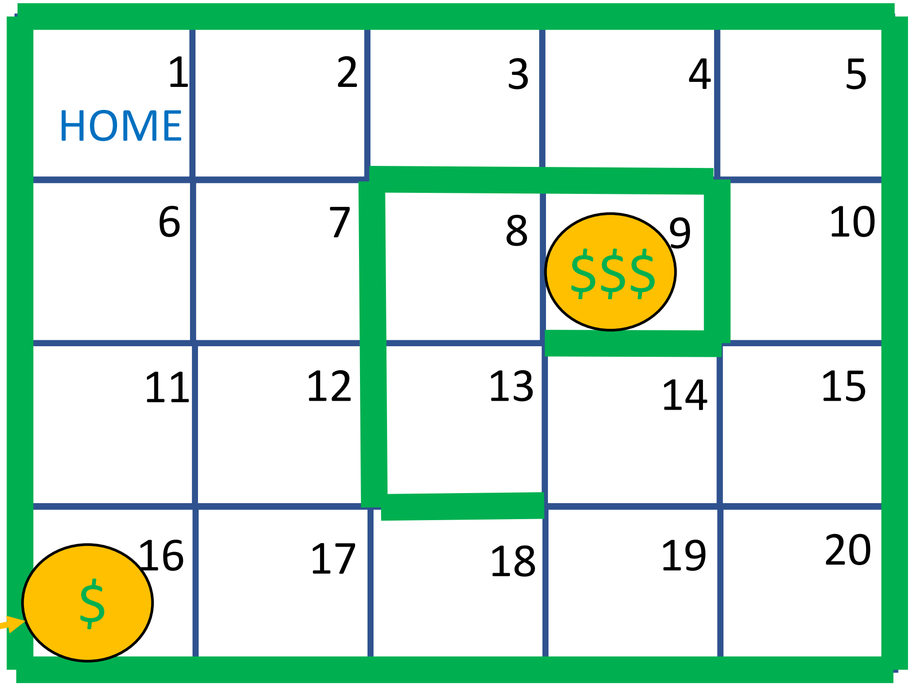
Rewards appear iid Bern(1/2), then:

$\boxed{R_9(t) \sim U[0,20]}$

$R_{16}(t) \sim U[0,4]$

$R_i(t) \sim U[0,1]$

Compare to Fine-Tuned Heuristic



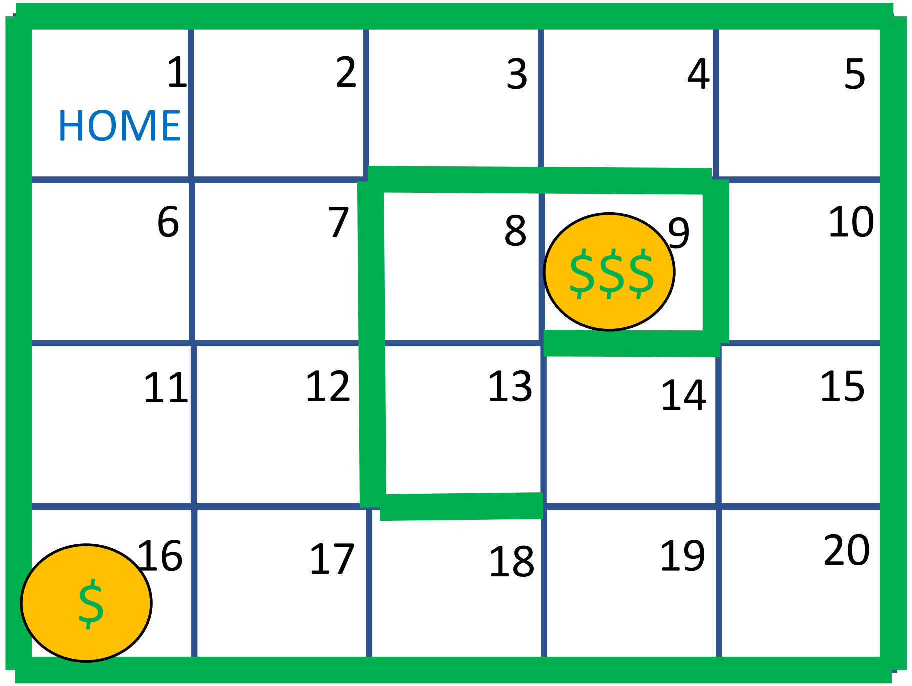
Rewards appear iid Bern(1/2), then:

$R_9(t) \sim U[0,20]$

$\boxed{R_{16}(t) \sim U[0,4]}$

$R_i(t) \sim U[0,1]$

Compare to Fine-Tuned Heuristic



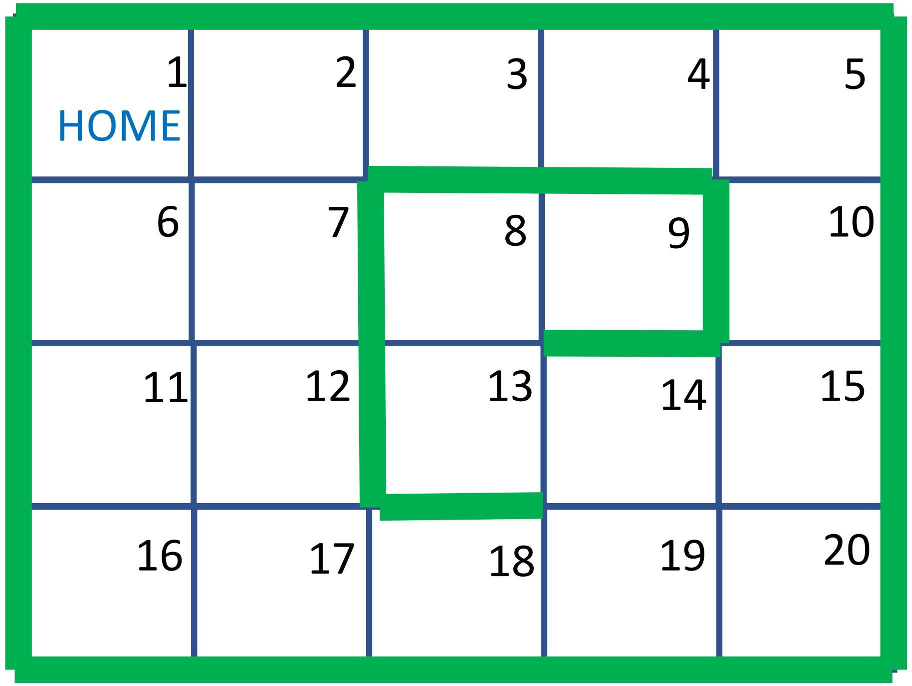
Rewards appear iid Bern(1/2), then:

$R_9(t) \sim U[0,20]$

$R_{16}(t) \sim U[0,4]$

$\boxed{R_i(t) \sim U[0,1]}$

Compare to Fine-Tuned Heuristic



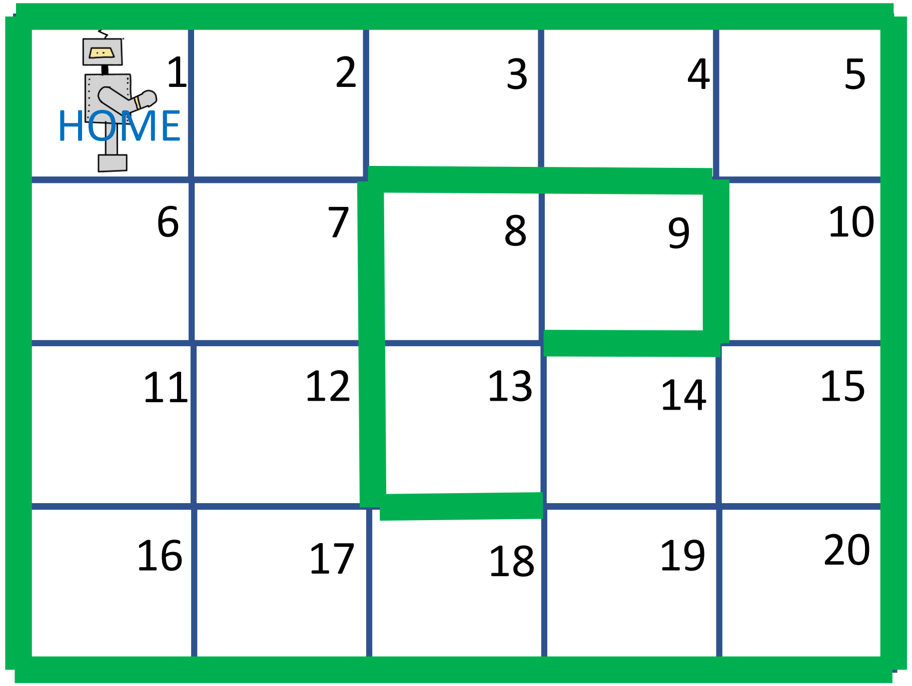
Rewards appear iid Bern(1/2), then:

$R_9(t) \sim U[0,20]$

$R_{16}(t) \sim U[0,4]$

$R_i(t) \sim U[0,1]$

Compare to Fine-Tuned Heuristic



Rewards appear iid Bern(1/2), then:

$R_9(t) \sim U[0,20]$

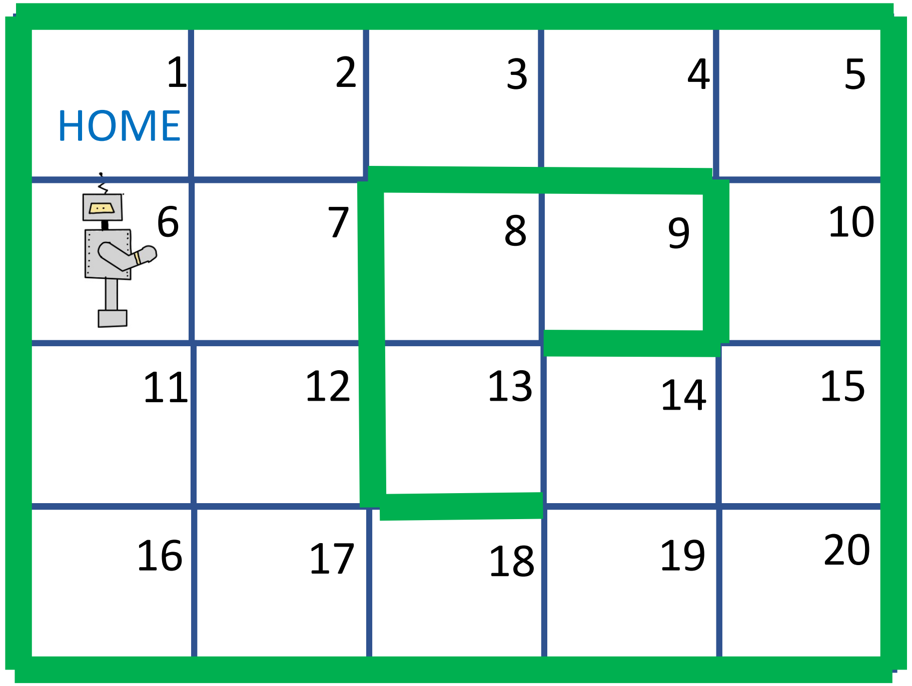
$R_{16}(t) \sim U[0,4]$

$R_i(t) \sim U[0,1]$

<u>Threshold-based heuristic</u>:

1. Robot moves from home to 16 by shortest path (ignoring all rewards it sees along the way).

Compare to Fine-Tuned Heuristic



Rewards appear iid Bern(1/2), then:

$R_9(t) \sim U[0,20]$

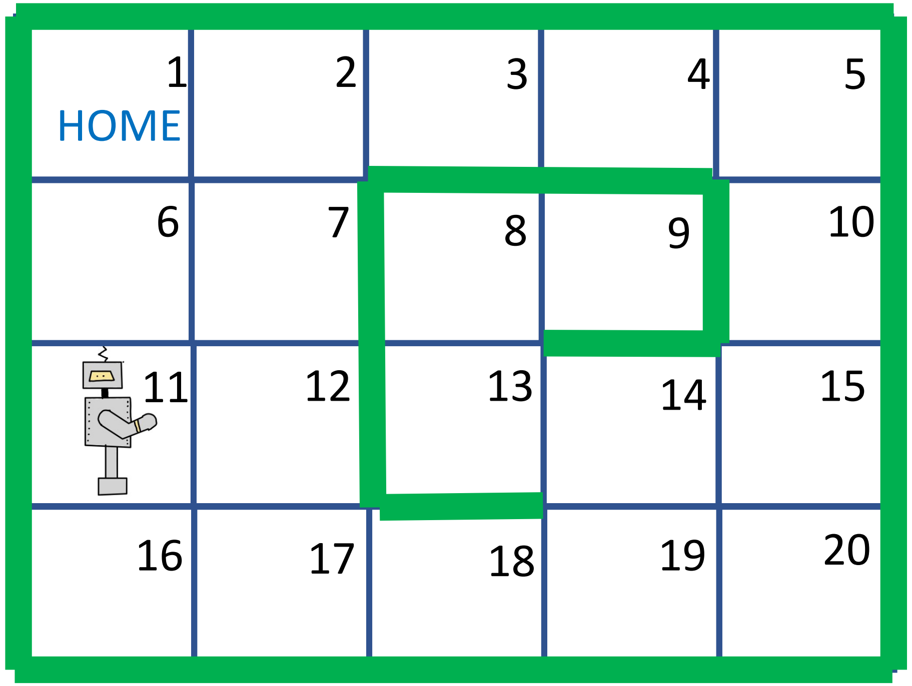
$R_{16}(t) \sim U[0,4]$

$R_i(t) \sim U[0,1]$

<u>Threshold-based heuristic</u>:

1. Robot moves from home to 16 by shortest path (ignoring all rewards it sees along the way).

Compare to Fine-Tuned Heuristic



Rewards appear iid Bern(1/2), then:

$R_9(t) \sim U[0,20]$

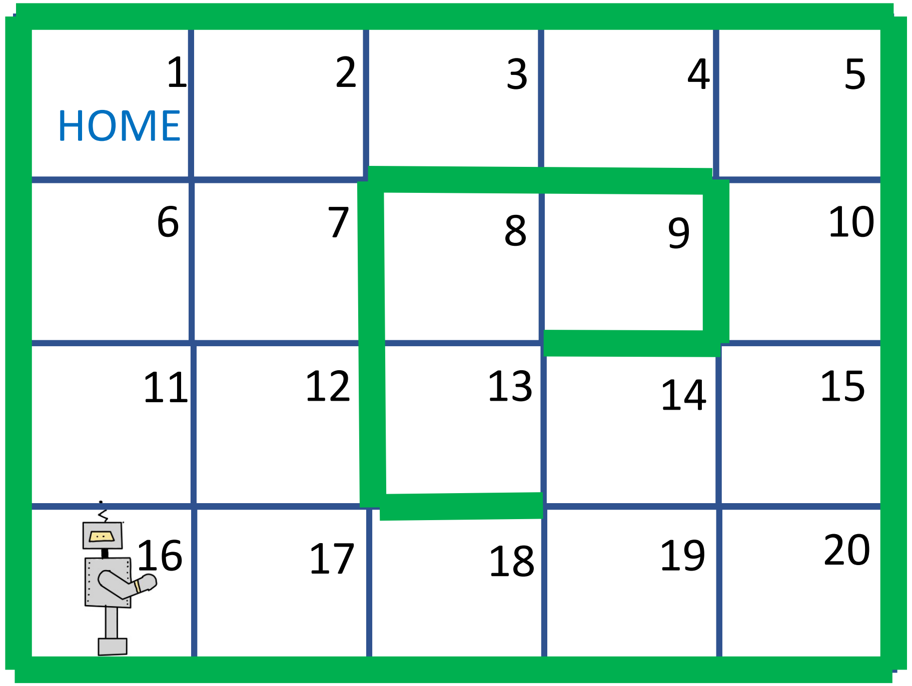
$R_{16}(t) \sim U[0,4]$

$R_i(t) \sim U[0,1]$

Threshold-based heuristic:

1. Robot moves from home to 16 by shortest path (ignoring all rewards it sees along the way).

Compare to Fine-Tuned Heuristic



Rewards appear iid Bern(1/2), then:

$R_9(t) \sim U[0,20]$

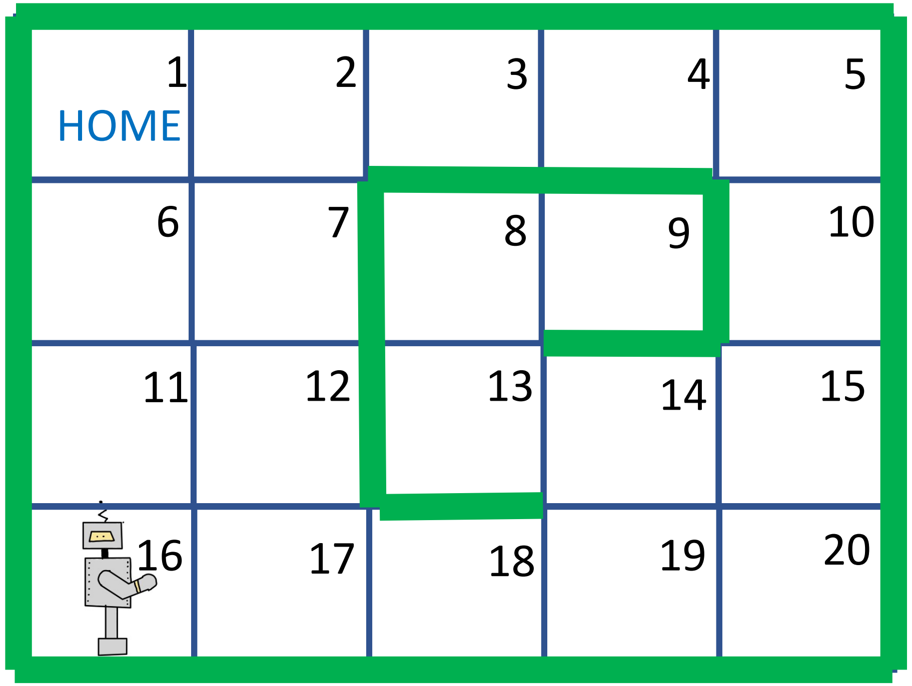
$R_{16}(t) \sim U[0,4]$

$R_i(t) \sim U[0,1]$

<u>Threshold-based heuristic</u>:

1. Robot moves from home to 16 by shortest path (ignoring all rewards it sees along the way).

Compare to Fine-Tuned Heuristic



Rewards appear iid Bern(1/2), then:

$R_9(t) \sim U[0,20]$

$R_{16}(t) \sim U[0,4]$

$R_i(t) \sim U[0,1]$

<u>Threshold-based heuristic</u>:

1. Robot moves from home to 16 by shortest path (ignoring all rewards it sees along the way).
2. Stay in 16 for ***only 1 slot***:

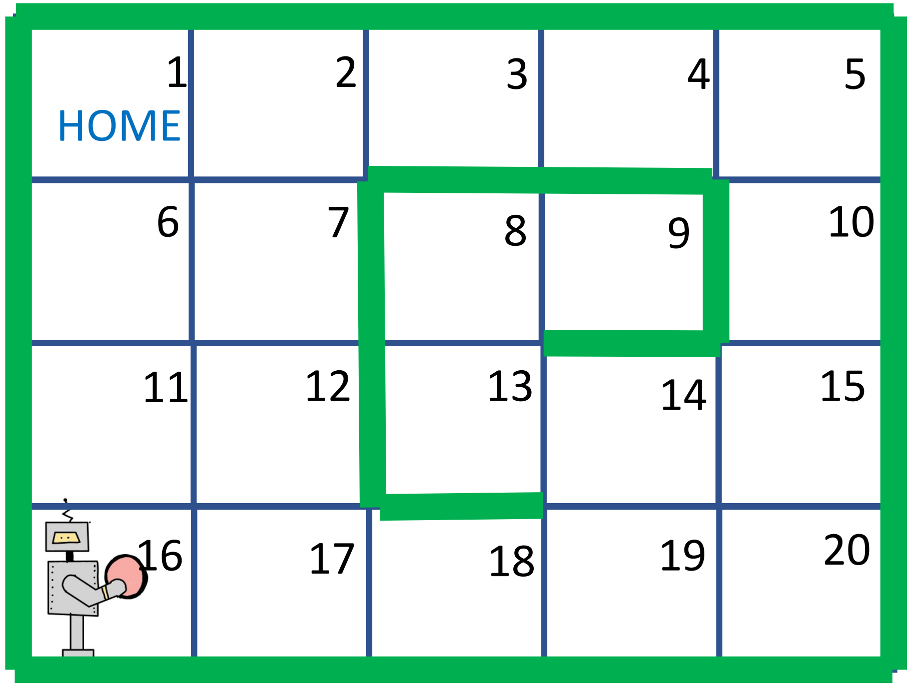
   If reward there with value > θ_1:

   Pick up. Return home by shortest path.

   Else:

   Continue to 9 via shortest path.

Compare to Fine-Tuned Heuristic



Rewards appear iid Bern(1/2), then:

$R_9(t) \sim U[0,20]$

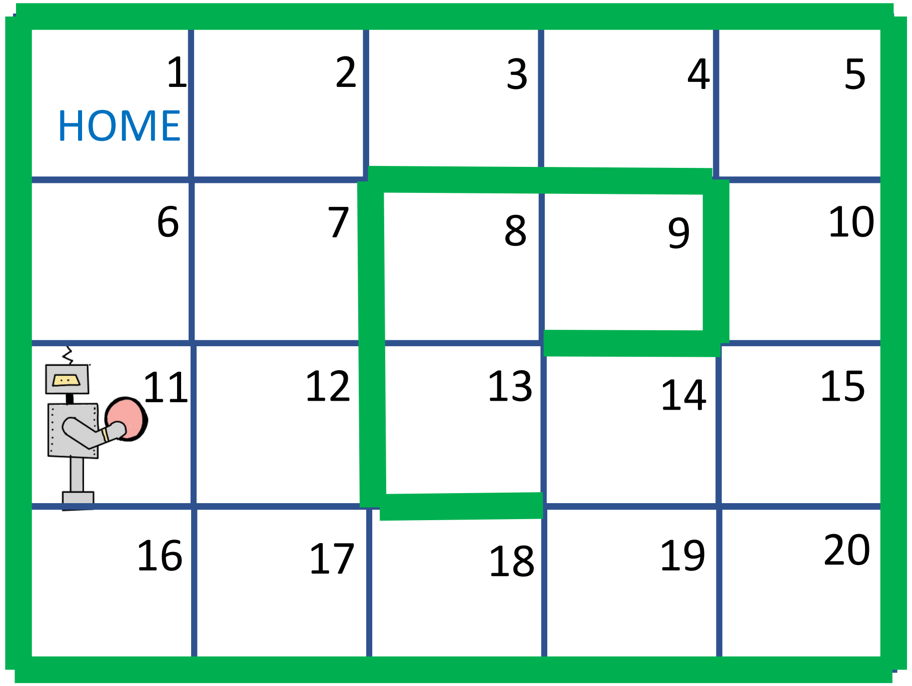
$R_{16}(t) \sim U[0,4]$

$R_i(t) \sim U[0,1]$

<u>Threshold-based heuristic</u>:

1. Robot moves from home to 16 by shortest path (ignoring all rewards it sees along the way).
2. Stay in 16 for ***only 1 slot***:
   If reward there with value $> \theta_1$:
   Pick up. Return home by shortest path.
   Else:
   Continue to 9 via shortest path.

Compare to Fine-Tuned Heuristic



Rewards appear iid Bern(1/2), then:

$R_9(t) \sim U[0,20]$

$R_{16}(t) \sim U[0,4]$

$R_i(t) \sim U[0,1]$

<u>Threshold-based heuristic</u>:

1. Robot moves from home to 16 by shortest path (ignoring all rewards it sees along the way).
2. Stay in 16 for ***only 1 slot***:

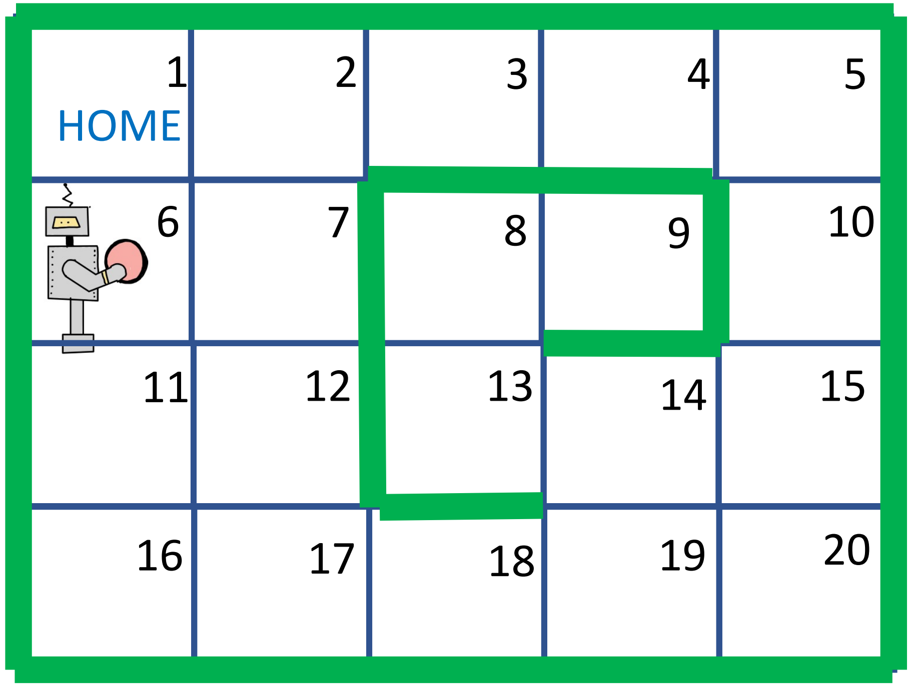
   If reward there with value > θ_1:

   Pick up. Return home by shortest path.

   Else:

   Continue to 9 via shortest path.

Compare to Fine-Tuned Heuristic



Rewards appear iid Bern(1/2), then:

$R_9(t) \sim U[0,20]$

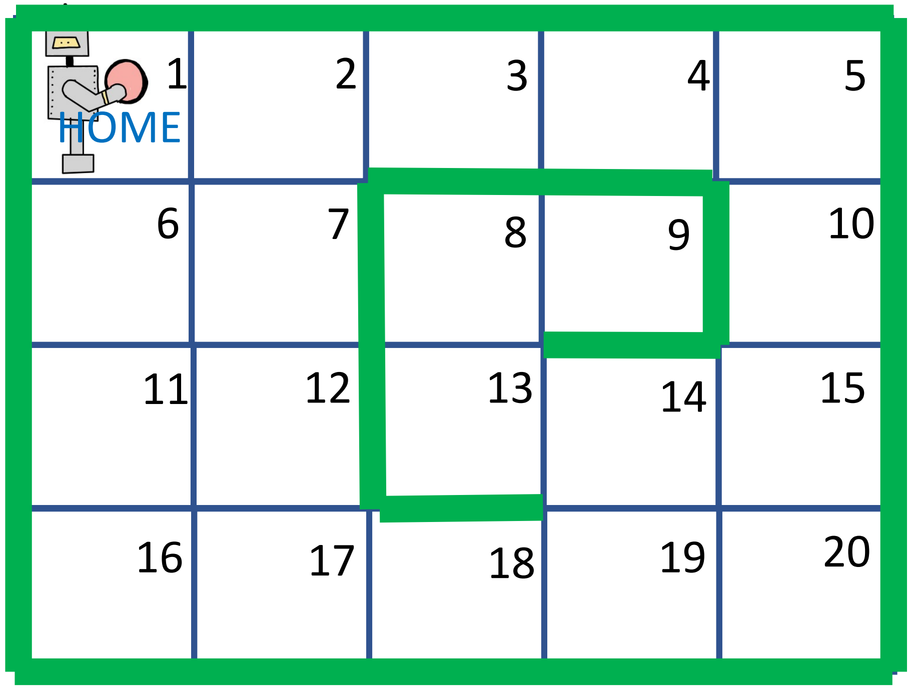
$R_{16}(t) \sim U[0,4]$

$R_i(t) \sim U[0,1]$

Threshold-based heuristic:

1. Robot moves from home to 16 by shortest path (ignoring all rewards it sees along the way).
2. Stay in 16 for ***only 1 slot***:
   If reward there with value > θ_1:
      Pick up. Return home by shortest path.
   Else:
      Continue to 9 via shortest path.

Compare to Fine-Tuned Heuristic



Rewards appear iid Bern(1/2), then:

$R_9(t) \sim U[0,20]$

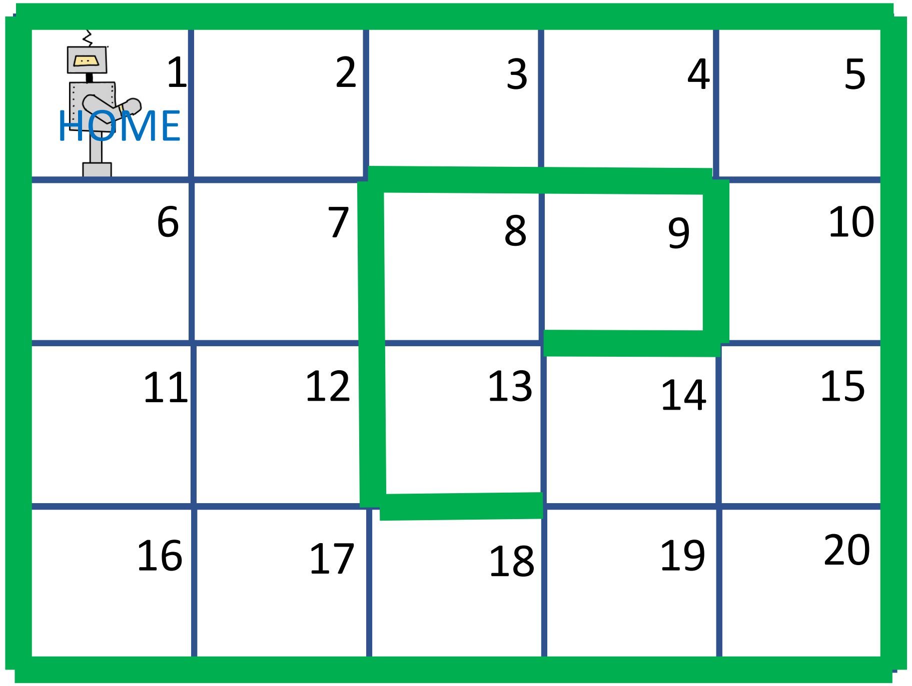
$R_{16}(t) \sim U[0,4]$

$R_i(t) \sim U[0,1]$

<u>Threshold-based heuristic</u>:

1. Robot moves from home to 16 by shortest path (ignoring all rewards it sees along the way).
2. Stay in 16 for ***only 1 slot***:
   If reward there with value > θ_1:
      Pick up. Return home by shortest path.
   Else:
      Continue to 9 via shortest path.

Compare to Fine-Tuned Heuristic



Rewards appear iid Bern(1/2), then:

$R_9(t) \sim U[0,20]$

$R_{16}(t) \sim U[0,4]$

$R_i(t) \sim U[0,1]$

<u>Threshold-based heuristic</u>:

1. Robot moves from home to 16 by shortest path (ignoring all rewards it sees along the way).
2. Stay in 16 for ***only 1 slot***:

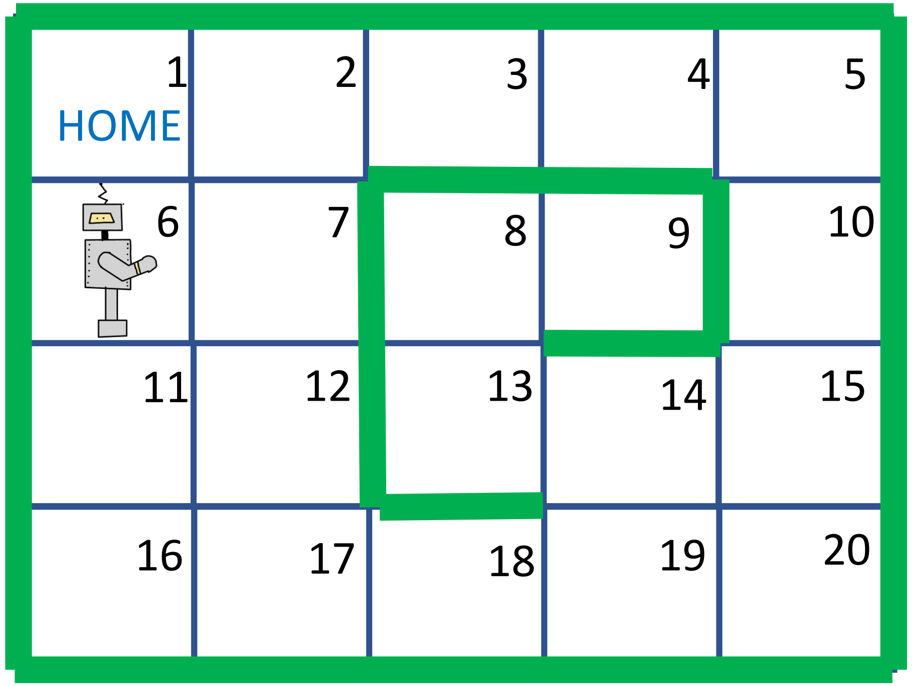
   If reward there with value > θ_1:

   Pick up. Return home by shortest path.

   Else:

   Continue to 9 via shortest path.

Compare to Fine-Tuned Heuristic



Rewards appear iid Bern(1/2), then:

$R_9(t) \sim U[0,20]$

$R_{16}(t) \sim U[0,4]$

$R_i(t) \sim U[0,1]$

<u>Threshold-based heuristic</u>:

1. Robot moves from home to 16 by shortest path (ignoring all rewards it sees along the way).
2. Stay in 16 for ***only 1 slot***:

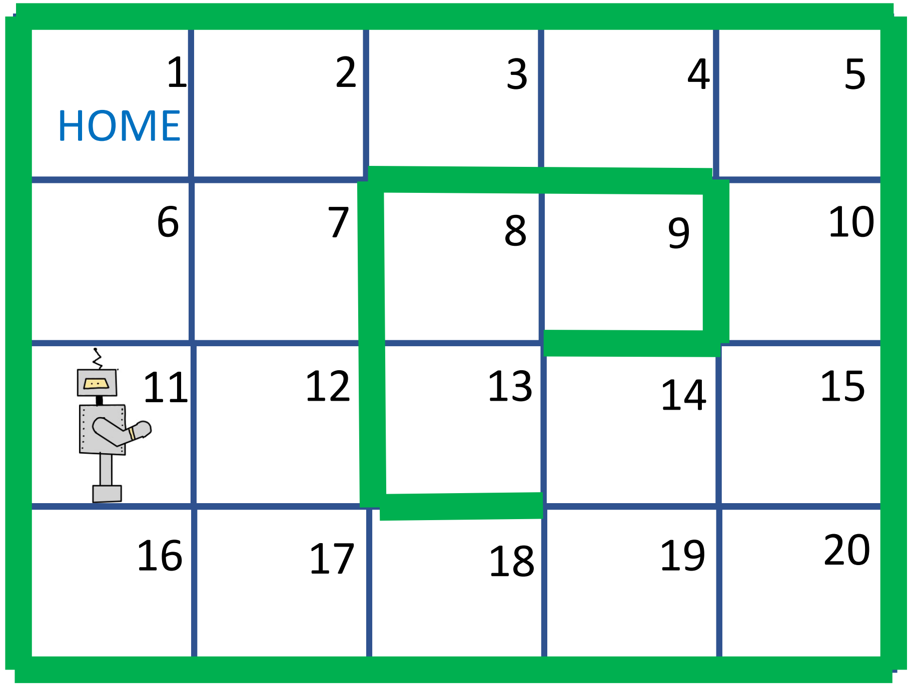
   If reward there with value $> \theta_1$:

   Pick up. Return home by shortest path.

   Else:

   Continue to 9 via shortest path.

Compare to Fine-Tuned Heuristic



Rewards appear iid Bern(1/2), then:

$R_9(t) \sim U[0,20]$

$R_{16}(t) \sim U[0,4]$

$R_i(t) \sim U[0,1]$

<u>Threshold-based heuristic</u>:

1. Robot moves from home to 16 by shortest path (ignoring all rewards it sees along the way).
2. Stay in 16 for ***only 1 slot***:

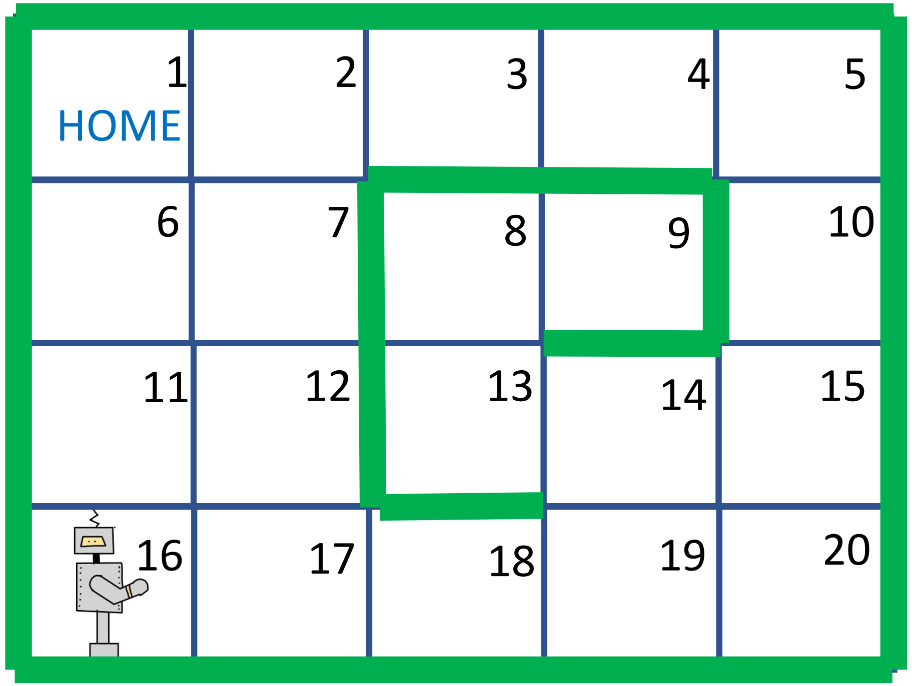
   If reward there with value $> \theta_1$:

   Pick up. Return home by shortest path.

   Else:

   Continue to 9 via shortest path.

Compare to Fine-Tuned Heuristic



Rewards appear iid Bern(1/2), then:

$R_9(t) \sim U[0,20]$

$R_{16}(t) \sim U[0,4]$

$R_i(t) \sim U[0,1]$

Threshold-based heuristic:

1. Robot moves from home to 16 by shortest path (ignoring all rewards it sees along the way).
2. Stay in 16 for ***only 1 slot***:

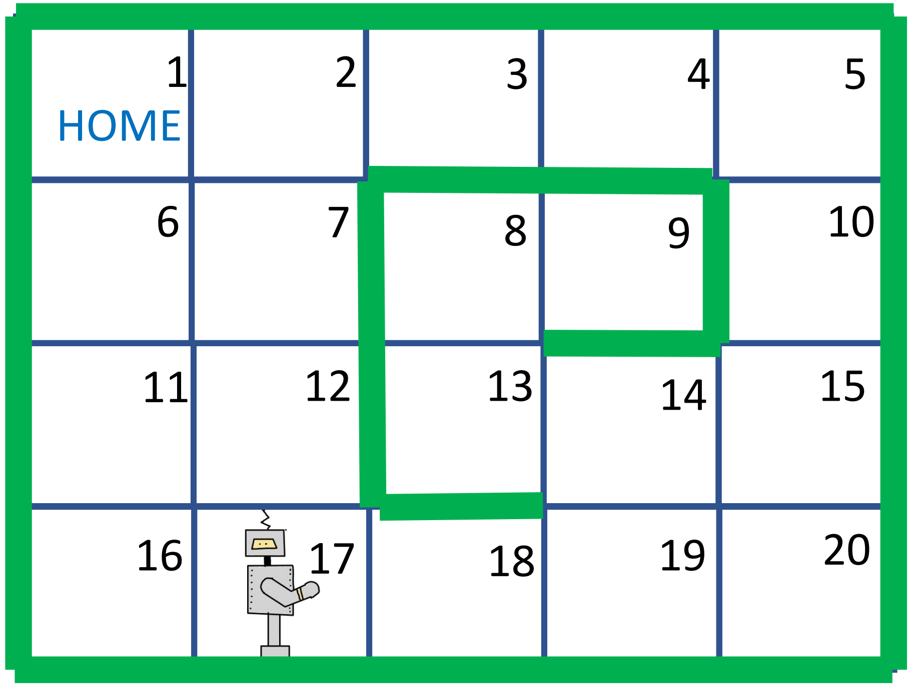
   If reward there with value $> \theta_1$:

   Pick up. Return home by shortest path.

   Else:

   Continue to 9 via shortest path.

Compare to Fine-Tuned Heuristic



Rewards appear iid Bern(1/2), then:

$R_9(t) \sim U[0,20]$

$R_{16}(t) \sim U[0,4]$

$R_i(t) \sim U[0,1]$

<u>Threshold-based heuristic</u>:

1. Robot moves from home to 16 by shortest path (ignoring all rewards it sees along the way).
2. Stay in 16 for ***only 1 slot***:

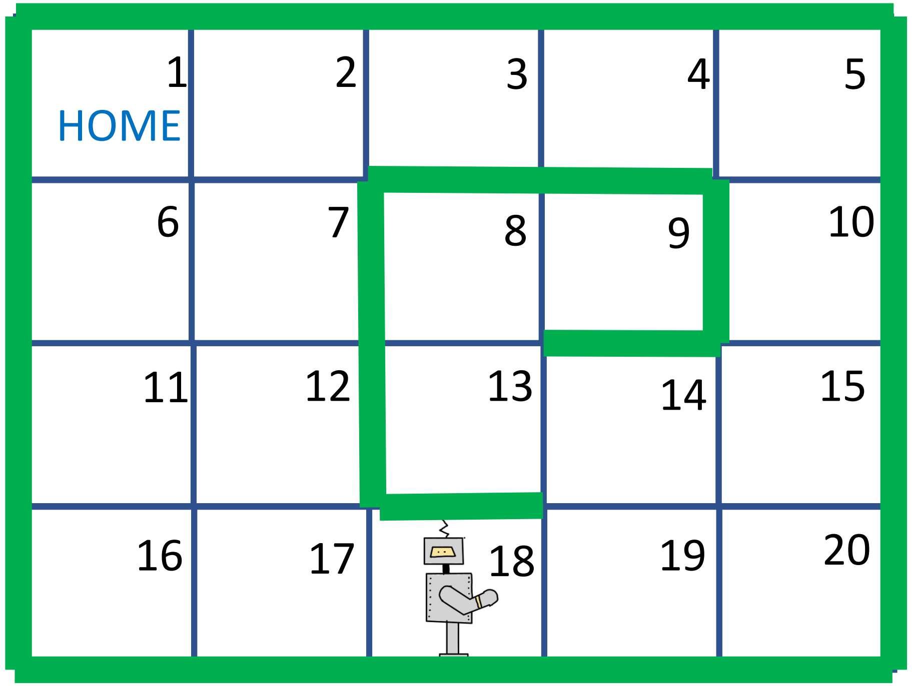
   If reward there with value > θ_1:

   Pick up. Return home by shortest path.

   Else:

   Continue to 9 via shortest path.

Compare to Fine-Tuned Heuristic



Rewards appear iid Bern(1/2), then:

$R_9(t) \sim U[0,20]$

$R_{16}(t) \sim U[0,4]$

$R_i(t) \sim U[0,1]$

<u>Threshold-based heuristic</u>:

1. Robot moves from home to 16 by shortest path (ignoring all rewards it sees along the way).
2. Stay in 16 for ***only 1 slot***:

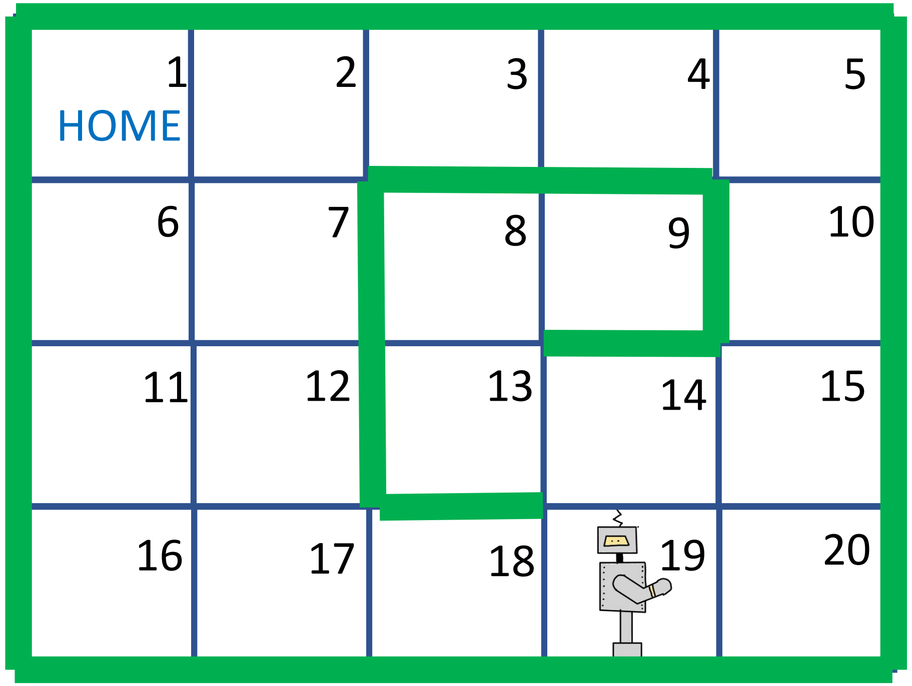
   If reward there with value $> \theta_1$:

   Pick up. Return home by shortest path.

   Else:

   Continue to 9 via shortest path.

Compare to Fine-Tuned Heuristic



Rewards appear iid Bern(1/2), then:

$R_9(t) \sim U[0,20]$

$R_{16}(t) \sim U[0,4]$

$R_i(t) \sim U[0,1]$

<u>Threshold-based heuristic</u>:

1. Robot moves from home to 16 by shortest path (ignoring all rewards it sees along the way).
2. Stay in 16 for ***only 1 slot***:

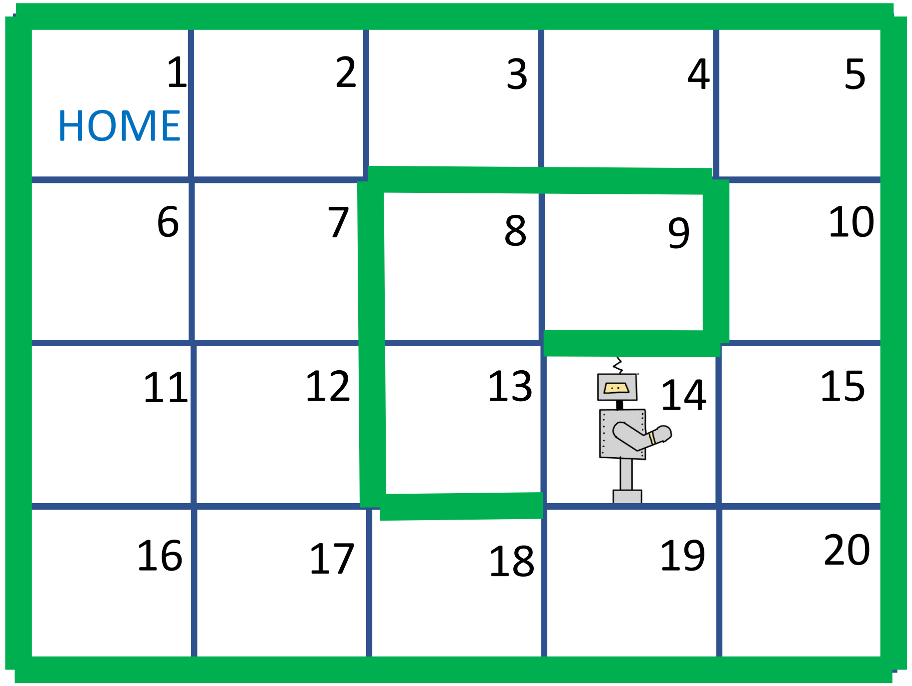
   If reward there with value $> \theta_1$:

   Pick up. Return home by shortest path.

   Else:

   Continue to 9 via shortest path.

Compare to Fine-Tuned Heuristic



Rewards appear iid Bern(1/2), then:

$R_9(t) \sim U[0,20]$

$R_{16}(t) \sim U[0,4]$

$R_i(t) \sim U[0,1]$

Threshold-based heuristic:

1. Robot moves from home to 16 by shortest path (ignoring all rewards it sees along the way).
2. Stay in 16 for ***only 1 slot***:

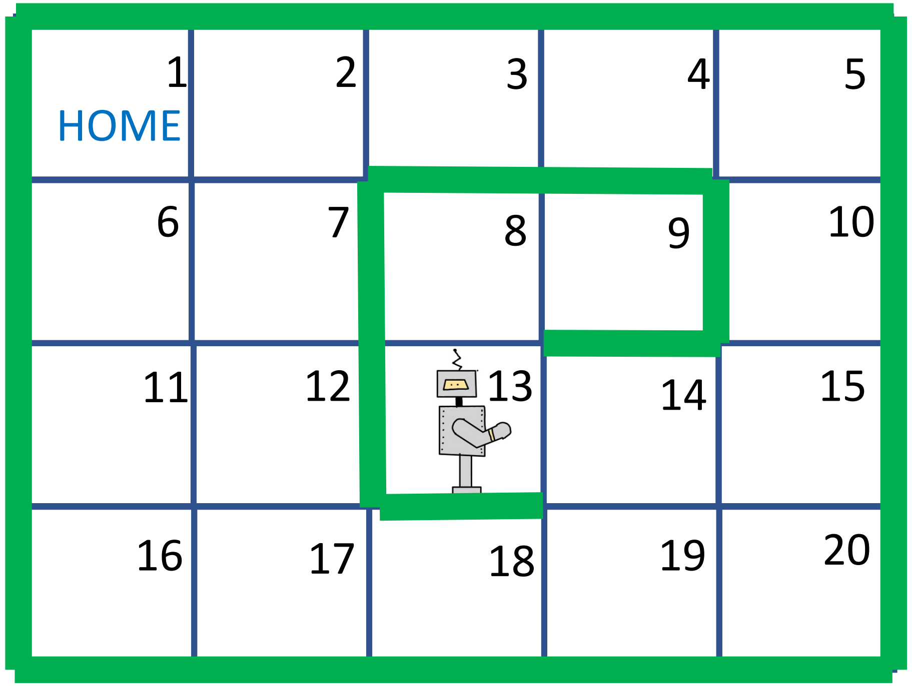
   If reward there with value $> \theta_1$:

   Pick up. Return home by shortest path.

   Else:

   Continue to 9 via shortest path.

Compare to Fine-Tuned Heuristic



Rewards appear iid Bern(1/2), then:

$R_9(t) \sim U[0,20]$

$R_{16}(t) \sim U[0,4]$

$R_i(t) \sim U[0,1]$

Threshold-based heuristic:

1. Robot moves from home to 16 by shortest path (ignoring all rewards it sees along the way).
2. Stay in 16 for ***only 1 slot***:

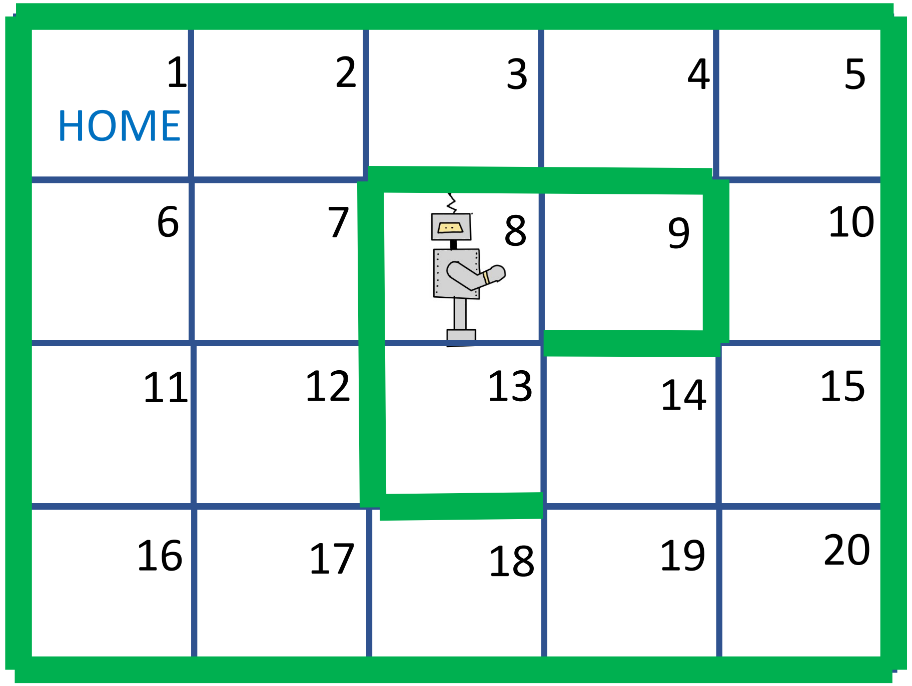
   If reward there with value $> \theta_1$:

   Pick up. Return home by shortest path.

   Else:

   Continue to 9 via shortest path.

Compare to Fine-Tuned Heuristic



Rewards appear iid Bern(1/2), then:

$R_9(t) \sim U[0,20]$

$R_{16}(t) \sim U[0,4]$

$R_i(t) \sim U[0,1]$

<u>Threshold-based heuristic</u>:

1. Robot moves from home to 16 by shortest path (ignoring all rewards it sees along the way).
2. Stay in 16 for ***only 1 slot***:

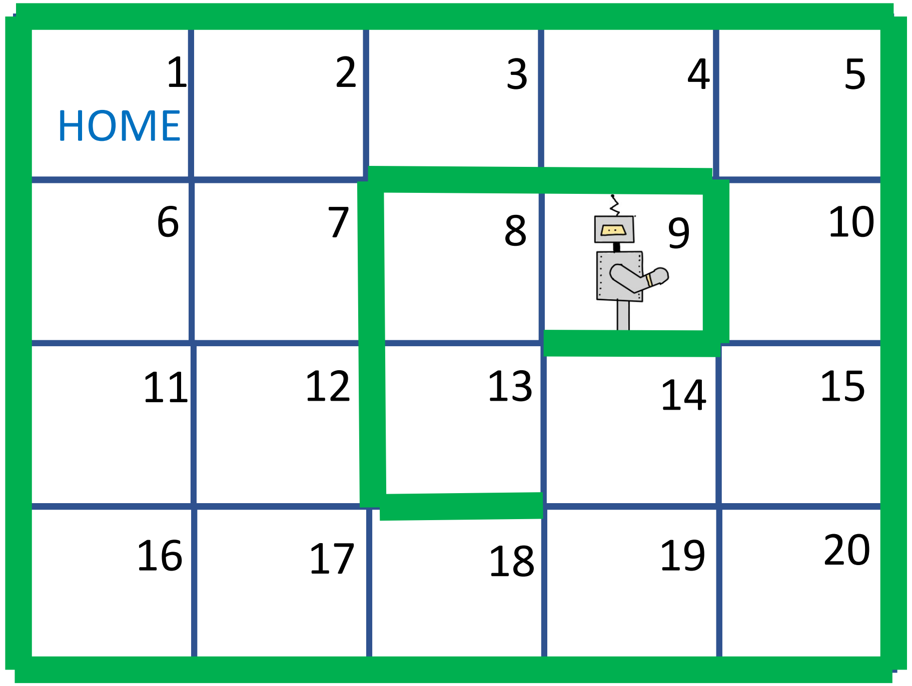
   If reward there with value $> \theta_1$:

   Pick up. Return home by shortest path.

   Else:

   Continue to 9 via shortest path.

Compare to Fine-Tuned Heuristic



Rewards appear iid Bern(1/2), then:

$R_9(t) \sim U[0,20]$

$R_{16}(t) \sim U[0,4]$

$R_i(t) \sim U[0,1]$

<u>Threshold-based heuristic</u>:

1. Robot moves from home to 16 by shortest path (ignoring all rewards it sees along the way).
2. Stay in 16 for ***only 1 slot***:

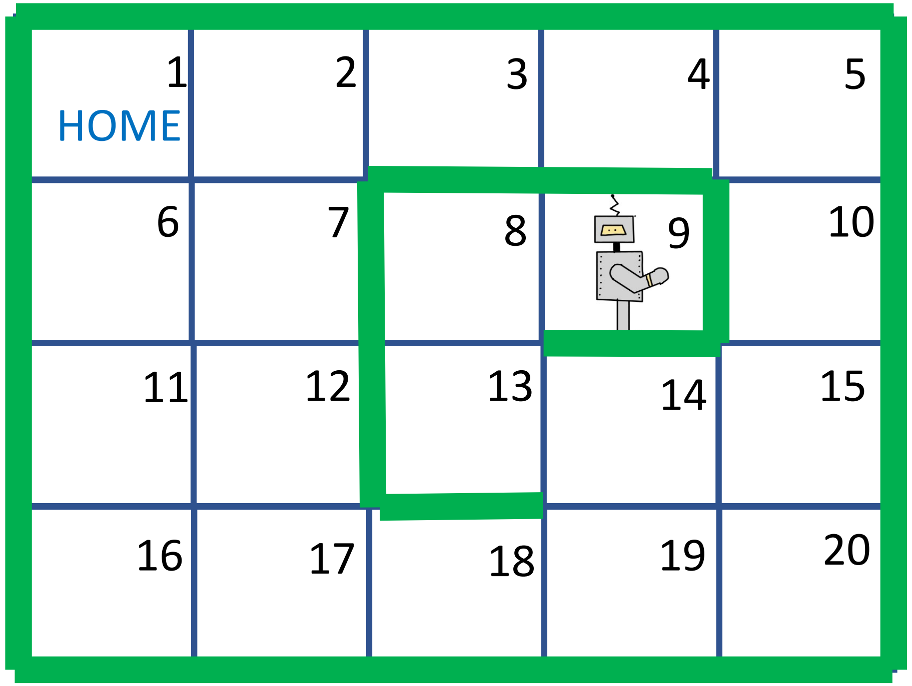
   If reward there with value > θ_1:

   Pick up. Return home by shortest path.

   Else:

   Continue to 9 via shortest path.

Compare to Fine-Tuned Heuristic



Rewards appear iid Bern(1/2), then:

$R_9(t) \sim U[0,20]$

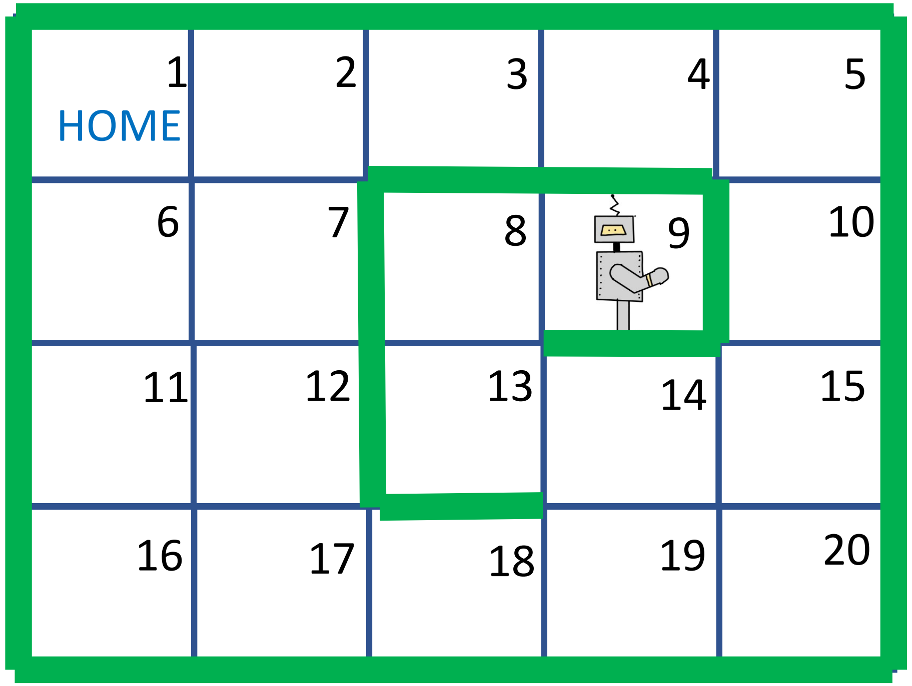
$R_{16}(t) \sim U[0,4]$

$R_i(t) \sim U[0,1]$

<u>Threshold-based heuristic</u>:

1. Robot moves from home to 16 by shortest path (ignoring all rewards it sees along the way).
2. Stay in 16 for ***only 1 slot***:
   If reward there with value > θ_1:
   Pick up. Return home by shortest path.
   Else:
   Continue to 9 via shortest path
3. Wait in 9 until reward with value> θ_2

Compare to Fine-Tuned Heuristic



Rewards appear iid Bern(1/2), then:

$R_9(t) \sim U[0,20]$

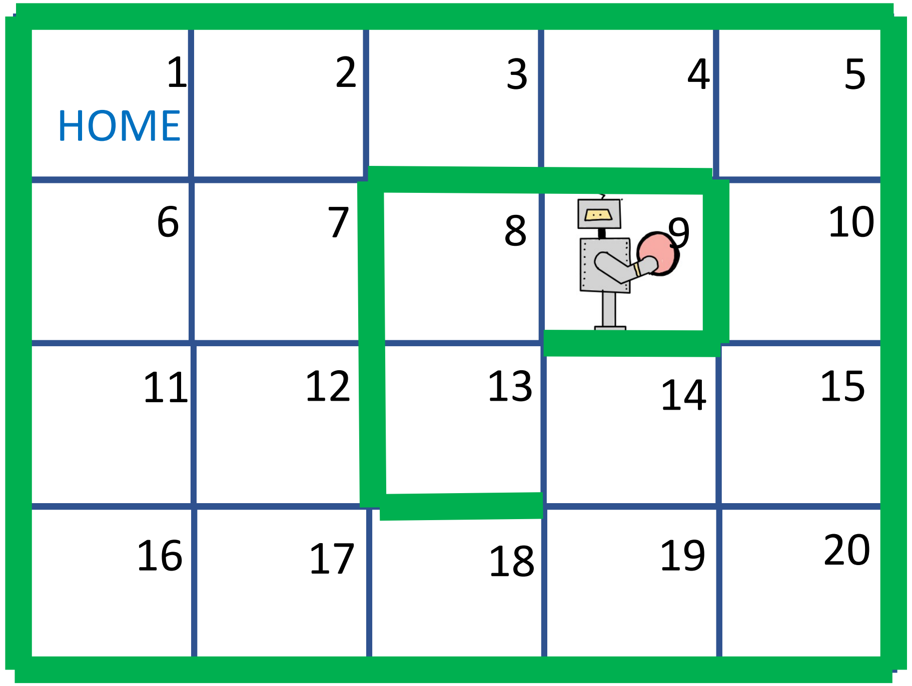
$R_{16}(t) \sim U[0,4]$

$R_i(t) \sim U[0,1]$

<u>Threshold-based heuristic</u>:

1. Robot moves from home to 16 by shortest path (ignoring all rewards it sees along the way).
2. Stay in 16 for ***only 1 slot***:
   If reward there with value $> \theta_1$:
   Pick up. Return home by shortest path.
   Else:
   Continue to 9 via shortest path
3. Wait in 9 until reward with value $> \theta_2$
4. Pick up. Return home by any shortest path.

Compare to Fine-Tuned Heuristic



Rewards appear iid Bern(1/2), then:

$R_9(t) \sim U[0,20]$

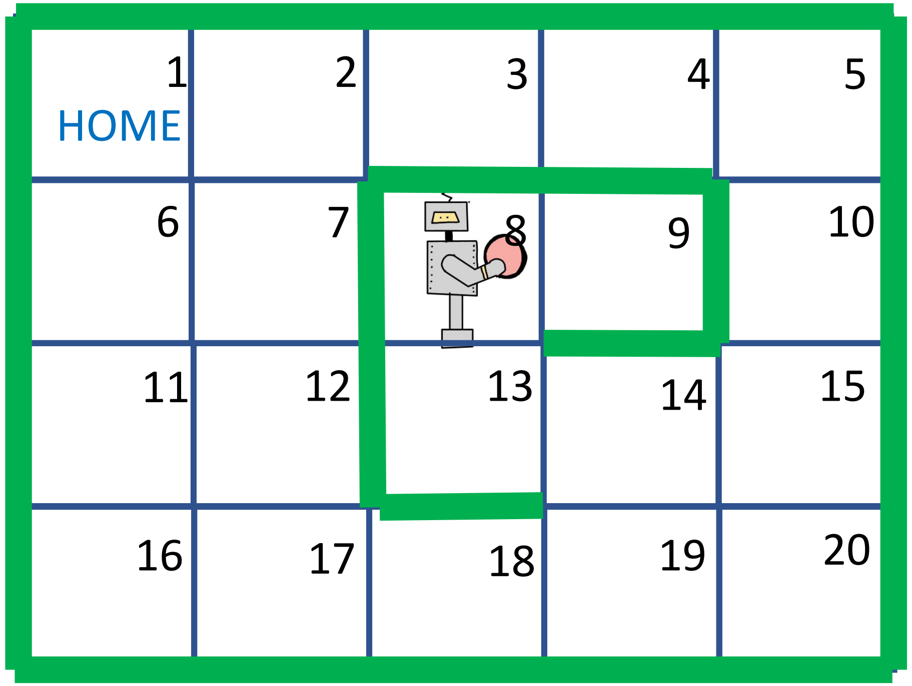
$R_{16}(t) \sim U[0,4]$

$R_i(t) \sim U[0,1]$

<u>Threshold-based heuristic</u>:

1. Robot moves from home to 16 by shortest path (ignoring all rewards it sees along the way).
2. Stay in 16 for ***only 1 slot***:
   If reward there with value $> \theta_1$:
      Pick up. Return home by shortest path.
   Else:
      Continue to 9 via shortest path
3. Wait in 9 until reward with value$> \theta_2$
4. Pick up. Return home by any shortest path.

Compare to Fine-Tuned Heuristic



Rewards appear iid Bern(1/2), then:

$R_9(t) \sim U[0,20]$

$R_{16}(t) \sim U[0,4]$

$R_i(t) \sim U[0,1]$

<u>Threshold-based heuristic</u>:

1. Robot moves from home to 16 by shortest path (ignoring all rewards it sees along the way).
2. Stay in 16 for ***only 1 slot***:

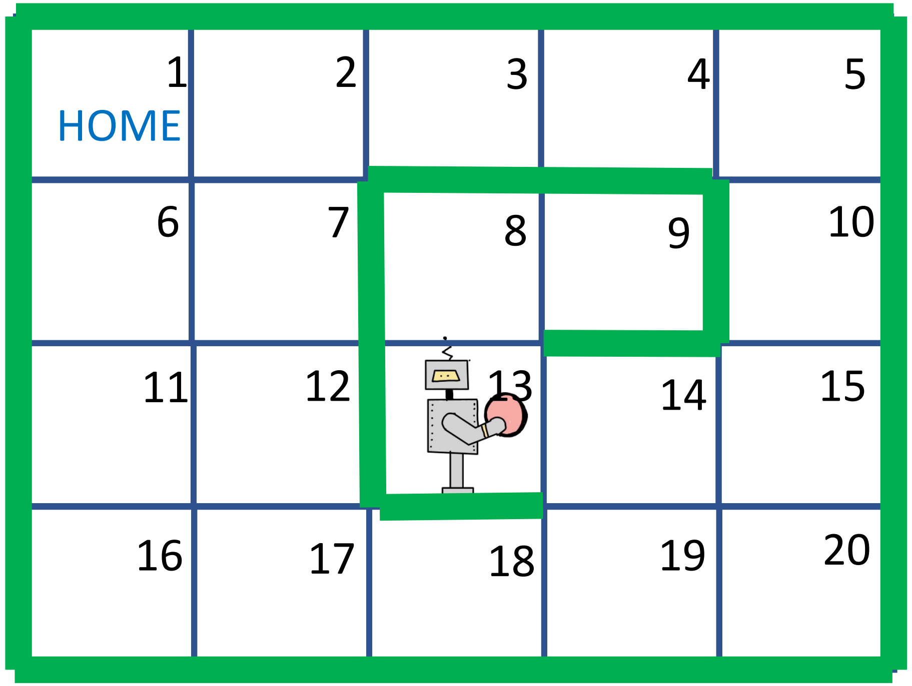
   If reward there with value > θ_1:

   Pick up. Return home by shortest path.

   Else:

   Continue to 9 via shortest path
3. Wait in 9 until reward with value> θ_2
4. Pick up. Return home by any shortest path.

Compare to Fine-Tuned Heuristic



Rewards appear iid Bern(1/2), then:

$R_9(t) \sim U[0,20]$

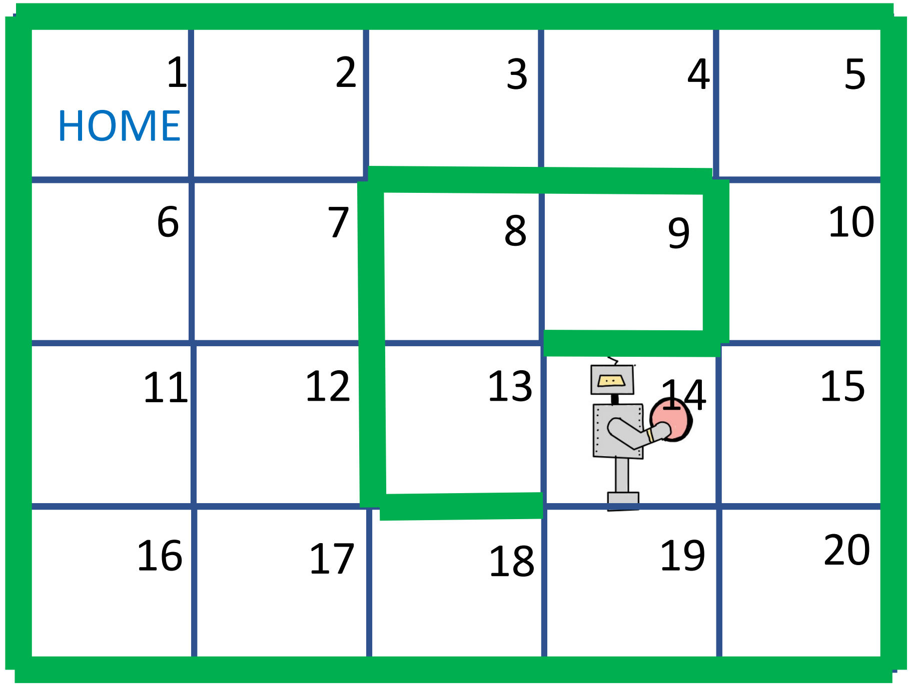
$R_{16}(t) \sim U[0,4]$

$R_i(t) \sim U[0,1]$

<u>Threshold-based heuristic</u>:

1. Robot moves from home to 16 by shortest path (ignoring all rewards it sees along the way).
2. Stay in 16 for ***only 1 slot***:
   If reward there with value > θ_1:
   Pick up. Return home by shortest path.
   Else:
   Continue to 9 via shortest path
3. Wait in 9 until reward with value> θ_2
4. Pick up. Return home by any shortest path.

Compare to Fine-Tuned Heuristic



Rewards appear iid Bern(1/2), then:

$R_9(t) \sim U[0,20]$

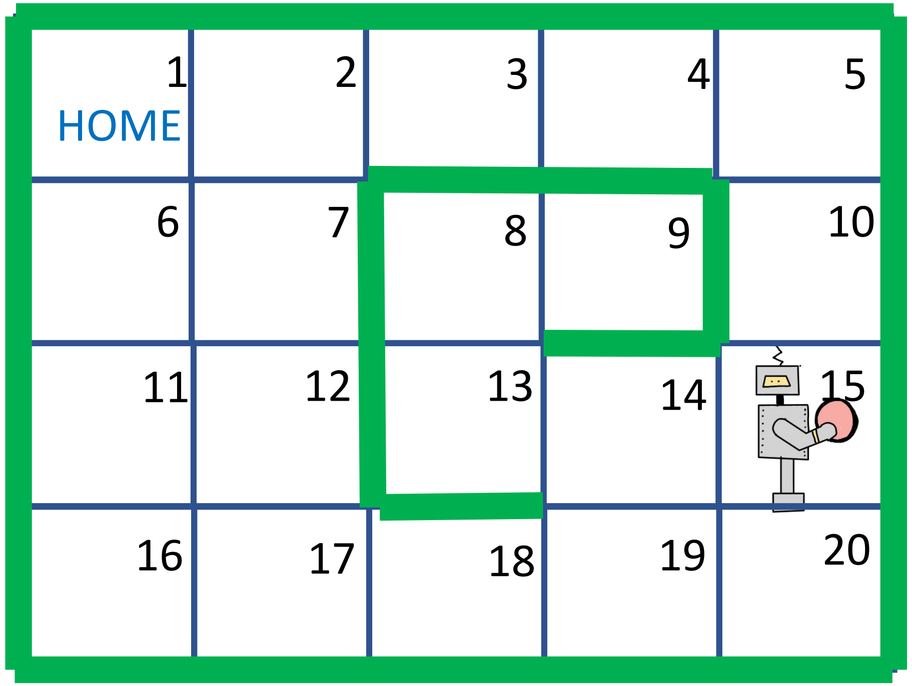
$R_{16}(t) \sim U[0,4]$

$R_i(t) \sim U[0,1]$

<u>Threshold-based heuristic</u>:

1. Robot moves from home to 16 by shortest path (ignoring all rewards it sees along the way).
2. Stay in 16 for ***only 1 slot***:
   If reward there with value $> \theta_1$:
   Pick up. Return home by shortest path.
   Else:
   Continue to 9 via shortest path
3. Wait in 9 until reward with value $> \theta_2$
4. Pick up. Return home by any shortest path.

Compare to Fine-Tuned Heuristic



Rewards appear iid Bern(1/2), then:

$R_9(t) \sim U[0,20]$

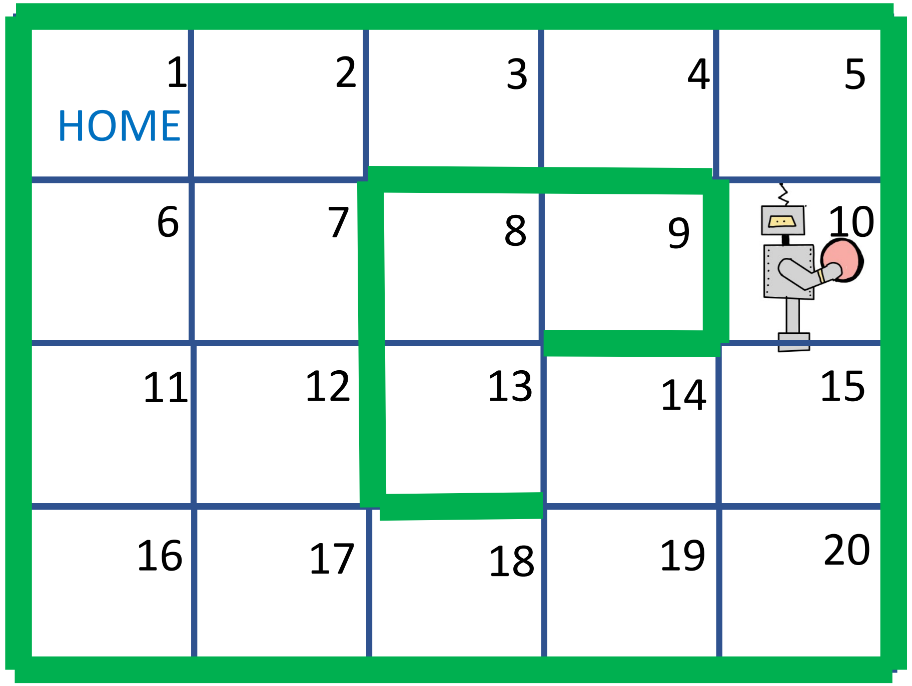
$R_{16}(t) \sim U[0,4]$

$R_i(t) \sim U[0,1]$

Threshold-based heuristic:

1. Robot moves from home to 16 by shortest path (ignoring all rewards it sees along the way).
2. Stay in 16 for ***only 1 slot***:
   If reward there with value $> \theta_1$:
      Pick up. Return home by shortest path.
   Else:
      Continue to 9 via shortest path
3. Wait in 9 until reward with value$> \theta_2$
4. Pick up. Return home by any shortest path.

Compare to Fine-Tuned Heuristic



Rewards appear iid Bern(1/2), then:

$R_9(t) \sim U[0,20]$

$R_{16}(t) \sim U[0,4]$

$R_i(t) \sim U[0,1]$

<u>Threshold-based heuristic</u>:

1. Robot moves from home to 16 by shortest path (ignoring all rewards it sees along the way).
2. Stay in 16 for ***only 1 slot***:

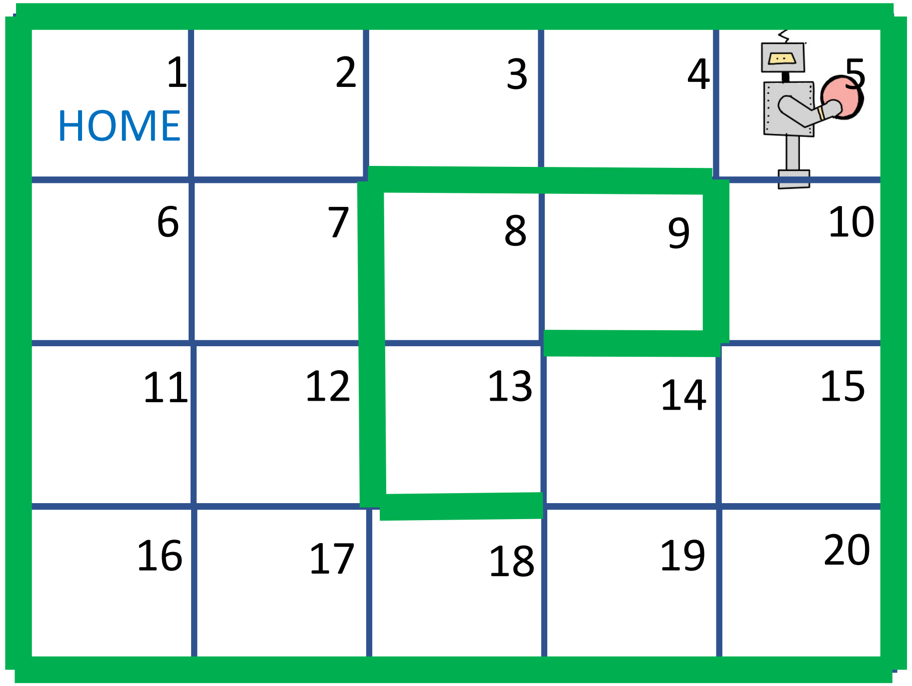
   If reward there with value $> \theta_1$:

   Pick up. Return home by shortest path.

   Else:

   Continue to 9 via shortest path
3. Wait in 9 until reward with value$> \theta_2$
4. Pick up. Return home by any shortest path.

Compare to Fine-Tuned Heuristic



Rewards appear iid Bern(1/2), then:

$R_9(t) \sim U[0,20]$

$R_{16}(t) \sim U[0,4]$

$R_i(t) \sim U[0,1]$

<u>Threshold-based heuristic</u>:

1. Robot moves from home to 16 by shortest path (ignoring all rewards it sees along the way).
2. Stay in 16 for ***only 1 slot***:

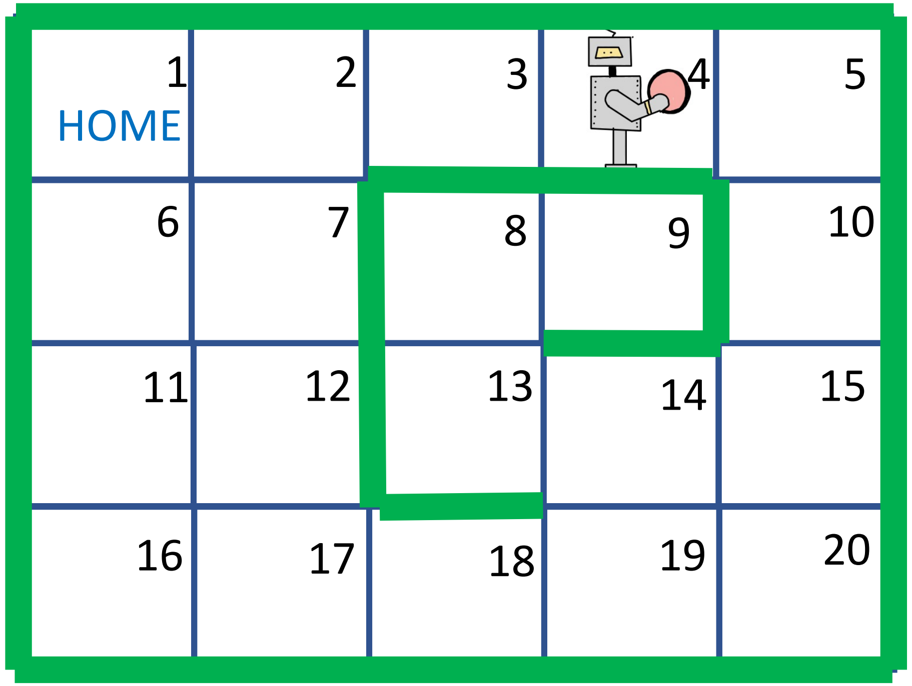
   If reward there with value $> \theta_1$:

   Pick up. Return home by shortest path.

   Else:

   Continue to 9 via shortest path
3. Wait in 9 until reward with value $> \theta_2$
4. Pick up. Return home by any shortest path.

Compare to Fine-Tuned Heuristic



Rewards appear iid Bern(1/2), then:

$R_9(t) \sim U[0,20]$

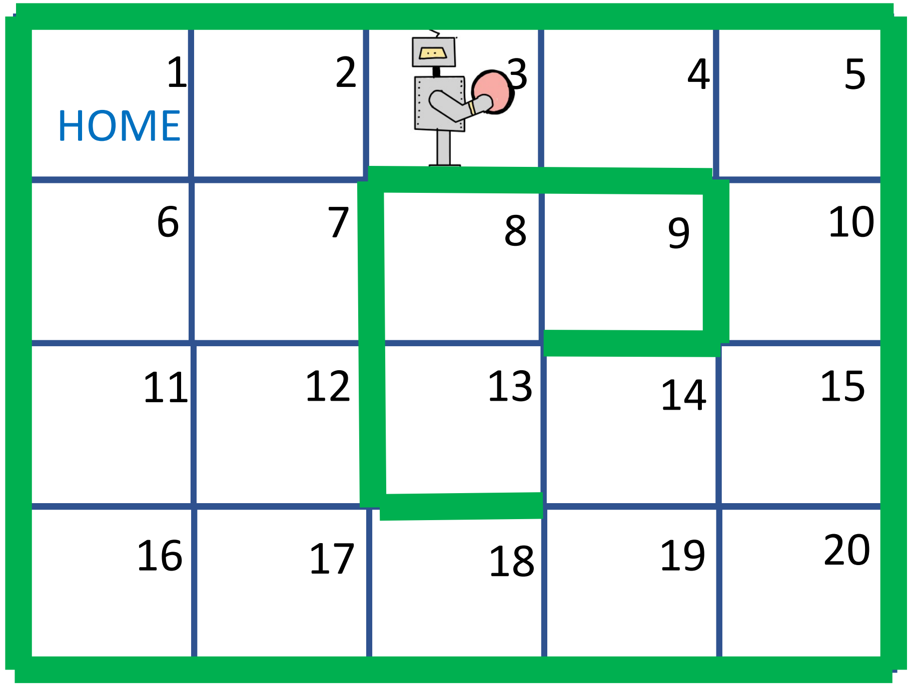
$R_{16}(t) \sim U[0,4]$

$R_i(t) \sim U[0,1]$

<u>Threshold-based heuristic</u>:

1. Robot moves from home to 16 by shortest path (ignoring all rewards it sees along the way).
2. Stay in 16 for ***only 1 slot***:
   If reward there with value > θ_1:
      Pick up. Return home by shortest path.
   Else:
      Continue to 9 via shortest path
3. Wait in 9 until reward with value> θ_2
4. Pick up. Return home by any shortest path.

Compare to Fine-Tuned Heuristic



Rewards appear iid Bern(1/2), then:

$R_9(t) \sim U[0,20]$

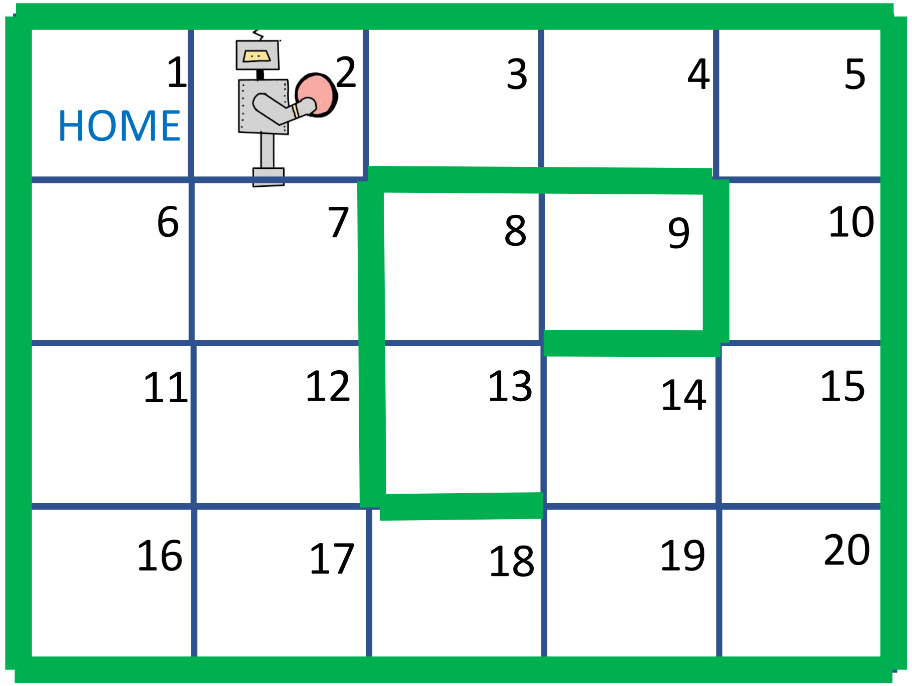
$R_{16}(t) \sim U[0,4]$

$R_i(t) \sim U[0,1]$

<u>Threshold-based heuristic</u>:

1. Robot moves from home to 16 by shortest path (ignoring all rewards it sees along the way).
2. Stay in 16 for ***only 1 slot***:
   If reward there with value $> \theta_1$:
      Pick up. Return home by shortest path.
   Else:
      Continue to 9 via shortest path
3. Wait in 9 until reward with value $> \theta_2$
4. Pick up. Return home by any shortest path.

Compare to Fine-Tuned Heuristic



Rewards appear iid Bern(1/2), then:

$R_9(t) \sim U[0,20]$

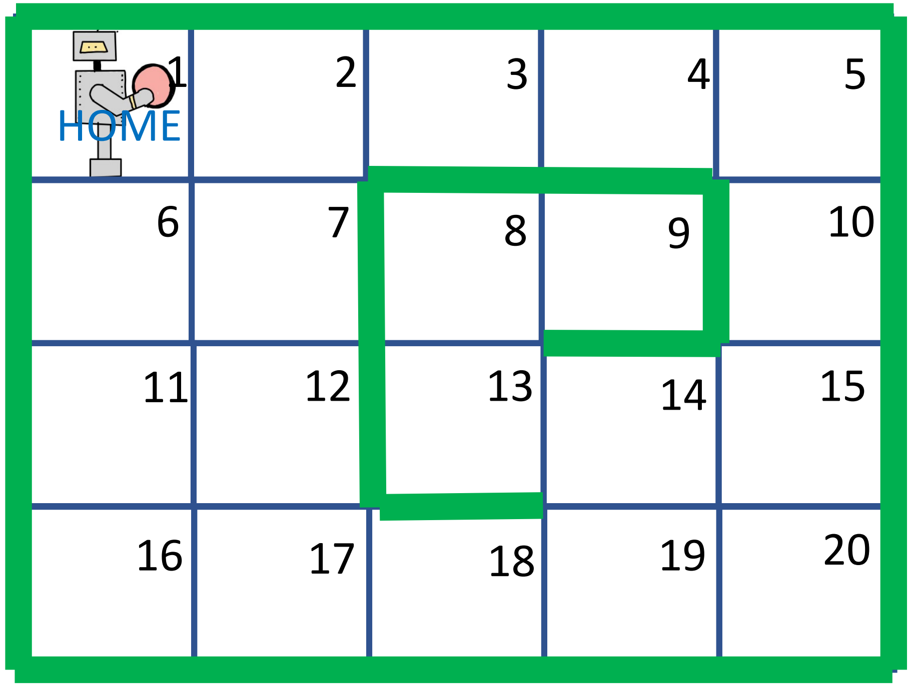
$R_{16}(t) \sim U[0,4]$

$R_i(t) \sim U[0,1]$

<u>Threshold-based heuristic</u>:

1. Robot moves from home to 16 by shortest path (ignoring all rewards it sees along the way).
2. Stay in 16 for ***only 1 slot***:
   If reward there with value $> \theta_1$:
      Pick up. Return home by shortest path.
   Else:
      Continue to 9 via shortest path
3. Wait in 9 until reward with value$> \theta_2$
4. Pick up. Return home by any shortest path.

Compare to Fine-Tuned Heuristic



Rewards appear iid Bern(1/2), then:

$R_9(t) \sim U[0,20]$

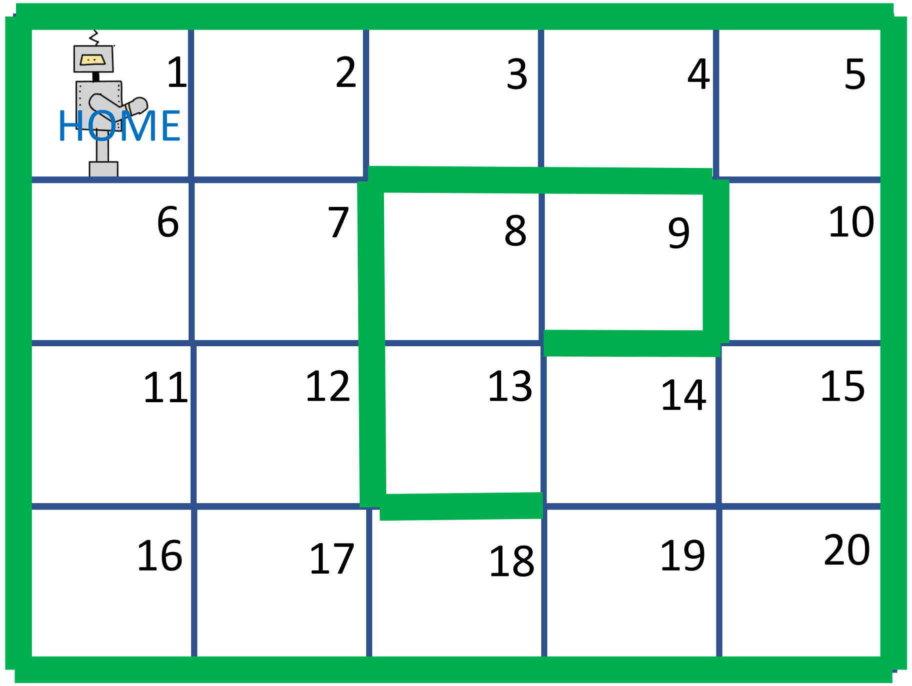
$R_{16}(t) \sim U[0,4]$

$R_i(t) \sim U[0,1]$

<u>Threshold-based heuristic</u>:

1. Robot moves from home to 16 by shortest path (ignoring all rewards it sees along the way).
2. Stay in 16 for ***only 1 slot***:
   If reward there with value > θ_1:
   Pick up. Return home by shortest path.
   Else:
   Continue to 9 via shortest path
3. Wait in 9 until reward with value> θ_2
4. Pick up. Return home by any shortest path.

Compare to Fine-Tuned Heuristic



Rewards appear iid Bern(1/2), then:

$R_9(t) \sim U[0,20]$

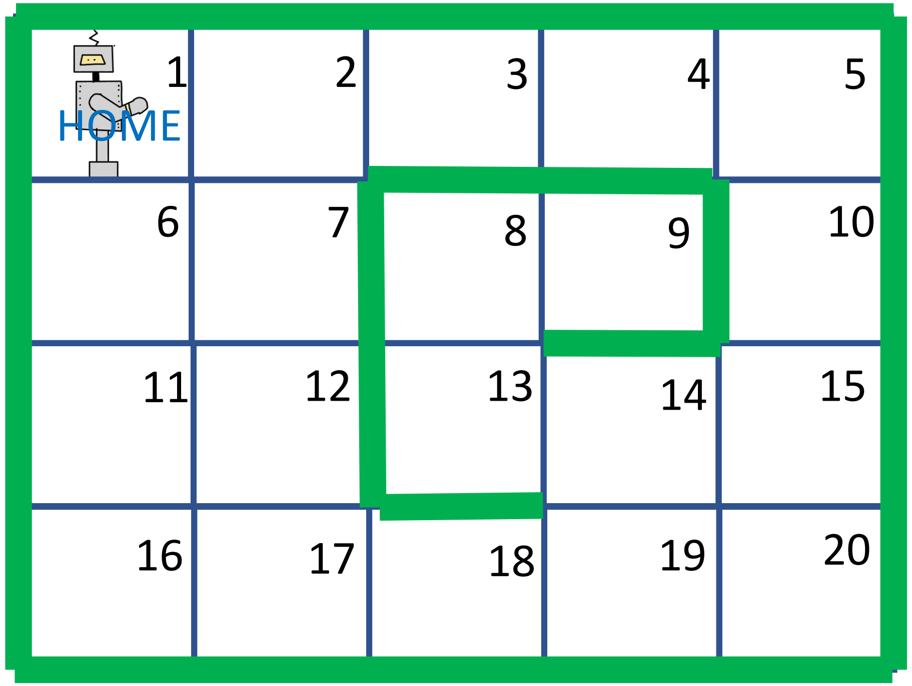
$R_{16}(t) \sim U[0,4]$

$R_i(t) \sim U[0,1]$

<u>Threshold-based heuristic</u>:

1. Robot moves from home to 16 by shortest path (ignoring all rewards it sees along the way).
2. Stay in 16 for ***only 1 slot***:
   If reward there with value > θ_1:
   Pick up. Return home by shortest path.
   Else:
   Continue to 9 via shortest path
3. Wait in 9 until reward with value> θ_2
4. Pick up. Return home by any shortest path.

Compare to Fine-Tuned Heuristic



Rewards appear iid Bern(1/2), then:

$R_9(t) \sim U[0,20]$

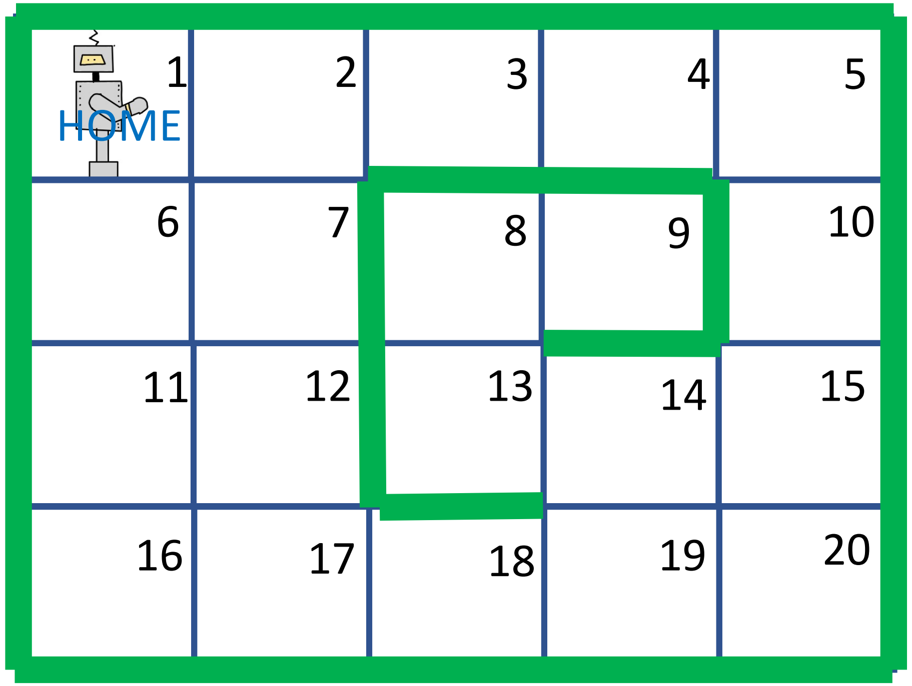
$R_{16}(t) \sim U[0,4]$

$R_i(t) \sim U[0,1]$

Threshold-based heuristic:

1. Robot moves from home to 16 by shortest path (ignoring all rewards it sees along the way).
2. Stay in 16 for ***only 1 slot***:
   If reward there with value $> \theta_1$:
   Pick up. Return home by shortest path.
   Else:
   Continue to 9 via shortest path
3. Wait in 9 until reward with value$> \theta_2$
4. Pick up. Return home by any shortest path.
5. Repeat.

Compare to Fine-Tuned Heuristic



Rewards appear iid Bern(1/2), then:

$R_9(t) \sim U[0,20]$

$R_{16}(t) \sim U[0,4]$

$R_i(t) \sim U[0,1]$

<u>Threshold-based heuristic</u>:

1. Robot moves from home to 16 by shortest path (ignoring all rewards it sees along the way).
2. Stay in 16 for ***only 1 slot***:
   If reward there with value $> \theta_1$:
      Pick up. Return home by shortest path.
   Else:
      Continue to 9 via shortest path
3. Wait in 9 until reward with value $> \theta_2$
4. Pick up. Return home by any shortest path.
5. Repeat.

Comparison to proposed online alg

Want to maximize time average reward.
Run over 10^6 slots.

1. Heuristic with best thresholds θ_1, θ_2: 0.6679 reward/time
2. Proposed *virtual system*: 0.6672 reward/time
3. Proposed *actual system*: 0.6604 reward/time

- Heuristic is fine-tuned with knowledge of problem structure and distribution
- Proposed MDP policy has no knowledge of problem structure or distribution

Idea

- Online algorithm on *virtual system*
- Use this to inform decision on *actual system*
- Treat p(t) as decision vector for desired steady state.
- Choose p(t) in prob simplex for n basic states:

$$p(t) \text{ in } \mathcal{P} = \{ (p_1, \ldots, p_n) : p_i \geq 0, \sum p_i = 1 \}$$

Same idea as:

[1] M. J. Neely, "Online Fractional Programming for Markov Decision Systems," Proc. Allerton Conf. on Communication, Control, and Computing, Sept. 2011.

Time averages for Virtual System

Minimize: $$\sum_{i=1}^{n} \overline{p_i(t) c_{i,0}(W(t), A_i(t))}$$

Subject to: $$\overline{p_j(t)} = \sum_{i=1}^{n} \overline{p_i(t) P_{ij}(W(t), A_i(t))} \quad \forall j \in \{1, \dots, n\}$$

$$\sum_{i=1}^{n} \overline{p_i(t) c_{i,l}(W(t), A_i(t))} \leq 0 \quad \forall l \in \{1, \dots, k\}$$

$$p(t) \in \mathcal{P} \quad \forall t \in \{0, 1, 2, \dots\}$$

$$A_i(t) \in \mathcal{A}_i \quad \forall t \in \{0, 1, 2, \dots\}$$

$p(t)$ and $W(t)$ are independent for $t \in \{0, 1, 2, \dots\}$

Time averages for Virtual System

Minimize: $\boxed{\sum_{i=1}^{n} \overline{p_i(t) c_{i,0}(W(t), A_i(t))}}$

Time average cost

Subject to: $\overline{p_j(t)} = \sum_{i=1}^{n} \overline{p_i(t) P_{ij}(W(t), A_i(t))} \quad \forall j \in \{1, \ldots, n\}$

$$\sum_{i=1}^{n} \overline{p_i(t) c_{i,l}(W(t), A_i(t))} \leq 0 \quad \forall l \in \{1, \ldots, k\}$$

$$p(t) \in \mathcal{P} \quad \forall t \in \{0, 1, 2, \ldots\}$$

$$A_i(t) \in \mathcal{A}_i \quad \forall t \in \{0, 1, 2, \ldots\}$$

$p(t)$ and $W(t)$ are independent for $t \in \{0, 1, 2, \ldots\}$

Time averages for Virtual System

Minimize: $$\sum_{i=1}^{n} \overline{p_i(t) c_{i,0}(W(t), A_i(t))}$$

Subject to: $$\boxed{\overline{p_j(t)} = \sum_{i=1}^{n} \overline{p_i(t) P_{ij}(W(t), A_i(t))} \quad \forall j \in \{1, \ldots, n\}}$$

Global Balance Eq

$$\sum_{i=1}^{n} \overline{p_i(t) c_{i,l}(W(t), A_i(t))} \leq 0 \quad \forall l \in \{1, \ldots, k\}$$

$$p(t) \in \mathcal{P} \quad \forall t \in \{0, 1, 2, \ldots\}$$

$$A_i(t) \in \mathcal{A}_i \quad \forall t \in \{0, 1, 2, \ldots\}$$

$p(t)$ and $W(t)$ are independent for $t \in \{0, 1, 2, \ldots\}$

Time averages for Virtual System

Minimize: $$\sum_{i=1}^{n} \overline{p_i(t)c_{i,0}(W(t), A_i(t))}$$

Subject to: $$\overline{p_j(t)} = \sum_{i=1}^{n} \overline{p_i(t)P_{ij}(W(t), A_i(t))} \quad \forall j \in \{1, \ldots, n\}$$

$$\boxed{\sum_{i=1}^{n} \overline{p_i(t)c_{i,l}(W(t), A_i(t))} \leq 0 \quad \forall l \in \{1, \ldots, k\}}$$

Additional avg constraints

$$p(t) \in \mathcal{P} \quad \forall t \in \{0, 1, 2, \ldots\}$$

$$A_i(t) \in \mathcal{A}_i \quad \forall t \in \{0, 1, 2, \ldots\}$$

$p(t)$ and $W(t)$ are independent for $t \in \{0, 1, 2, \ldots\}$

Time averages for Virtual System

Minimize: $$\sum_{i=1}^{n} \overline{p_i(t) c_{i,0}(W(t), A_i(t))}$$

Subject to: $$\overline{p_j(t)} = \sum_{i=1}^{n} \overline{p_i(t) P_{ij}(W(t), A_i(t))} \quad \forall j \in \{1, \ldots, n\}$$

$$\sum_{i=1}^{n} \overline{p_i(t) c_{i,l}(W(t), A_i(t))} \leq 0 \quad \forall l \in \{1, \ldots, k\}$$

$$\boxed{\begin{array}{l} p(t) \in \mathcal{P} \quad \forall t \in \{0, 1, 2, \ldots\} \\ A_i(t) \in \mathcal{A}_i \quad \forall t \in \{0, 1, 2, \ldots\} \end{array}}$$

$p(t)$ and $W(t)$ are independent for $t \in \{0, 1, 2, \ldots\}$

Treat p(t) as decision variable

Time averages for Virtual System

Minimize: $\sum_{i=1}^{n} \overline{p_i(t) c_{i,0}(W(t), A_i(t))}$

Subject to: $\overline{p_j(t)} = \sum_{i=1}^{n} \overline{p_i(t) P_{ij}(W(t), A_i(t))} \quad \forall j \in \{1, \ldots, n\}$

$$\sum_{i=1}^{n} \overline{p_i(t) c_{i,l}(W(t), A_i(t))} \leq 0 \quad \forall l \in \{1, \ldots, k\}$$

$$p(t) \in \mathcal{P} \quad \forall t \in \{0, 1, 2, \ldots\}$$

$$A_i(t) \in \mathcal{A}_i \quad \forall t \in \{0, 1, 2, \ldots\}$$

$\boxed{p(t) \text{ and } W(t) \text{ are independent for } t \in \{0, 1, 2, \ldots\}}$

Nonstandard!

How to enforce?

Relation to LP formulations

- Structure is reminiscent of LP formulations for basic (non-opportunistic) MDPs
- Our problem structure is nonconvex
- Similar in spirit to linear fractional program:

  [2] B. Fox, "Markov renewal programming by linear fractional programming," Siam J. Appl. Math 1966.

  [3] S. Boyd, L. Vandenberghe, *Convex Optimization*, 2004.

- Our opportunistic structure has a unique (*and pesky*) independence constraint

New idea

Single timescale hierarchical decision structure:

1. Choose $p(t)$ in $\mathcal{P}$ without knowledge of $W(t)$.

2. Force $p(t)$ and $p(t-1)$ to be close by adding Kullback-Liebler penalty:

$$D(p(t); p(t-1)) = \sum p_i(t) \log(p_i(t)/p_i(t-1))$$

3. Observe $W(t)$. Make *contingency action*

$A_i(t)$ in $\mathcal{A}_i$ for each i in $\{1, \ldots, n\}$

”Max-Weight” Alg on Virtual System

Virtual queue for GBE constraint:

$$Q_j(t+1) = Q_j(t) + p_j(t) - \sum_i p_i(t-1)\, P_{ij}(W(t-1),A_i(t-1))$$

Layer 1: Choose p(t) in P to minimize

$$p(t)^T\,[\ (1/\varepsilon)C_0(t-1) + Y(t-1)Q(t) + G(t-1)Z(t)\ ] + \alpha D(p(t);p(t-1))$$

Layer 2: For each i in {1, …, n} choose $A_i(t)$ to minimize:

$$(1/\varepsilon)c_{i,0}(\ W(t), A_i(t)\ ) + Z^T(t)C_i(\ W(t), A_i(t)\ ) - Q(t)^T P_i(\ W(t), A_i(t)\ )$$

Dimension of W(t) or cardinality of its set of possible values is irrelevant!

Corresponding Actual System

If *actual system* is in state $S(t) = i$, then choose $A(t) = A_i(t)$

Theorem on Virtual System

Given parameter $\varepsilon>0$ for the algorithm.

<u>Theorem</u>: After time $T=O(1/\varepsilon^2)$ we have

1. (virtual time avg cost) $\leq$ (optimal) + $O(\varepsilon)$

2. (virtual time avg penalty) $\leq O(\varepsilon)$

3. (virtual time avg. GBE violation) $\leq O(\varepsilon)$

Relation to Actual System

At every time t, the *actual and virtual systems match* in

1. conditional transition probabilities

$$P[\ S(t+1)=j \mid S(t)=i\ ] = \text{match for all } i, j$$

2. conditional costs

$$E[\ C_k(t) \mid S(t)=i\ ] = \text{match for all } i, k$$

Relation to Actual System

At every time t, the *actual and virtual systems match* in

1. conditional transition probabilities

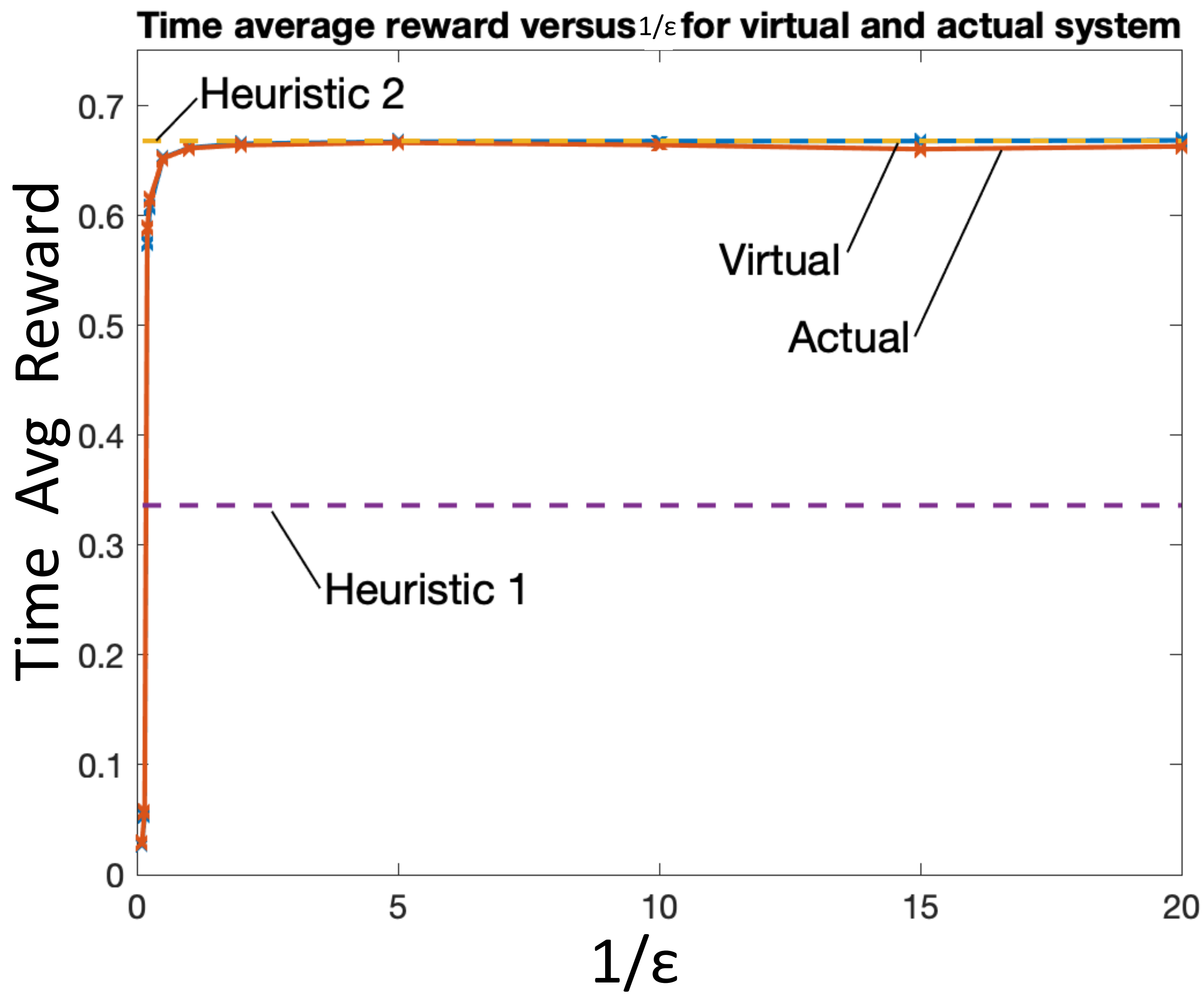
$$P[\, S(t+1)=j \mid S(t)=i \,] = \text{match for all } i, j$$

2. conditional costs

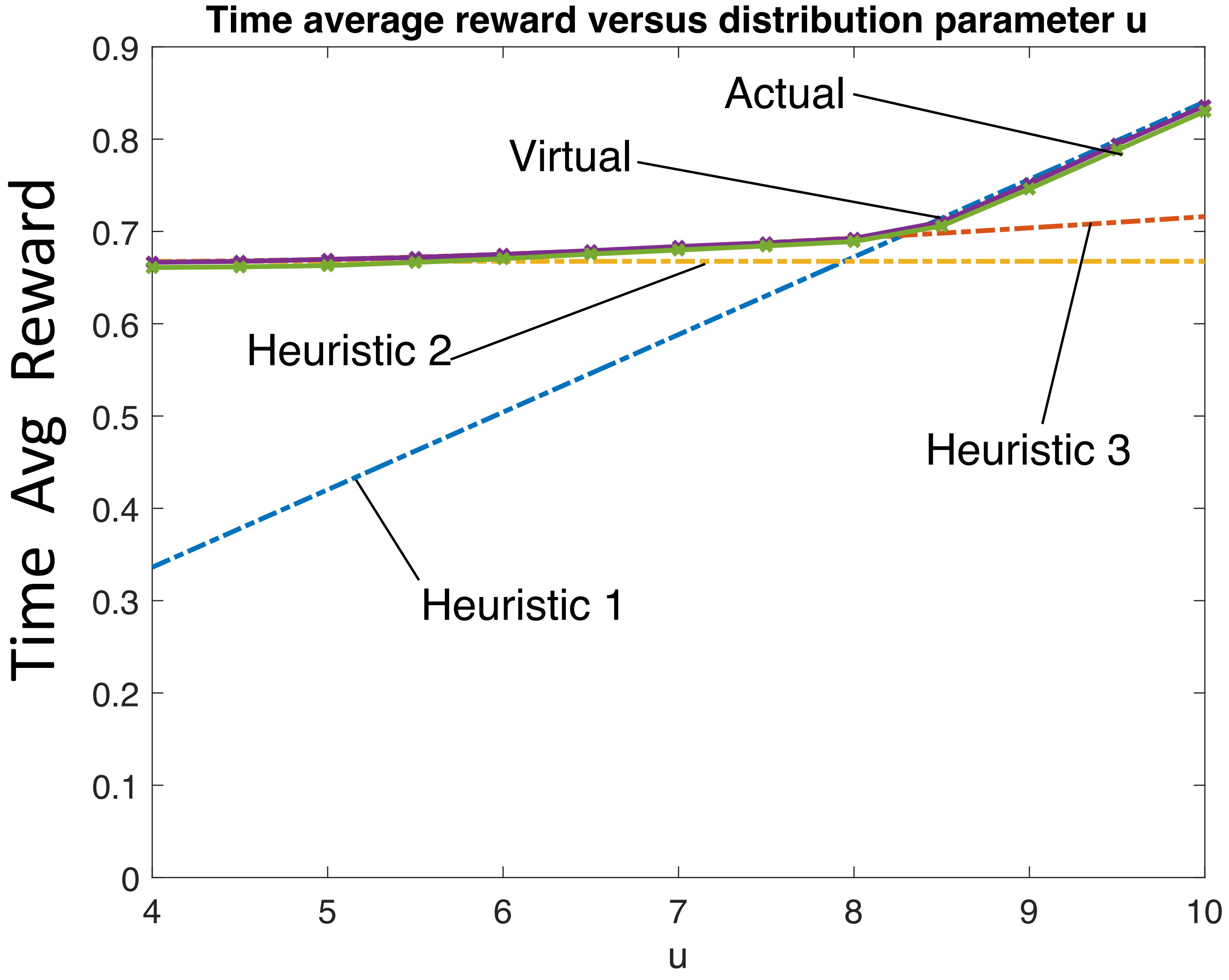
$$E[\, C_k(t) \mid S(t)=i \,] = \text{match for all } i, k$$

Caveat: No proof that unconditionals match!

Let’s verify by simulation



Varying a distribution parameter u



Simulated steady state in virtual system

(actual system has similar numbers)

Holding

HOME .041	.022	.015	.016	.016
.019	.011	.041	.000 [0,20]	.016
.014	.017	.041	.041	.016
.009 [0,4]	.025	.025	.025	.000

Not holding

HOME .000	.018	.010	.010	.010
.023	.014	.041	.224 [0,20]	.010
.017	.015	.041	.041	.010
.016 [0,4]	.031	.031	.031	.000

Conclusion: Opportunistic MDPs

- Extends Lyapunov drift theory to Markov decision systems

- Learning via time-averaged GBEs

- Overcomes a challenging independence constraint

- Easily incorporates additional constraints on power, cost, etc.

- Complexity and convergence *independent of dimension of W(t).*
  (Also independent of cardinality of set of possible W(t) values.)