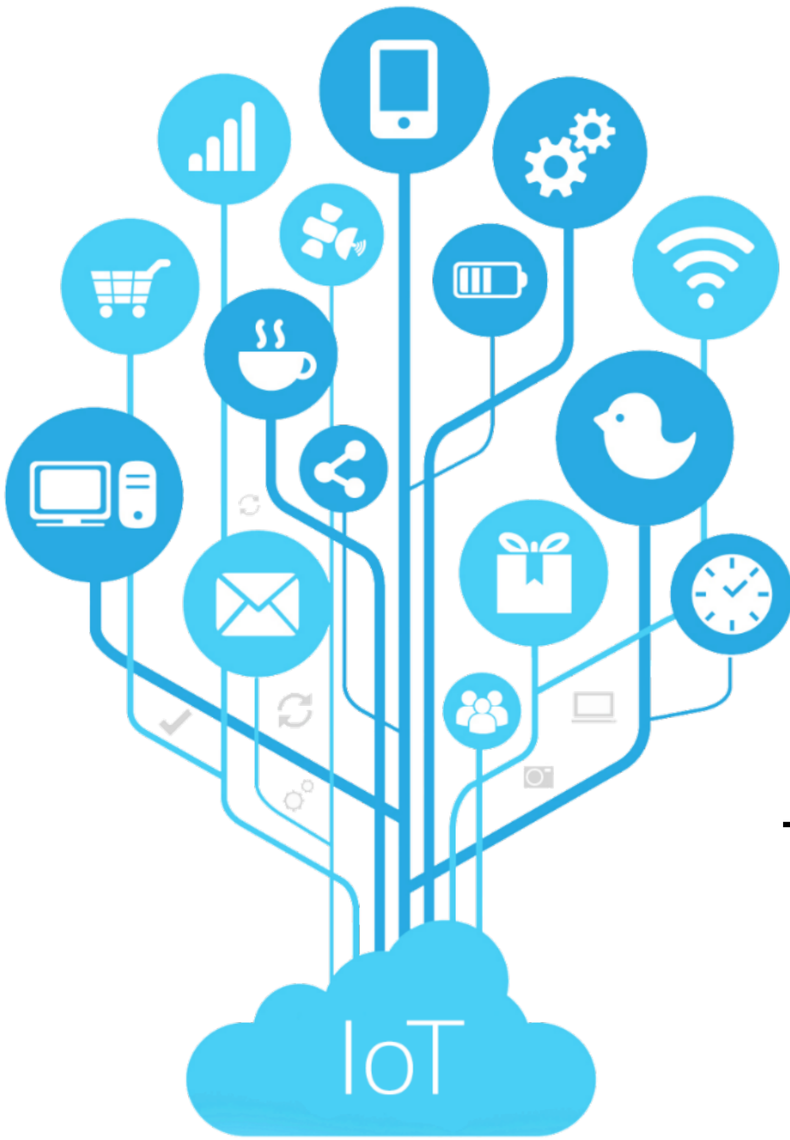
Robot Path Planning in Wireless Communication:

Using Reinforcement Learning

Anushka Vashishtha
The University of Hong Kong
Final Presentation
April 2019

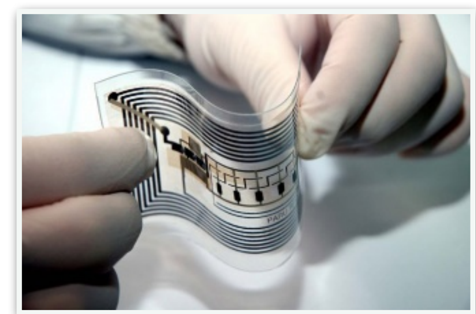
Objective



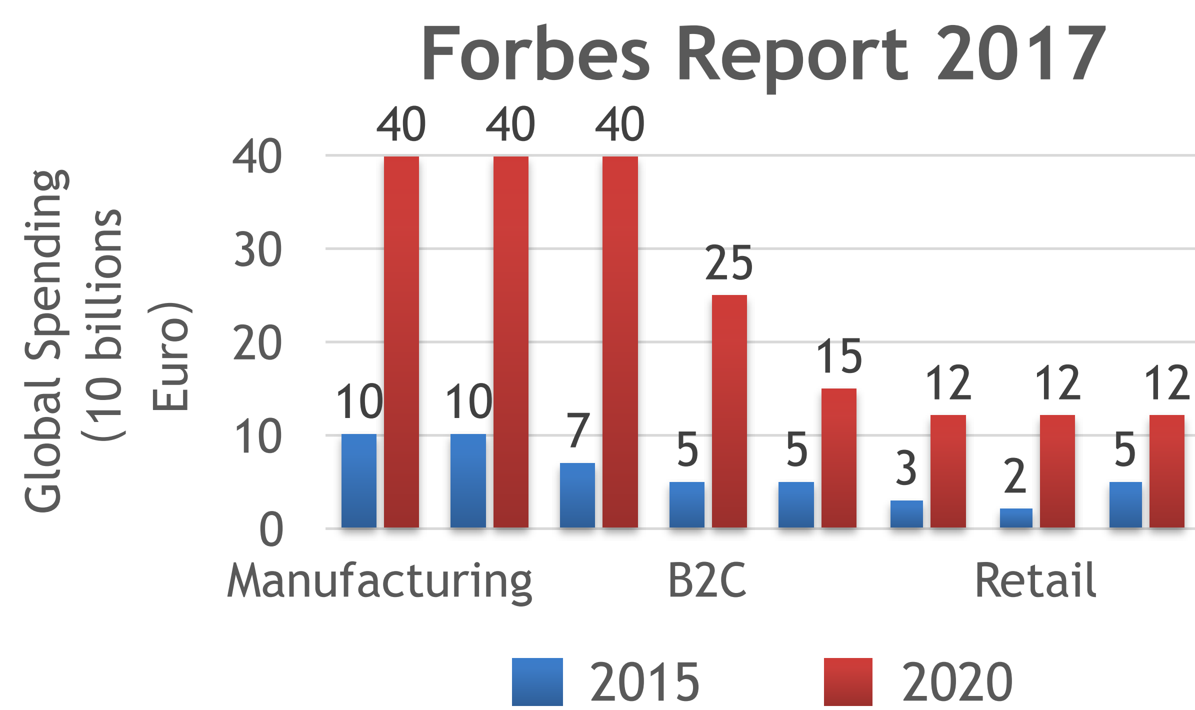
Internet of Things

Tiny Size

Charging DIFFICULT



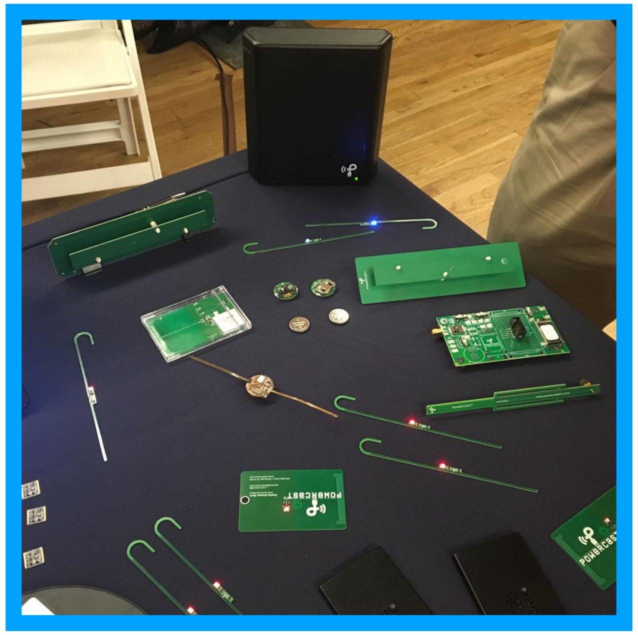
Huge Number



Background

<u>Wireless Power and Information Transmission</u>

- Powercast demonstrated its distance-charging technique in New York City in the summer of 2017.
- Collaborator demonstrated a prototype WPIT system in Guangzhou in 2018.



Smart Warehouse

Path Planning

Charging Region Model

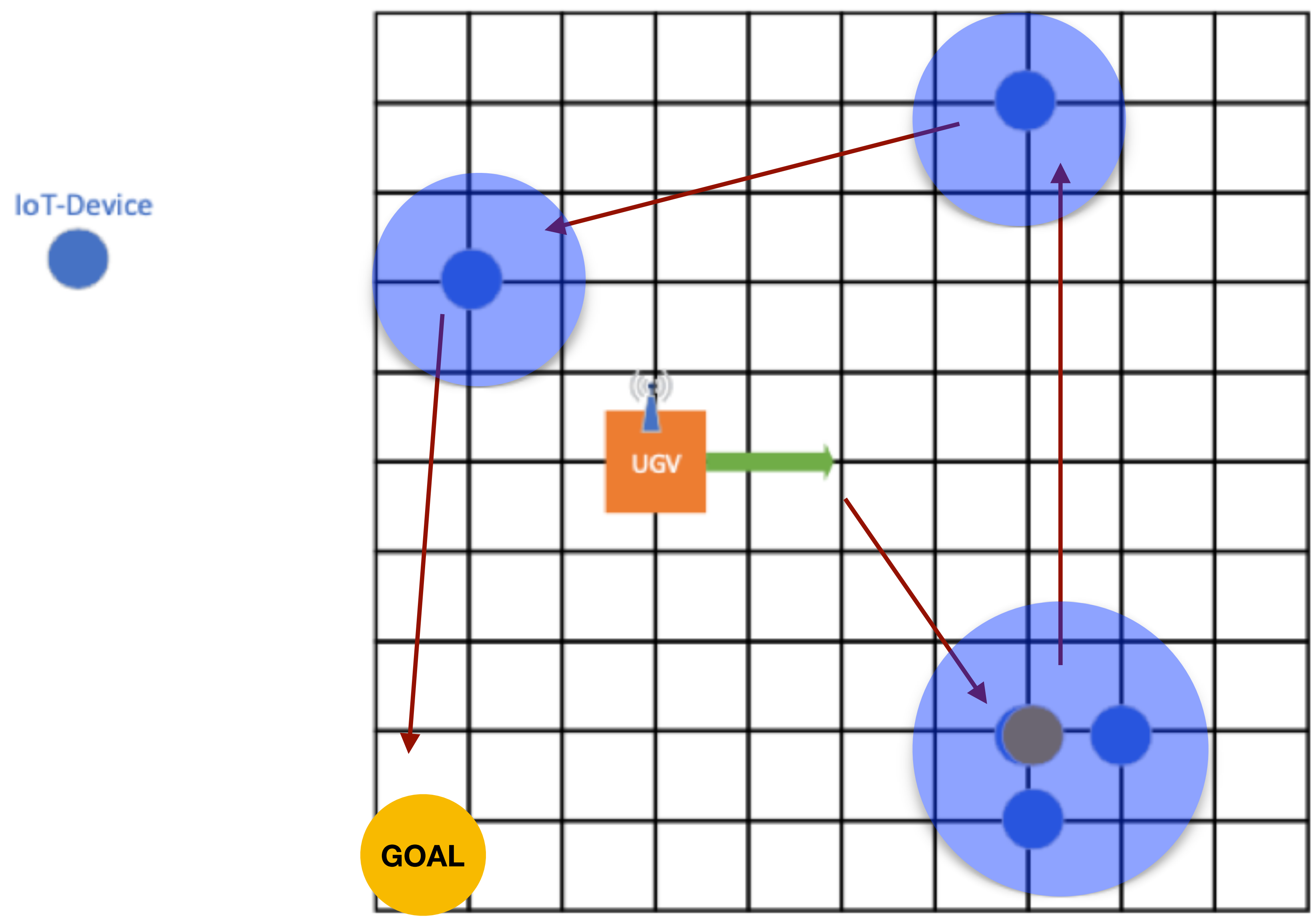


Figure 1 Graphical abstraction of the problem

Multi Integer Non-Linear Programming

$$\min_{\mathbf{v}, \mathbf{X}, \{\lambda_m\}} E_M = (\frac{\alpha_1}{a} + \alpha_2)\mathrm{Tr}(\mathbf{D}^T\mathbf{W})$$

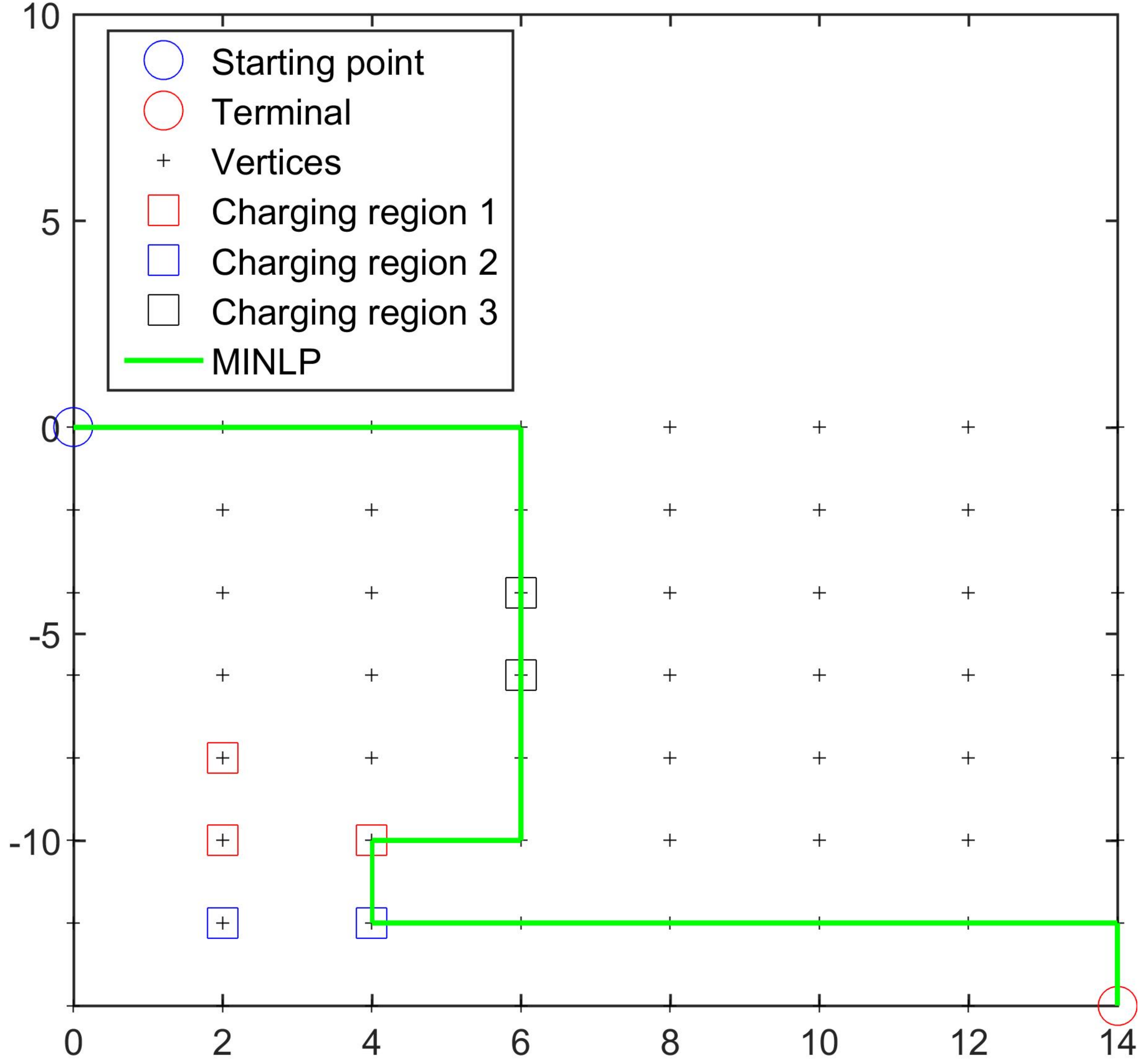
Energy Lost due to movement in Joules

Parameter Table

$\mathbf{v}$	Visit a point in the grid or not (Boolean variable)
$\mathbf{X}$	Link between two points in the grid or not (Boolean matrix)
α_1, α_2	Toning parameter: pioneer's 3DX robot experiment result at MIT(constant)
a	Velocity of UGV (constant)
$\mathbf{D}$	Distance between two point in the grid or not (Boolean matrix)
$\mathbf{W}$	Summation of X values (matrix)
M	length & width of the grid (variable)
K	Total number of IoT devices (variable)

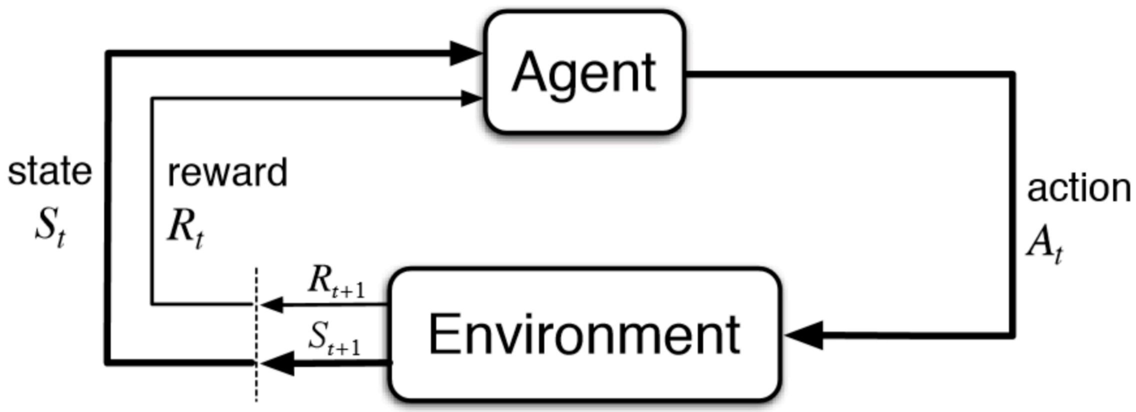
$$
\begin{aligned}
\text{s.t.}\quad & v_1 = v_M = 1,\ (\textit{select starting and end points}) \\
& v_m \in \{0,1\},\ \ \forall 2 \le m \le M-1,\ (\textit{selection is binary}) \\
& \sum_{m \in \mathcal{C}_i} v_m \ge 1,\ \ \forall i = 1,\cdots,K,\ (\textit{charge all IoT users}) \\
& W_{m,j} \in \{0,1\},\ \ \forall m,j,\ \ W_{m,m} = 0,\ \ \forall m,\ (\textit{flow selection is binary}) \\
& \sum_{j=1}^{M} W_{1,j} = 1,\ \sum_{j=1}^{M} W_{j,1} = 0,\ (\textit{flow from starting point}) \\
& \sum_{j=1}^{M} W_{M,j} = 0,\ \sum_{j=1}^{M} W_{j,M} = 1,\ (\textit{flow to end point}) \\
& \sum_{j=1}^{M} W_{m,j} = v_m,\ \sum_{j=1}^{M} W_{j,m} = v_m,\ \ \forall m = 2,\cdots,M, \\
& (\textit{flow passing selected points; no flow passing abandoned points}) \\
& \lambda_m - \lambda_j + \left(\sum_{l=1}^{M-1} v_l - 1\right) W_{m,j} + \left(\sum_{l=1}^{M-1} v_l - 3\right) W_{j,m} \\
& \le \sum_{l=1}^{M-1} v_l - 2 + J\left(2 - v_m - v_j\right),\ \ \forall 2 \le m,j \le M-1,\ m \ne j, \\
& v_m \le \lambda_m \le \left(\sum_{l=1}^{M-1} v_l - 1\right) v_m,\ \ \forall m \ge 2. \\
& (\textit{guarantee flow connected})
\end{aligned}
$$

Constraints



MINLP : lower bound

Q-learning



$$R_{t+1} = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+2} + \gamma^3 r_{t+2} + \ldots + \gamma^{n-t-1} r_n$$

$$q_*(s,a) = \max_{\pi} q_\pi(s,a)$$

$$q_*(s,a) = E[R_{t+1} + \gamma \max_{a'} q_*(s',a')]$$

$$S = \{(x,y) \,|\, x,y \in [M]\} \ (\mathit{location\ of\ each\ point\ on\ the\ grid})$$

$$A = \{up, down, left, right\}$$

$$R(x_i,y_i) = \begin{cases} 10 + (5 * x) & \textbf{if } (x_i,y_i) = v_G,\ x = \textbf{ no. of IoT devices charged} \\ 10 & \textbf{if } (x_i,y_i) \textbf{ in charging region of a particular IoT for the first time} \\ -1 & \textbf{otherwise.} \end{cases}$$

Q-learning: Learn function $Q : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$

Require:

Sates $\mathcal{X} = \{1, \dots, n_x\}$

Actions $\mathcal{A} = \{1, \dots, n_a\}$, $\qquad A : \mathcal{X} \Rightarrow \mathcal{A}$

Reward function $R : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$

Black-box (probabilistic) transition function $T : \mathcal{X} \times \mathcal{A} \to \mathcal{X}$

Learning rate $\alpha \in [0, 1]$, typically $\alpha = 0.1$

Discounting factor $\gamma \in [0, 1]$

procedure QLEARNING($\mathcal{X}$, A, R, T, α, γ)

  Initialize $Q : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ arbitrarily

  while Q is not converged **do**

    Start in state $s \in \mathcal{X}$

    while s is not terminal **do**

      Calculate π according to Q and exploration strategy (e.g. $\pi(x) \leftarrow \arg\max_a Q(x, a)$)

      $a \leftarrow \pi(s)$

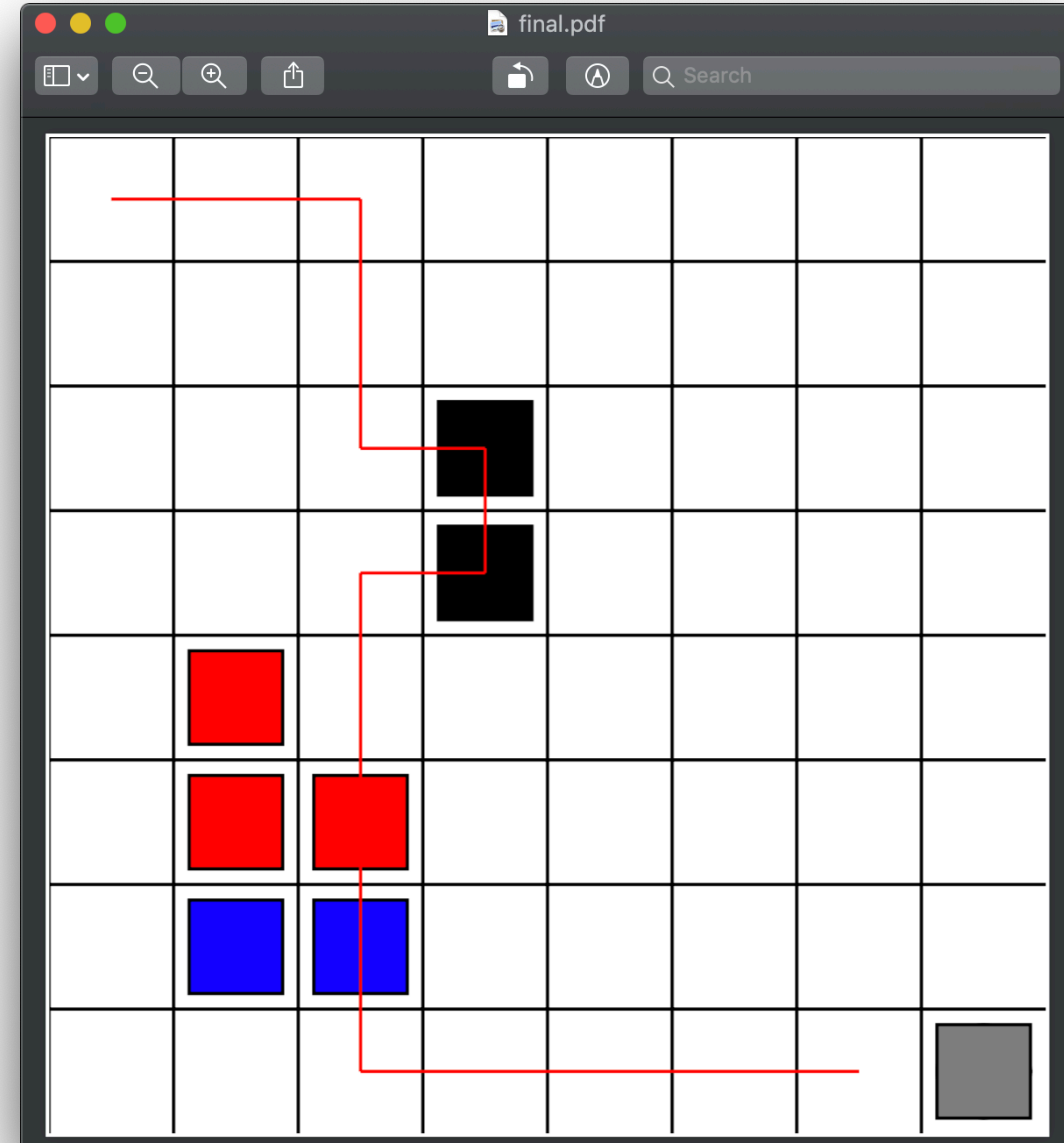
      $r \leftarrow R(s, a)$ ▷ Receive the reward

      $s' \leftarrow T(s, a)$ ▷ Receive the new state

      $Q(s', a) \leftarrow (1 - \alpha) \cdot Q(s, a) + \alpha \cdot (r + \gamma \cdot \max_{a'} Q(s', a'))$

      $s \leftarrow s'$

  return Q



Grid size: 8x8 Q-learning

Real World Scenario

- Dyna-Q (direct reinforcement learning + model learning)
  - Rewards can change
  - Uses Tabular Q-planning & Tabular Q-learning
  - Useful when not enough data

- Increase grid size from 8x8
  - Is Q-learning still convenient ?

Deep Q-learning

- ✤ Q-learning + Deep neural network
  - DQN approximates to optimal Q-function

$$q_*(s, a) - q_(s, a) = loss$$

$$q_*(s, a) = E[R_{t+1} + \gamma \max_{a'} q_*(s', a')]$$

  - Accommodate large grid size
  - Need more training episodes

- ✤ Features
  - Experienced Replay $e_t = (s_t, a_t, r_{t+1}, s_{t+1})$
  - Fixed Q-targets $target\_net\ updates\ every\ C\ steps$

deep Q-learning with experience replay and fixed Q-targets

evaluate_net or build_net

target_net

Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with weights $\bar{\theta} = \theta$
for $k = 1,\ M$ **do**
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  for $t = 1,\ T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \arg\max_a(Q(\phi(s_t), a; \theta))$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
    Sample random mini-batch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D

No correlation

$$\text{Set } y_j = \begin{cases} r_j & \text{episode ends}(j+1) \\ r_j + \gamma \arg\max'_a(Q(\phi(t+1), a'; \bar{\theta})) & \text{otherwise} \end{cases}$$

    Perform gradient descent step on $(y_j - Q(\phi(j), a_j; \theta)^2$ with respect to the network parameter θ
    Every C step reset $\hat{Q} = Q$

Efficient convergence to optimal Q-function

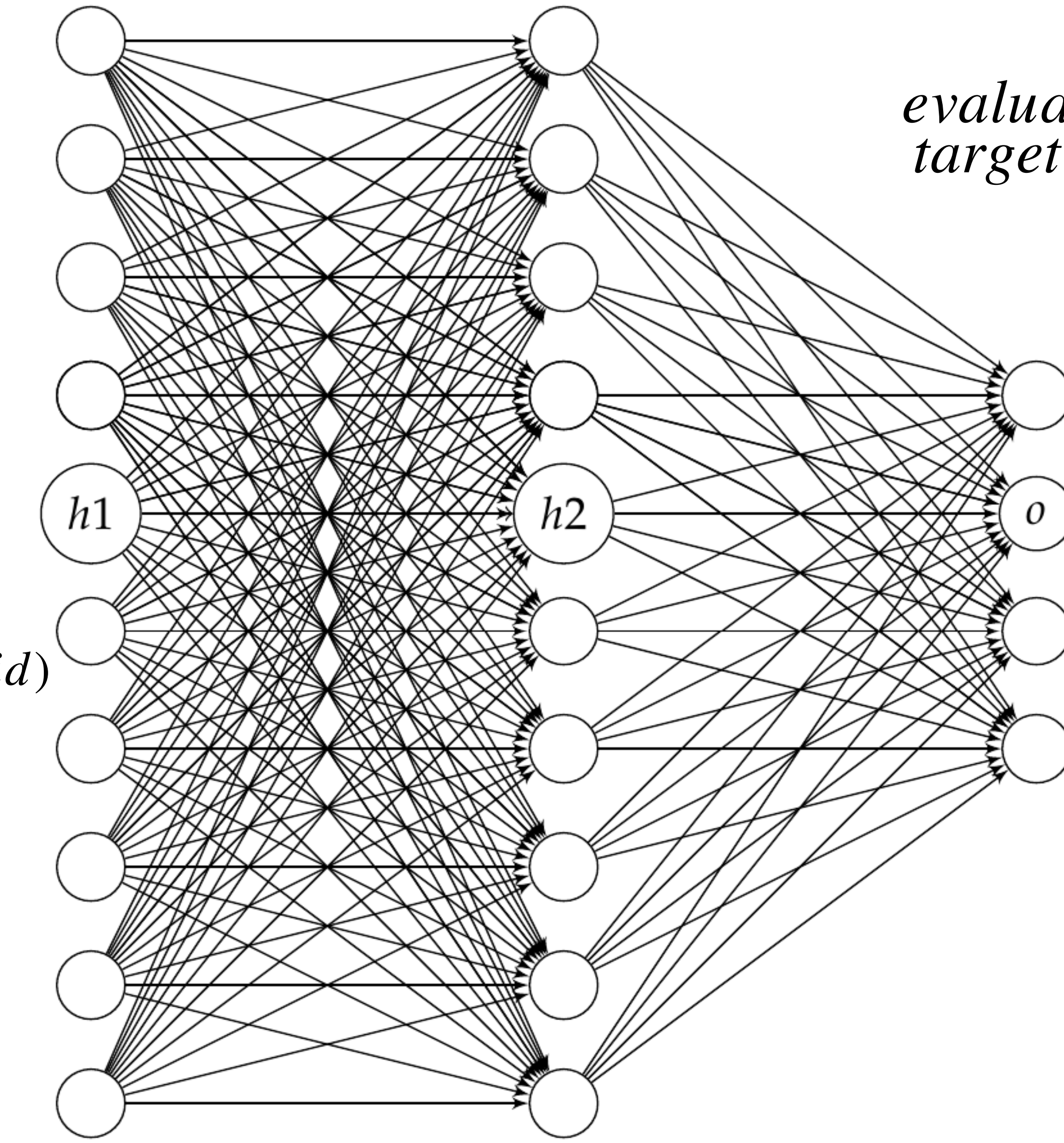
Neural Network Architecture

evaluate_net or build_net
target_net

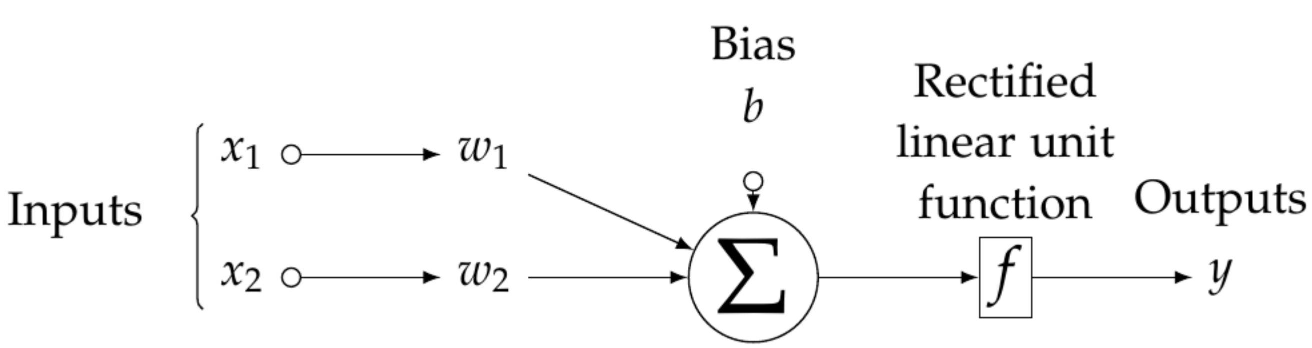
Input -> Hidden layers

(x_1, x_2)

$(location\ of\ each\ point\ on\ the\ grid)$



Output layer



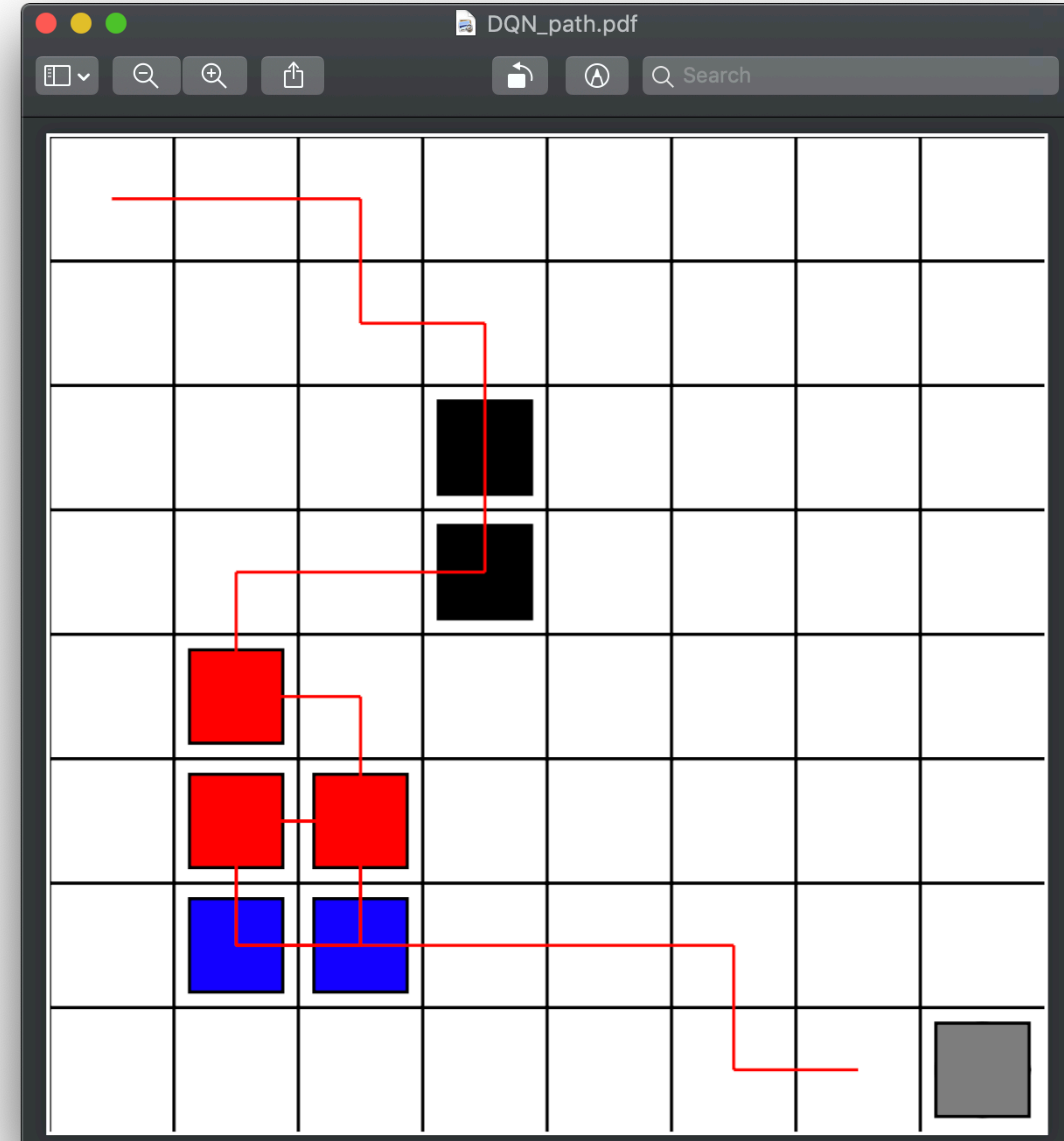
$0 \le w_1 \le 0.3$

$0 \le w_2 \le 0.3$

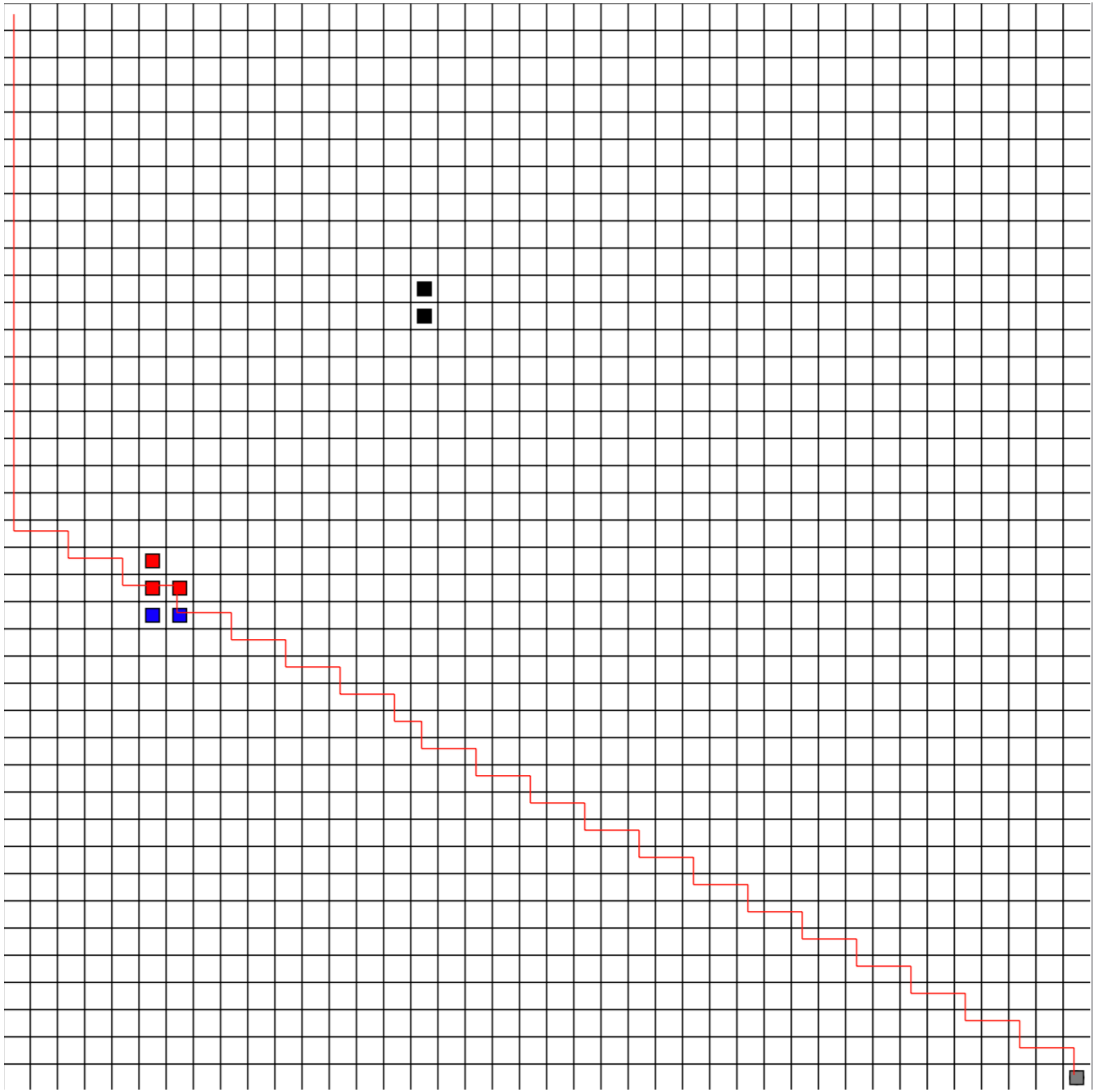
$b = 0.1$

Mean squared error as loss

RMSProp as optimisation function



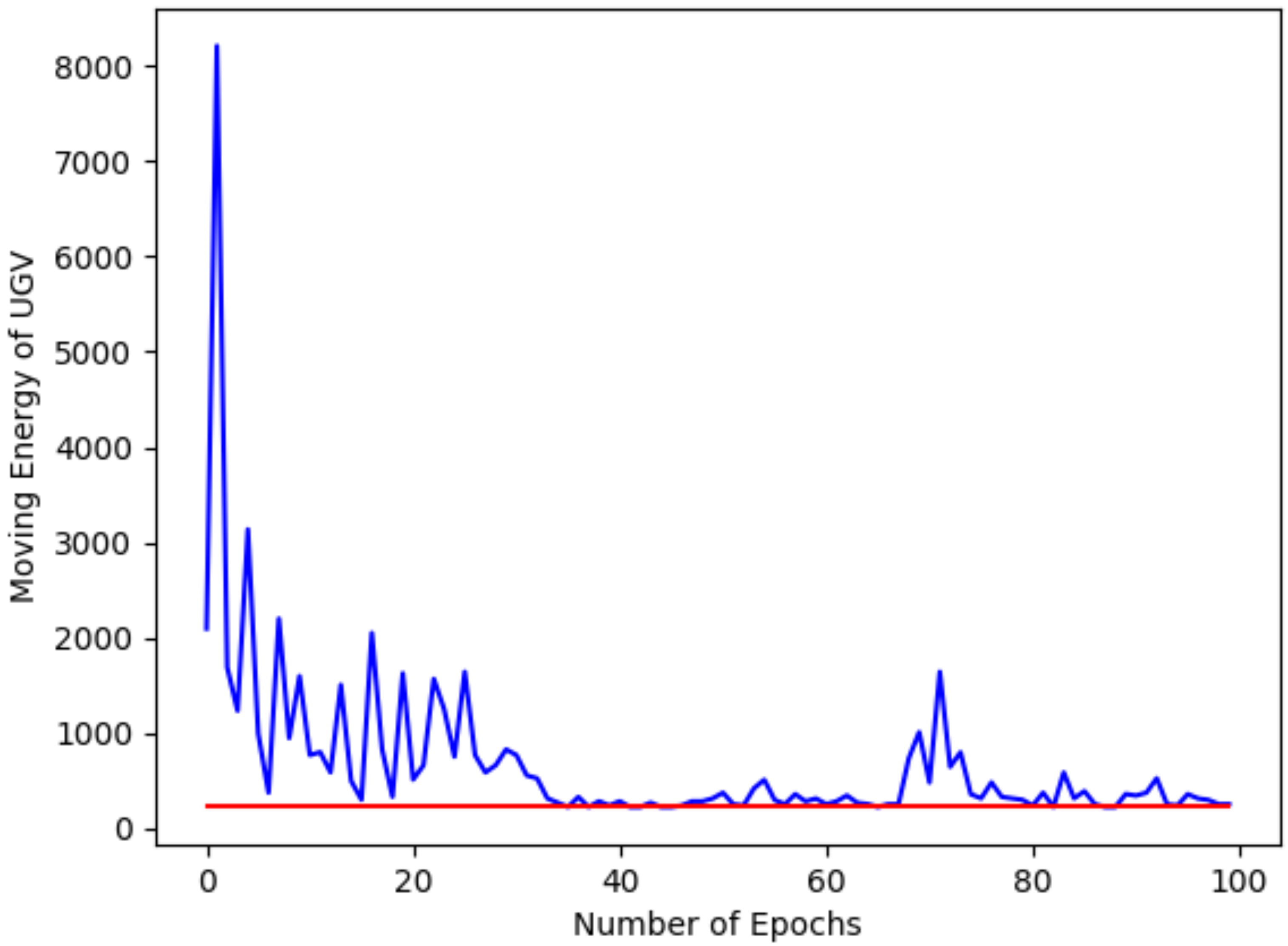
Grid size: 8x8 deep Q-learning



Grid size: 40x40 deep Q-learning

Results

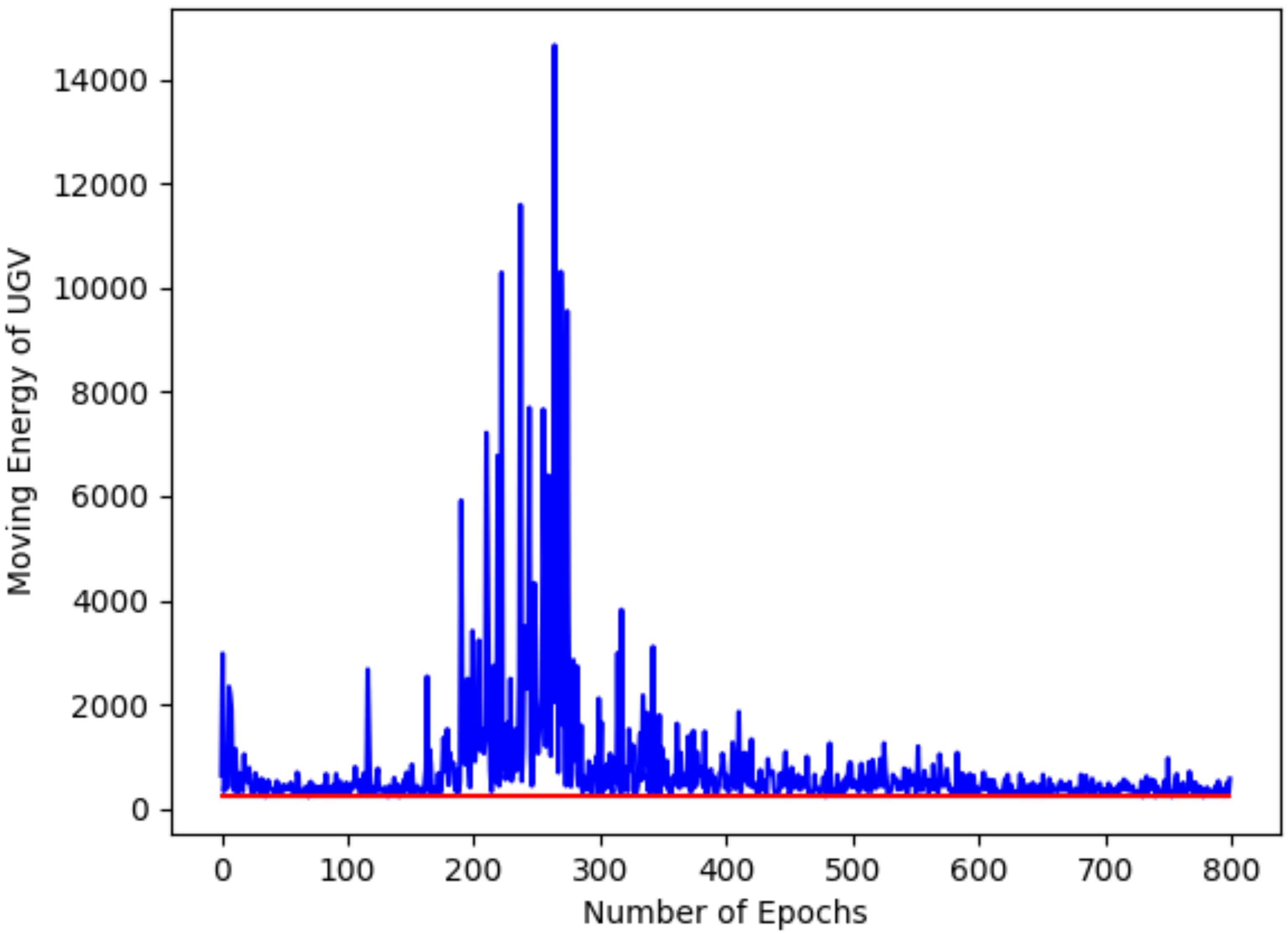
Grid size: 8x8



Q-Learning (252 J)

MINLP (241J)

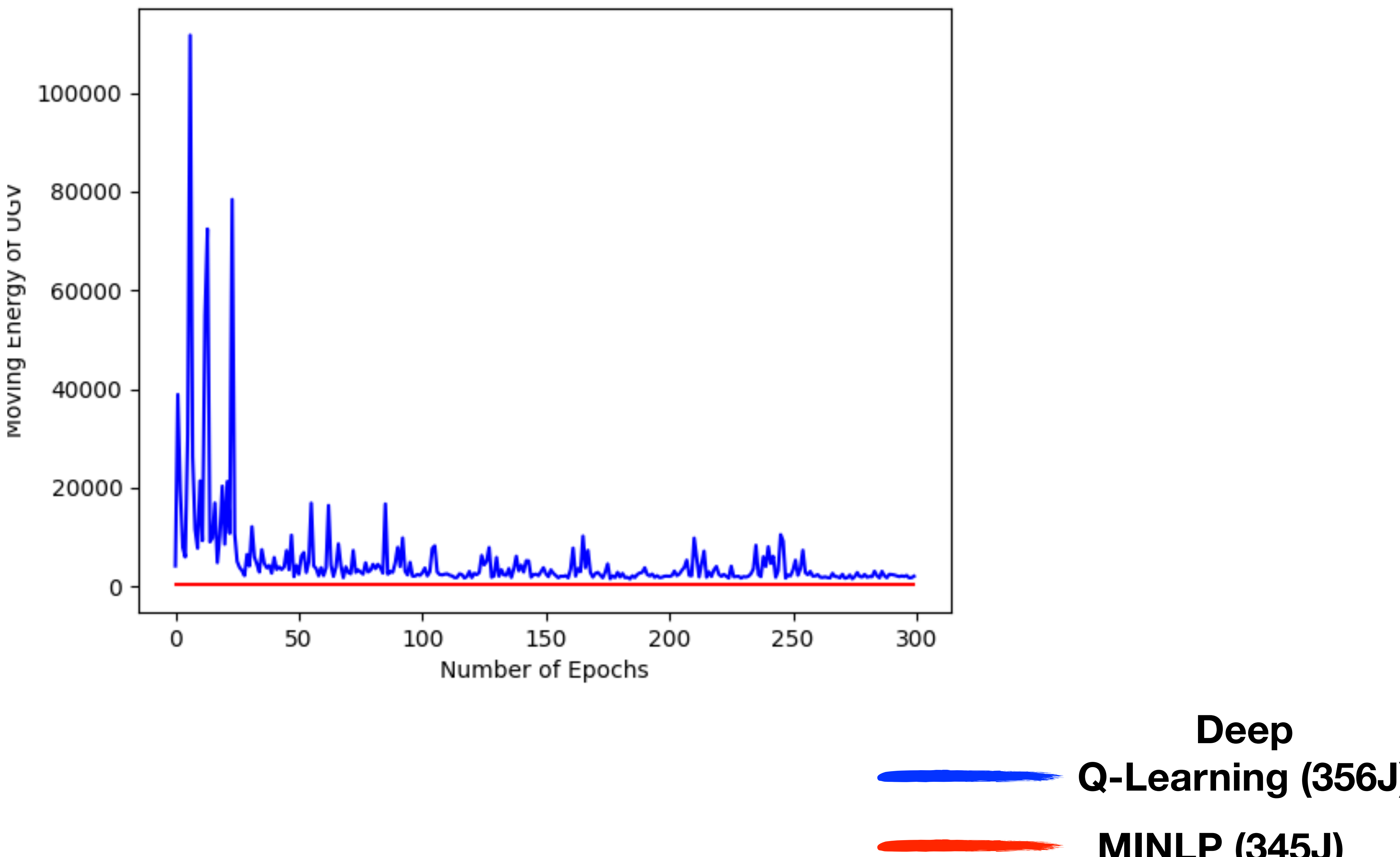
Grid size: 8x8



Deep Q-Learning (246J)

MINLP (241J)

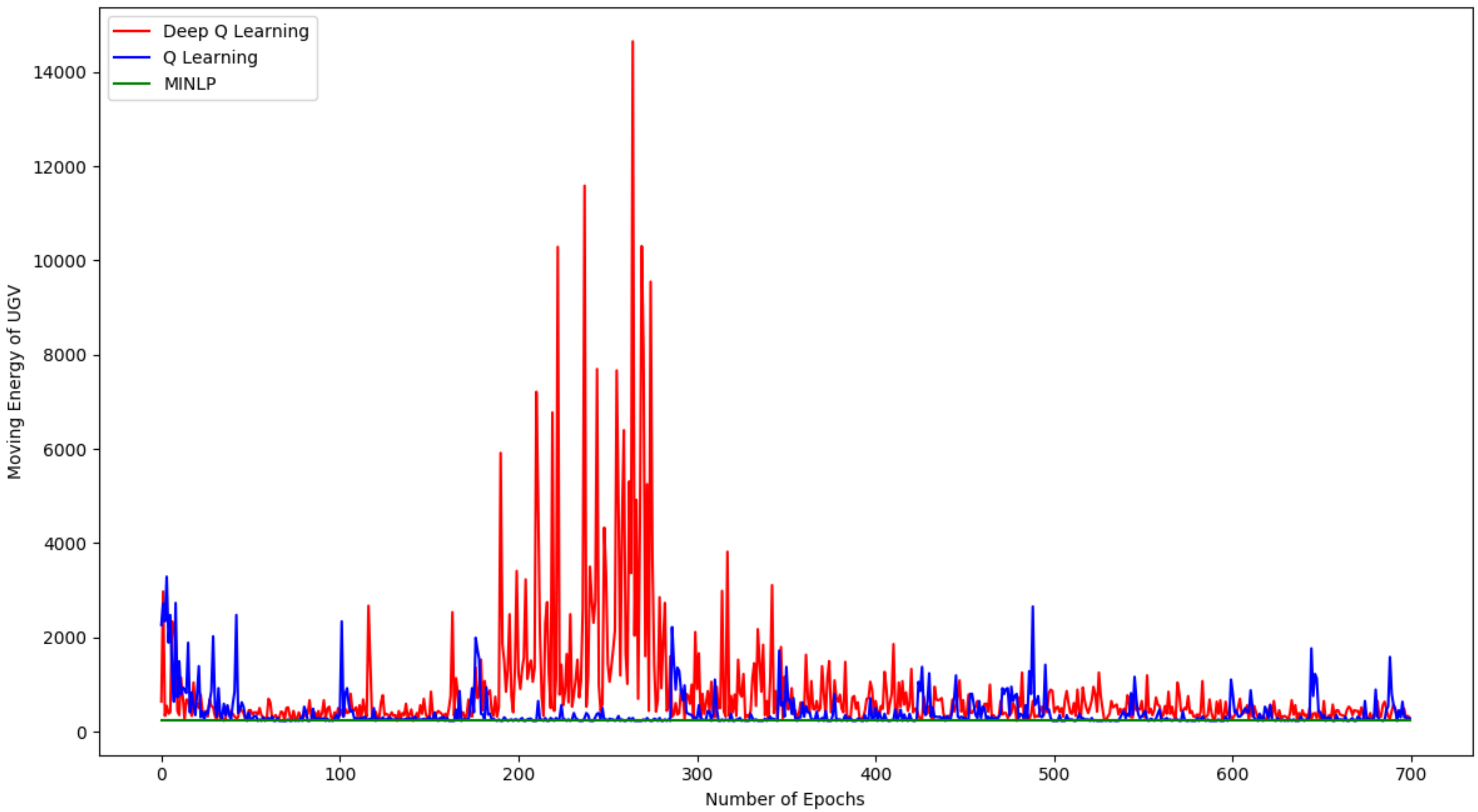
Grid size: 40x40



All Methods Comparison

Q-Learning
Deep Q-Learning
MINLP

Grid size: 8x8

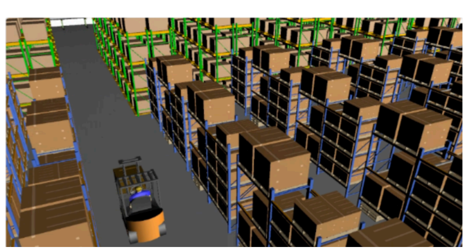


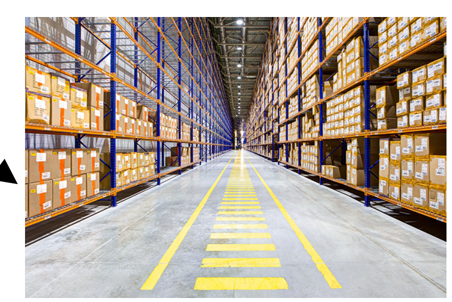
Conclusion

- Q- learning
  - Small operation area
  - Not enough training required
  - More effective with Dyna-Q if reward changes

- Deep Q- learning
  - Supports large operation area
  - Stable and more accurate≠ correlation and converges to optimal Q function

Simulated environment





Real world environment

Future Work

- Variable power given by UGV to IoT devices according to distance from IoT device
- Federated Q-Learning
- Limited energy present in UGV
- Safe Exploration
- Add convolutional layers to detect location of IoT devices
- Continuous charging model (multi-armed bandit)

Thank You!

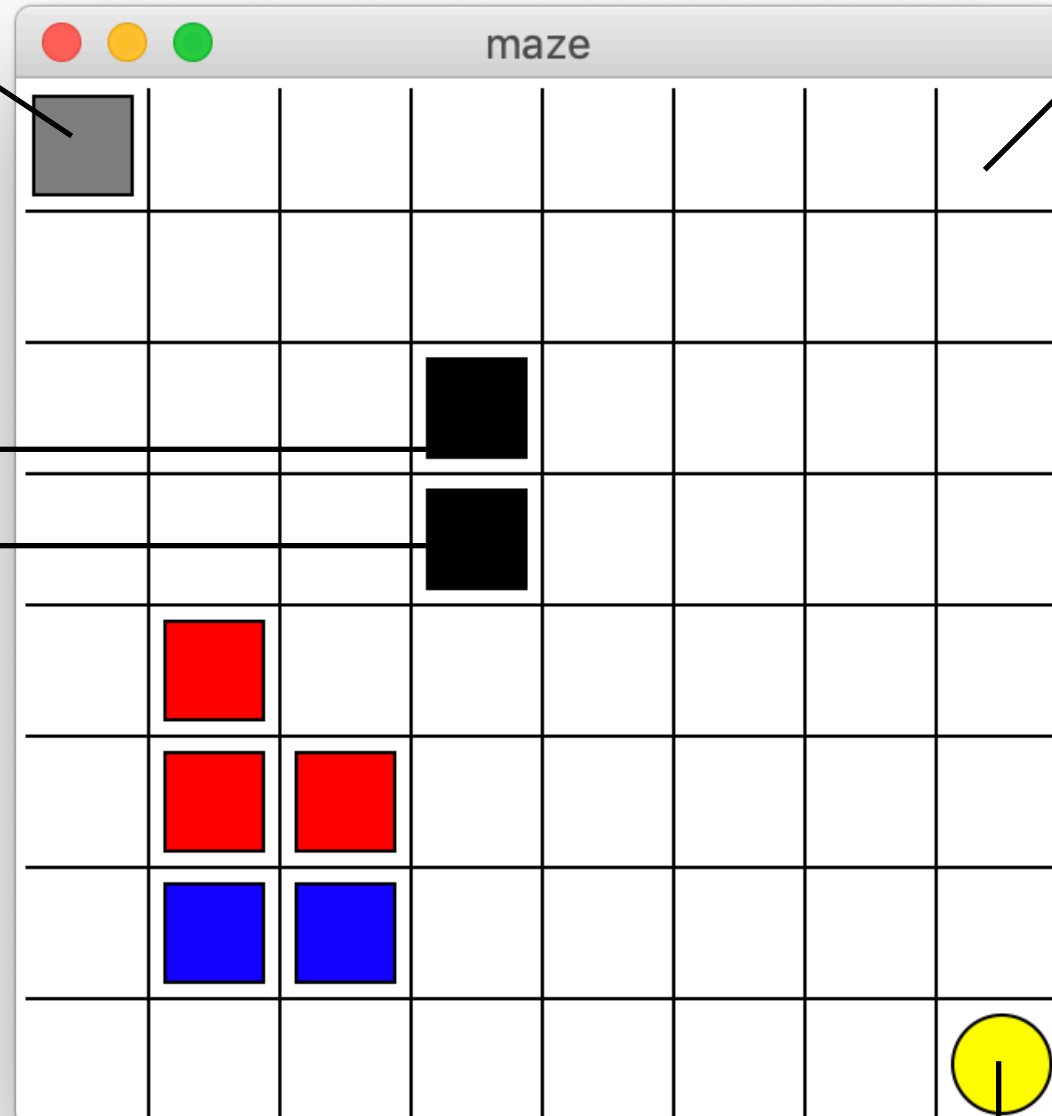
Any question?

Q - Learning

Location of UGV
Action = {up, down, left, right}

Location of each point is state
Reward = -1

Charging region of first IoT device (harvested power at each IoT)
if visited for the first time:
Reward = 10
else:
Reward = -1



Learning rate = 0.5
Reward_decay = 0.5
Epsilon = 0.9

Location of goal
*Reward = 10 + 5*x (x is the number of IoT devices charged)*

Environment

Continuous Charging model

Low probability for nearby arms

Arm

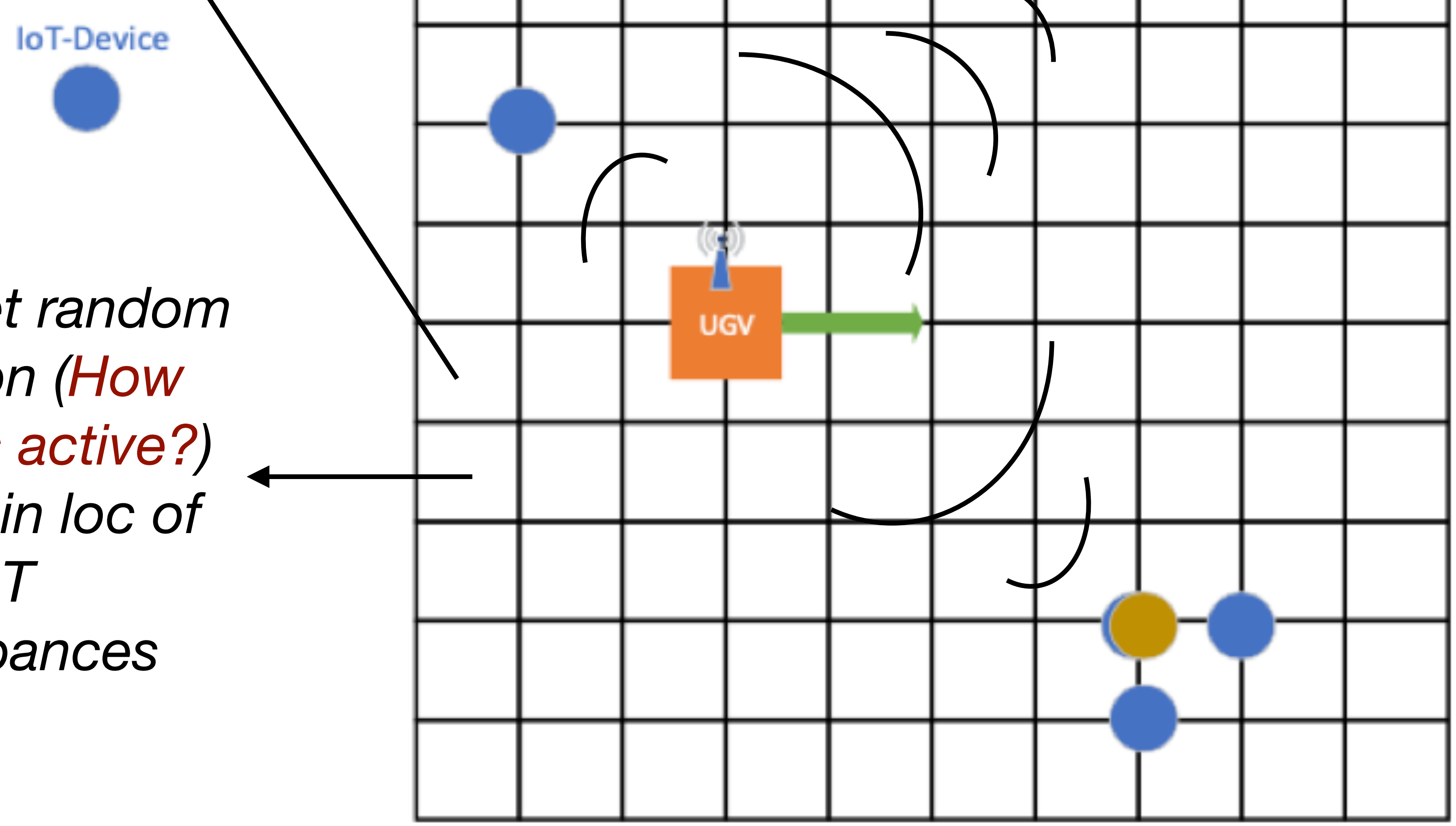


Figure 1 Graphical abstraction of the problem

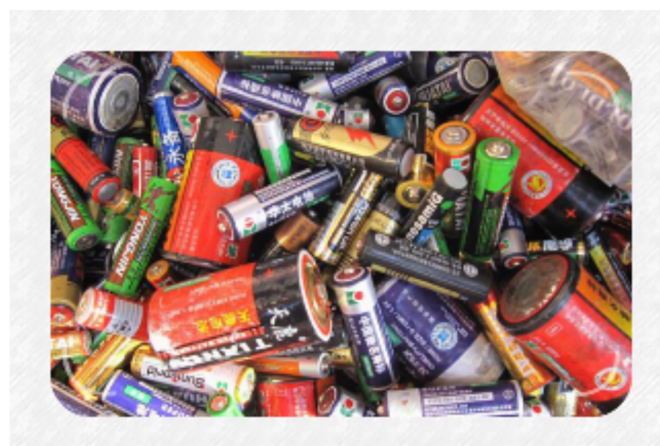
UCB algorithm to choose the arm.

If chosen get random observation (How many users active?)

- *Change in loc of IoT*
- *Disturbances*

Multi-armed Bandit with correlation

IoT Net = Tiny Size + Huge number



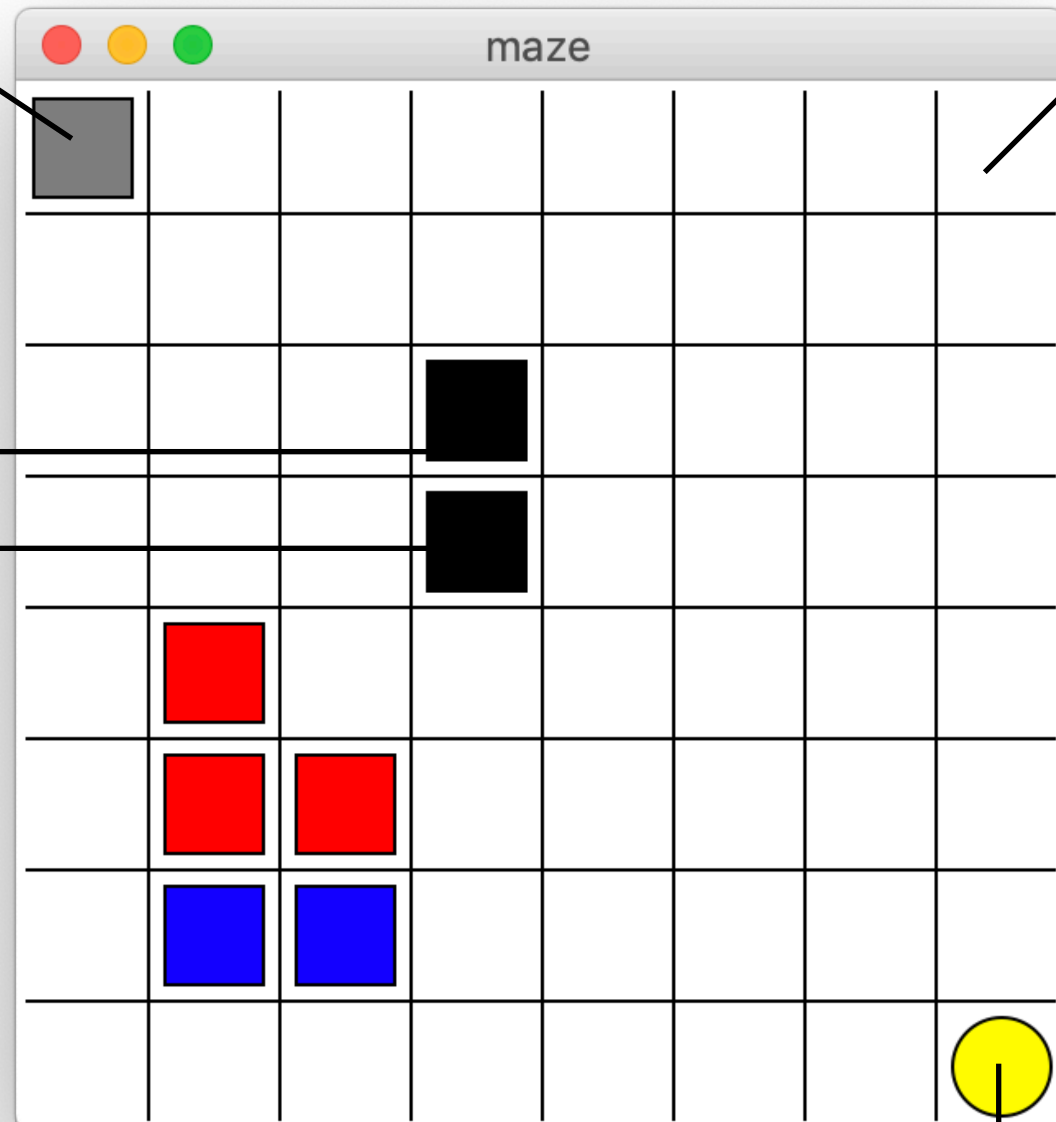
How to provide energy?

Q - Learning

Location of UGV
Action = {up, down, left, right}

Location of each point is state
Reward = -1

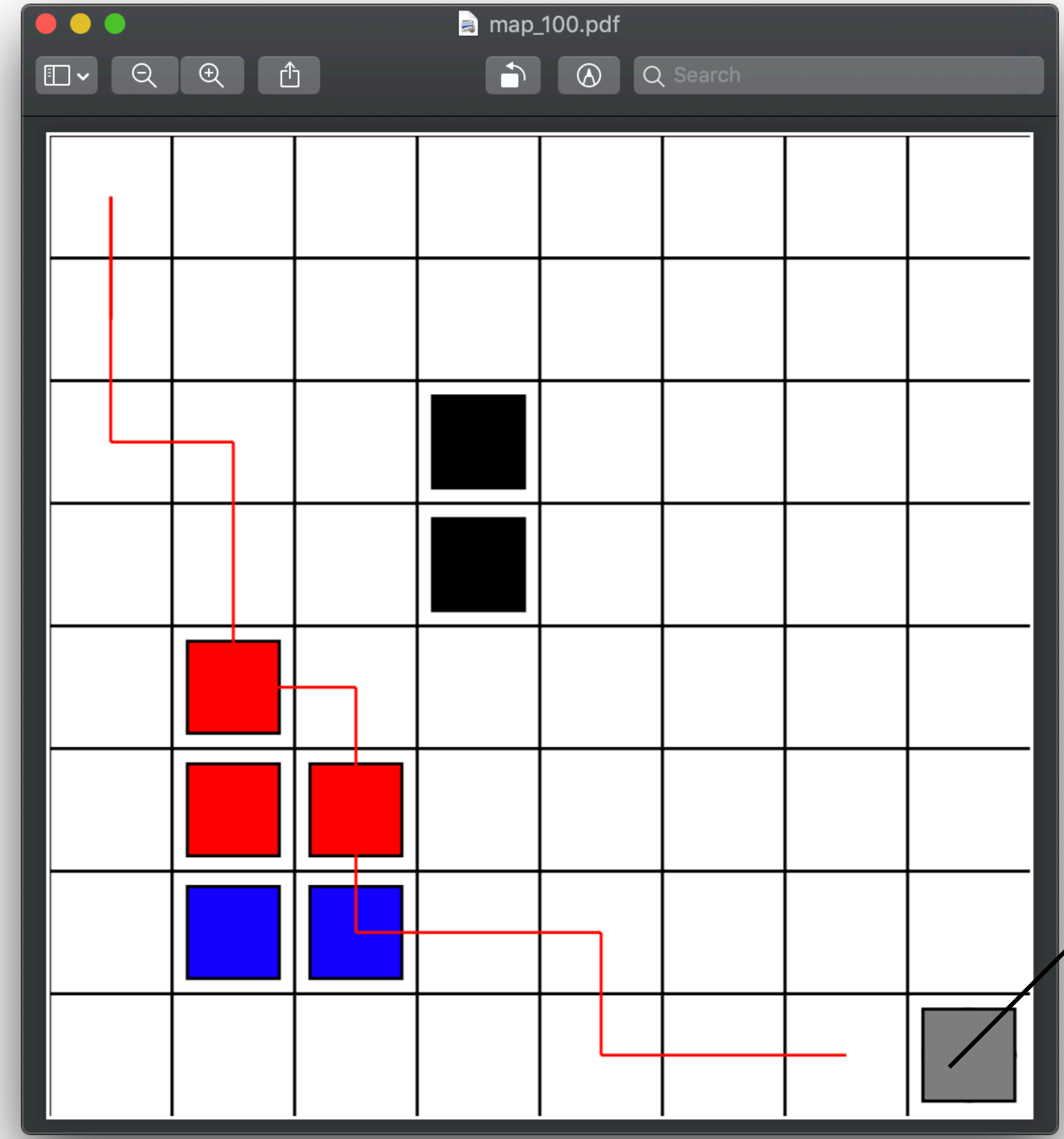
Charging region of first IoT device (harvested power at each IoT)
if visited for the first time:
Reward = 10
else:
Reward = -1



Location of goal
*Reward = 10 + 5*x (x is the number of IoT devices charged)*

Environment

Q-Learning



Reward at goal is too high!

*Reward = 100+ 5*x (x is the number of IoT devices charged)*

Q-Learning