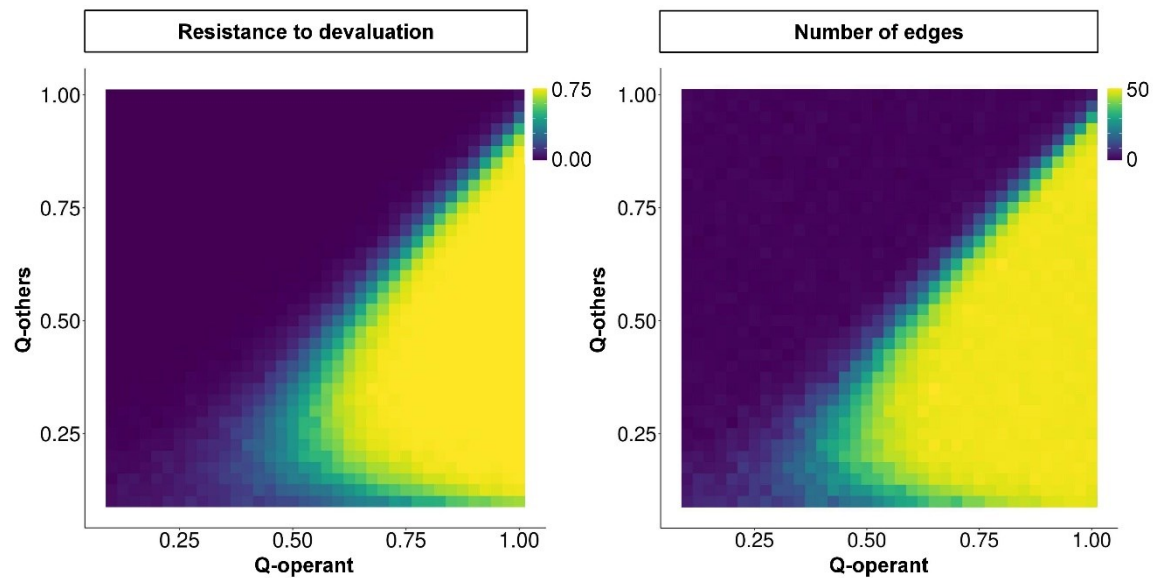


Supplementary Information

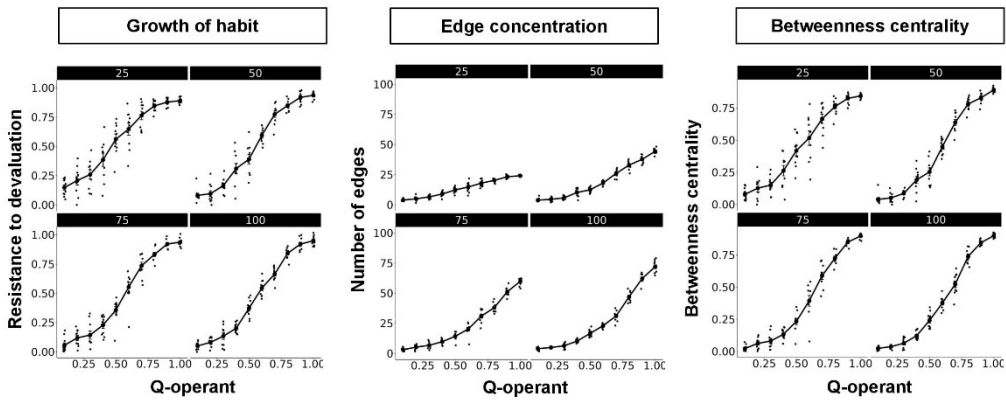
Supplementary Figures



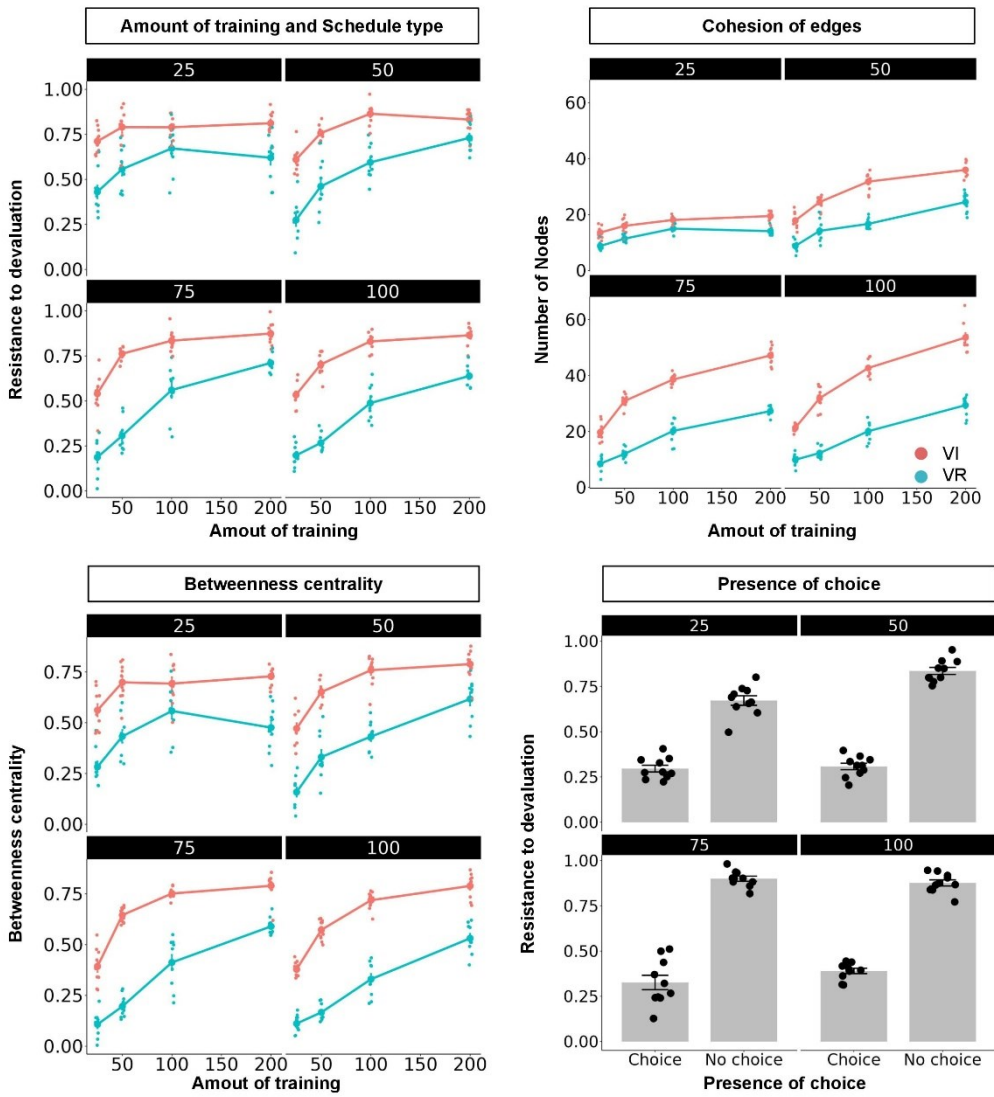
Supplementary Figure 1. Effects of systematic manipulation of Q_{operant} and Q_{others} on habit formation.

The dependencies of the resistance to devaluation (left) and number of edges that the operant response acquired (right) on the Q_{operant} and Q_{others} . As the Q_{operant} increased, resistance to devaluation and number of edges increased, suggesting we confirmed the same result in the Simulation 1.

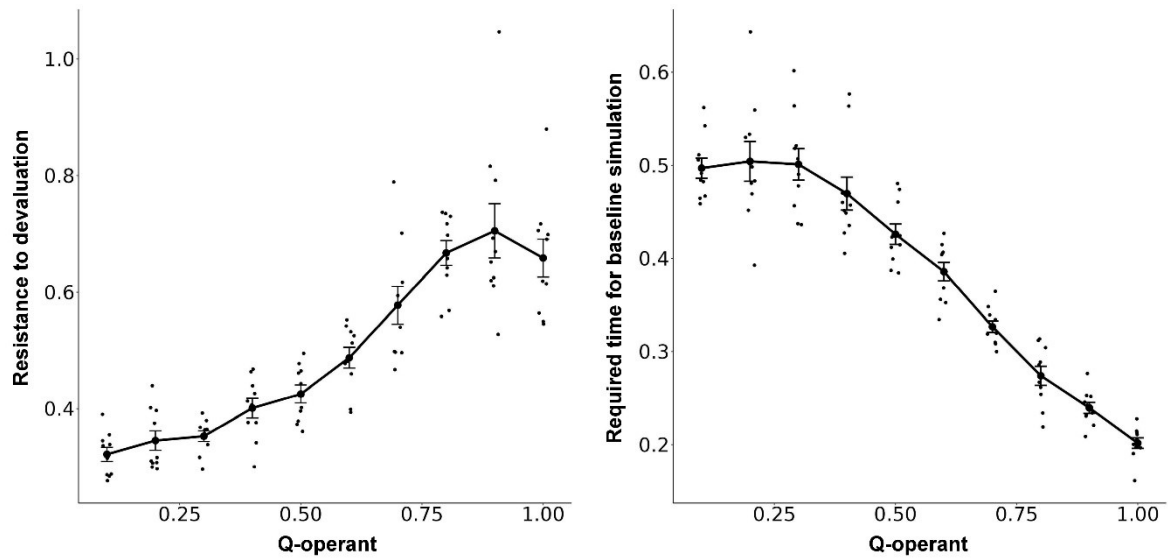
a



b

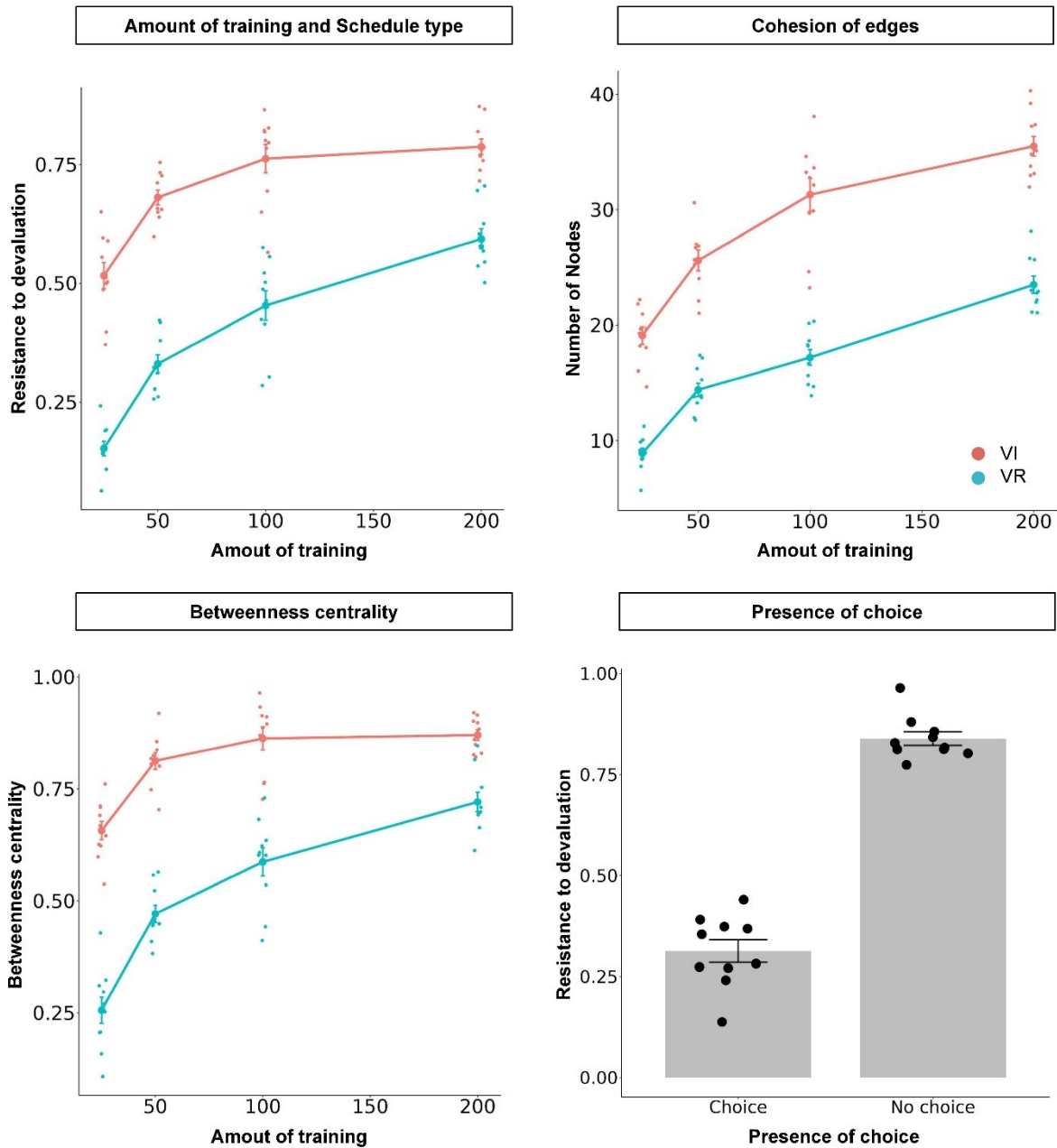


12 Supplementary Figure 2. Simulation results replicating Figure 2 (simulation 1), with the
13 different numbers of nodes (25–100).
14 We manipulate the number of nodes, 25, 50, 75, and 100, to confirm the results of our
15 simulation are replicated in different numbers of nodes and all results are replicated.
16

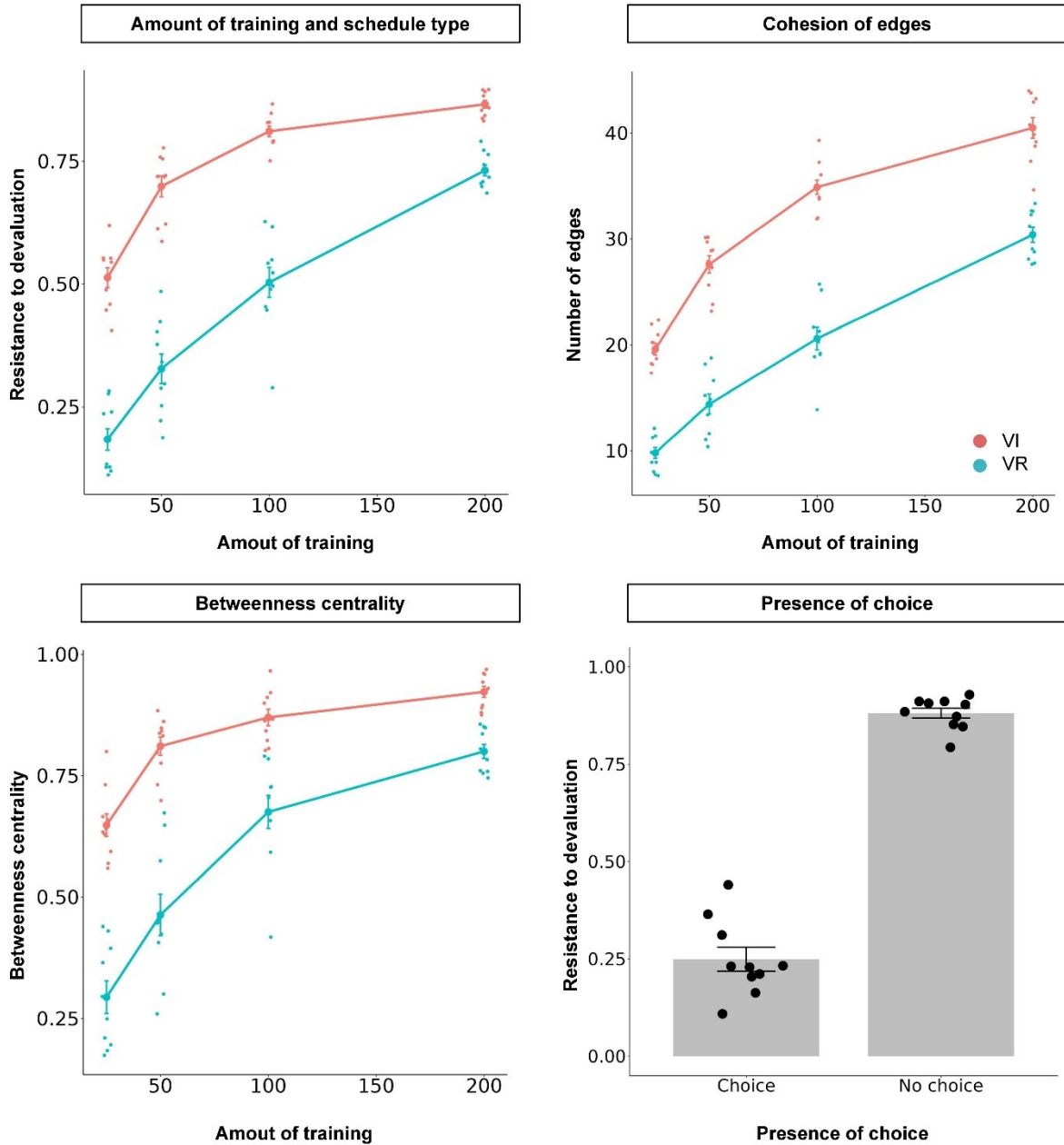


Supplementary Figure 3. Reproducibility of the results of Simulation 1 with a different response sequence generation algorithm.

In the Simulation 1, response sequences were generated by a shortest path search, Dijkstra's algorithm. We employed another algorithm that is more weakly constrained and not the shortest path searching algorithm. In the new algorithm, an agent chooses a response randomly if a response chosen as a goal is not connected to the current engaging response. If the goal response is connected to the current engaging response, the agent chooses the response. In other words, the agent searches the goal response locally in the new algorithm. Resistance to devaluation, Edge concentration and betweenness centrality, all of features are replicated with the new algorithm, suggesting habit formation does not depend on the shortest path search as long as the response sequences are generated goal-directed.



Supplementary Figure 4. Simulation results replicating Figure 3 (simulation 2), with a different schedule for other responses from the original simulation. We employed the VI 360 s schedule instead of FR 1 for other responses. We set their reward values as $1 / 50$.



36

37 Supplementary Figure 5. Simulation results replicating Figure 3 (simulation 3), with the
 38 different algorithm SARSA from the original algorithm Q-learning.

39

40 Supplementary Methods

Algorithm 1 Learn Q-matrix in the given environment

INPUT: N , $schedule$, $amout_training$

OUTPUT: Q

Q = generate an $N \times N$ matrix with all elements set to zero.

a_{t-1} = choose a response randomly from $[0, 1, 2, \dots, N]$

$response_durations$ = sample N samples from exponential distribution with $\lambda = 1 / 2.5$.

$r_{operant} = 0$

while $r_{operant} < amout_training$:

a_t = choose a response with softmax function and Q-values

τ = engage the chosen response for $response_durations[a_t]$ seconds.

r_t = schedule receive a_t and τ and return a reward

 update $Q(a_t, a_{t+1})$ according to Eq. 1

 if $a_t == 0$ and $r_t == 1$:

$r_{operant} += 1$

$a_{t-1} = a_t$

return Q

41

42 Supplementary Algorithm 1. How agents learn Q-matrix in a given environment.

43

Algorithm 2 Variable ratio schedule

INPUT: p , rewards , amout_training

OUTPUT: schedule

q = Divide the range 0 - 1 into N equal parts

$\text{reward_count} \leftarrow 0$

$\text{required_response} = q[\text{reward_count}]$ th quantile of geometry distribution with parameter p .

def $\text{schedule}(a_t, \tau)$:

 if $a_t == 0$:

$\text{required_response} -= 1$

 else:

 return $\text{rewards}[a_t]$

 if $\text{required_response} \leq 0$:

$\text{reward_count} += 1$

$\text{required_response} = q[\text{reward_count}]$ th quantile of geometry distribution with parameter p .

 return $\text{rewards}[a_t]$

 return schedule

44

45 Supplementary Algorithm 2. The implementation of the variable ratio schedule.

46

Algorithm 3 Variable interval schedule

INPUT: λ , *rewards*, *amout_training*

OUTPUT: *schedule*

q = Divide the range 0 - 1 into N equal parts

reward_count = 0

required_time = $q[\text{reward_count}]$ th quantile of exponential distribution with parameter λ .

def *schedule*(a_t , τ):

required_time -= τ

 if $a_t == 0$ and *required_time* <= 0:

reward_count += 1

required_time = $q[\text{reward_count}]$ th quantile of exponential distribution with parameter λ .

 return *rewards*[a_t]

 elif $a_t \neq 0$:

 return *rewards*[a_t]

return *schedule*

47

48 Supplementary Algorithm 3. The implementation of the variable interval schedule.

49

Algorithm 4 Concurrent VI VI schedule

INPUT: λ , *rewards*, *amout_training*

OUTPUT: *schedule*

q_1 = Divide the range 0 - 1 into N equal parts

q_2 = Divide the range 0 - 1 into N equal parts

reward_count_1 = 0

reward_count_2 = 0

required_time_1 = $q_1[\text{reward_count_1}]$ th quantile of exponential distribution with parameter λ .

required_time_2 = $q_2[\text{reward_count_2}]$ th quantile of exponential distribution with parameter λ .

def *schedule*(a_t , τ):

required_time_1 -= τ

required_time_2 -= τ

 if $a_t == 0$ and *required_time_1* <= 0:

reward_count_1 += 1

required_time_1 = $q_1[\text{reward_count_1}]$ th quantile of exponential distribution with parameter λ .

 return *rewards*[a_t]

 if $a_t == 1$ and *required_time_2* <= 0:

reward_count_2 += 1

required_time_2 = $q_2[\text{reward_count_2}]$ th quantile of exponential distribution with parameter λ .

 return *rewards*[a_t]

 elif $a_t \neq 0$:

 return *rewards*[a_t]

return *schedule*

50

51

Supplementary Algorithm 4. The implementation of the concurrent VI VI schedule.

52

Algorithm 5 Concurrent VI VT schedule

INPUT: λ , *rewards*, *amout_training*

OUTPUT: *schedule*

q_1 = Divide the range 0 - 1 into N equal parts

q_2 = Divide the range 0 - 1 into N equal parts

reward_count_1 = 0

reward_count_2 = 0

required_time_1 = $q_1[\text{reward_count_1}]$ th quantile of exponential distribution with parameter λ .

required_time_2 = $q_2[\text{reward_count_2}]$ th quantile of exponential distribution with parameter λ .

def *schedule*(a_t , r):

required_time_1 -= r

required_time_2 -= r

 if $a_t == 0$ and *required_time_1* <= 0:

reward_count_1 += 1

required_time_1 = $q_1[\text{reward_count_1}]$ th quantile of exponential distribution with parameter λ .

reward = *rewards*[a_t]

 elif $a_t \neq 0$:

reward = *rewards*[a_t]

 if *required_time_2* <= 0:

reward_count_2 += 1

required_time_2 = $q_2[\text{reward_count_2}]$ th quantile of exponential distribution with parameter λ .

reward += *rewards*[1]

 return *schedule*

53

54

Supplementary Algorithm 5. The implementation of the concurrent VI VT schedule.

55

Algorithm 6 Tandem VR VI schedule

INPUT: $p, \lambda, \text{rewards}, \text{amout_training}$

OUTPUT: *schedule*

q = Divide the range 0 - 1 into N equal parts

$\text{reward_count} = 0$

$\text{required_response} = q[\text{reward_count}]$ th quantile of geometry distribution with parameter p .

$\text{required_time} = q[\text{reward_count}]$ th quantile of geometry distribution with parameter λ .

def *schedule*(a_t, τ):

 if $a_t == 0$:

$\text{required_response} -= 1$

 else:

 return $\text{rewards}[a_t]$

 if $\text{required_response} \leq 0$:

$\text{required_time} -= \tau$

 if $\text{required_time} \leq 0$. and $a_t == 0$:

$\text{reward_count} += 1$

$\text{required_response} = q[\text{reward_count}]$ th quantile of geometry distribution with parameter p .

$\text{required_time} = q[\text{reward_count}]$ th quantile of geometry distribution with parameter λ .

 return $\text{rewards}[a_t]$

return *schedule*

56

57 Supplementary Algorithm 6. The implementation of the tandem VR VI schedule.

58

Algorithm 7 Tandem VI VR schedule

INPUT: $p, \lambda, \text{rewards}, \text{amout_training}$

OUTPUT: *schedule*

q = Divide the range 0 - 1 into N equal parts

$\text{reward_count} = 0$

$\text{required_response} = q[\text{reward_count}]$ th quantile of geometry distribution with parameter p .

$\text{required_time} = q[\text{reward_count}]$ th quantile of geometry distribution with parameter λ .

def *schedule*(a_t, r):

$\text{required_time} -= r$

 if $a_t == 0$ and $\text{required_time} \leq 0$:

$\text{required_response} -= 1$

 elif $a_t \neq 0$:

 return $\text{rewards}[a_t]$

 if $\text{required_response} \leq 0$:

$\text{reward_count} += 1$

$\text{required_response} = q[\text{reward_count}]$ th quantile of geometry distribution with parameter p .

$\text{required_time} = q[\text{reward_count}]$ th quantile of exponential distribution with parameter λ .

 return $\text{rewards}[a_t]$

return *schedule*

59

60 Supplementary Algorithm 7. The implementation of the tandem VI VR schedule.

61

Algorithm 8 Procedure of baseline and devaluation phases

INPUT: *Network, N, rewards, loop*

OUTPUT: *propotion_operant*

s = choose an initial response from 0 - *N* randomly

operant = 0

total = 0

for *_* in 1:*loop*:

t = choose a response according to Eq. 4

shortest_path = find a shortest path from *s* to *t* on the *Network*

total += number of response contained in *shortest_path*

 if 0 in *shortest_path*:

operant += 1

s = *t*

return *operant* / *total*

62

63 Supplementary Algorithm 8. How agents behave in the baseline and devaluation phases.