Investigation and Experiments of Exploration Strategies in a Non-stationary Grid World

Yuhan Ye [1]

Abstract

Within the reinforcement learning framework, the exploration–exploitation dilemma has remained unsolved. The agent could either explore in order to improve the policy in the future, which may result in higher rewards, or exploit in order to obtain the highest reward based on current knowledge. Some existing works have proposed many effective exploration strategies, and some experiments were conducted in a stationary environment. However, the stationary assumption on the environment is impractical and non-stationary environments are common in many real-world problems. This project includes investigation and experiments of different exploration strategies in a non-stationary grid world with refreshing obstacles. The strategies reported in the open literature, namely Greedy, ϵ-greedy, Decaying ϵ-greedy, Boltzmann distribution method, Pursuit method, Thompson sampling, Simulated Annealing, Count-based intrinsic reward exploration, and Upper-Confidence-Bound are implemented to study their performances in a non-stationary grid world. Under the experimental settings of this project, the Boltzmann distribution method can help the agent reach the goal within 60 steps in a 5x5 grid world with where obstacles are continually refreshed, and it also shows good performance in a 10x10 grid world. According to the experimental results, the Boltzmann distribution method outperforms other strategies in most cases and could be a preferred choice to try in a non-stationary environment.

1. Introduction

The trade-off between exploration and exploitation is one of the challenges that occur in reinforcement learning(RL) [1]. Exploration refers to the agent trying to discover and learn new features in the environment by choosing possible sub-optimal actions and exploitation refers to the agent obtaining the greatest possible reward by choosing the optimal action under the current policy based on past experience. Essentially, exploration is to get more information. The core of various exploration strategies is to obtain as much valuable information as possible with as little cost as possible. Without exploration, the agent will only choose the best option under the current situation and ignore other possible alternative options. Especially in a non-stationary environment, the performance of the agent will inevitably deteriorate over time.

There are some related studies on the comparison of exploration methods. For example, Guo et al. [2] compared boltzmann distribution method and Simulated Annealing to solve a 22x17 puzzle problem. Yogeswaran et al. [3] compared seven exploration strategies to let the agent manipulate pucks that are scattered in a 20x20 grid world. Julia et al. [4] compared seven exploration strategies to solve path planning in a 32x26 grid world.

However, the researches above are all limited to a stationary environment. In the traditional reinforcement learning setup, the agent is assumed to work in a stationary setting, with defined dynamics and rewards, [5]. Thus the problem of designing reinforcement learning algorithms for non-stationary environment models has received very little attention in the previous work [6], let alone comparison and research on exploration strategies in a non-stationary environment. However, the stationary assumption on the environment is very restrictive and impractical. Non-stationary environments, as understood literally, are environments that do not have a fixed state space, action space, and reward, thus the optimal policy is also changing over time. Non-stationary environments are common in many real-world problems such as self-driving control, path planning, traffic signal control, robotic applications, and in the context of lifelong learning systems [7]. Therefore, the research of reinforcement learning in non-stationary environments is very necessary and it could make reinforcement learning better fit the real life situations.

The main contribution of this paper is to investigate the effec-

[1]Department of Computer Engineering, University of Alberta, Alberta, Canada. Correspondence to: Yuhan Ye <yuhan1@ualberta.ca>.

tiveness of different exploration strategies used in the open literature to tackle the exploration and exploitation dilemma in a non-stationary environment with a Q-learning agent [8]. The investigated strategies include Greedy method, ϵ-greedy selection, Boltzmann distribution, Pursuit, Thompson Sampling, Simulated Annealing(SA), Count-based intrinsic reward exploration, and Upper-Confidence-Bound(UCB). The performances of the exploration strategies are determined by the reward received during the simulation process.

2. Background

2.1. Machine learning

The field of machine learning is concerned with the development and application of computer algorithms that improve with experience by the use of data, which can be seen as part of artificial intelligence [9]. The availability of a nearly limitless quantity of available data, combined with advances in algorithms and exponential increases in less expensive and more powerful computing power, has sparked unprecedented interest in machine learning. Machine learning algorithms are now widely used for large-scale classification, regression, clustering, and dimensionality reduction tasks involving high-dimensional input data [10]. As a result, machine learning has applications in many fields and has affected all aspects of daily life. For example, cutting-edge image recognition and detection methods, which are widely used in the preliminary processing of medical images [11], object detection in autonomous driving [12], video surveillance [13], and content recommendation [14], are powered by machine learning algorithms. It is not until the rapid development of deep learning technology that the accuracy of speech recognition has been greatly improved and the applications of speech recognition include voice dialing, call routing, interactive voice response, data entry and dictation, voice command and control, gaming, structured document creation, appliance control by voice, computer-aided language learning, content-based spoken audio search, etc. [15].

What is even more surprising and admirable is that machine learning has already surpassed human-level performance in many fields. The most famous example is that AlphaGo [16] developed by DeepMind defeated world champion Lee Sedol 4:1 in 2015, and the subsequent version AlphaZero defeated the strongest human professional players 60–0 in online games in January 2017 [17].

In general, machine learning techniques can be divided into three paradigms [18], as discussed next.

2.1.1. Supervised learning

Machine learning algorithms use the same set of features to represent each instance in the data set. The features may be continuous, categorical, or binary in nature. When instances are provided with known labels by a knowledgeable external supervisor, the learning is referred to as supervised learning [19]. The purpose of supervised learning is for the system to extrapolate or generalize its response in order to operate correctly in the absence of situations in the training set [20]. Supervised learning is studied in most current research in the field of machine learning.

2.1.2. Unsupervised learning

The goal of unsupervised learning is to find hidden patterns in unlabeled data. Some common algorithms used in unsupervised learning include Principle component analysis (PCA) [21], Independent component analysis (IDP) [22], K-means clustering [23], Hierarchical clustering [24], DBSCAN [25], Isolation forest [26], and etc. In recent years, Generative Adversarial Networks (GAN) [27] have been proposed and considered to be a very promising direction for generative modelling. It derived many of its variants such as Conditional Generative Adversarial Nets(CGAN) [28], Wasserstein Generative Adversarial Nets(WGAN) [29], and Deep Convolutional Generative Adversarial Networks(DCGAN) [30]. GAN has achieved unprecedented success in the field of image generation and video generation. Variational Autoencoder (VAE) is one of the cutting-edge methods of learning latent space variables in an unsupervised manner.

2.1.3. Reinforcement learning

The terms supervised learning and unsupervised learning seem to categorize machine learning paradigms exhaustively, but they do not. Reinforcement learning is a branch of machine learning that studies how intelligent agents map situations to actions in a given environment to maximize a numerical reward signal [31]. It differs from supervised learning and unsupervised learning in that it emphasizes learning by an agent from direct interaction with its environment, without requiring outstanding supervision or complete models of the environment. The learner is not told which actions to take; instead, it must discover which actions result in the most reward by trial and error. Usually, actions may affect not only the immediate reward signal but also all subsequent rewards in the following situations, which is called delayed reward [20]. Reinforcement learning (RL) is widely used in a variety of fields, including industrial automation, time sequence prediction, robot soccer competitions, and more [3]. Reinforcement learning is the main technique used in this project. More relevant knowledge is discussed in the following sections.

2.2. Markov Decision Process

For most Reinforcement learning problems, the agent interacts with the environment which is given by a Markov Decision Process (MDP) [32]. The MDP provides a mathematical framework for modeling decision making in situations and abstracts the issue of goal-directed learning from interaction [33], thus it is a mathematically idealized form of reinforcement learning problem [20]. In each time step t, the agent is present in a state S_t and takes an action A_t based on its policy $\pi : \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$. Depending on the action and state, it gets a reward R_{t+1} and goes to the next state S_{t+1}, given by the environment transition dynamics $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$. It is assumed that the Markov property is present throughout the interaction between the agent and the environment. The return G_t can be defined as the sum of discounted future rewards i.e.

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1},$$

where γ is the discount factor. The agent seeks to maximize the expected return under the policy π.

The state value function denoted $v_\pi(s)$, is defined as the expected return given the agent's state s and following the policy π. Similarly, the action-value function denoted $Q_\pi(s, a)$, is defined as the expected return when given the agent's state s, action a and following the policy π. They are formally defined by

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s] \\ Q_\pi(s, a) &= \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a] \end{aligned} \tag{1}$$

Two problems of interest in reinforcement learning are the problems of prediction and control. Prediction seeks to accurately estimate the value functions under a policy while control seeks to find a (near-)optimal policy that the agent can follow to maximize its return. For the purpose of this paper, a control algorithm called Q-learning [8] is investigated and is discussed in the next section.

2.3. Q-learning

Q-learning was proposed by Watkins in 1989, with proof that the algorithm converges to the optimum action-values as long as all actions in all states can be repeatedly sampled, and action-values $Q(s, a)$ can be discretely represented [8]. With a fixed finite MDP and discount factor γ, let the agent be in state s and it takes action a to get a reward r and goes to the next state s'. If the next action is given by a', we have the following update equation of the action values:

$$Q(s, a) \leftarrow Q(s, a) + \alpha(r + \gamma \max_{a'} Q(s', a') - Q(s, a)), \tag{2}$$

where α is the learning rate and $\alpha \in [0, 1)$. In many problems of interest, the state space can be extremely large. Under such conditions, state aggregation or tile coding techniques could help define the feature representation. Another way is that the value functions are approximated as a parameterized function of the state. Such parameterization can be done using a neural network with states as input or with linear function approximation of features derived from the states.

In Deep Reinforcement Learning, such an approximation of action values, $Q(s, a; \theta)$ is given by a Q-network, which is a deep neural network with parameters θ. The input of the network is the states and the network gives action values as its output. Such a network is trained by the following loss function L_i which changes at every iteration i:

$$L_i(\theta_i) = \mathbb{E}_{(s,a)\sim\rho(\cdot)}[target_i - Q(s, a; \theta_i)]^2 \tag{3}$$

where ρ is the behaviour distribution which is the probability distribution over states and actions given by the behaviour policy b of the agent, also called the state visitation distribution under behaviour policy b and the target at each iteration i is $\mathbb{E}_{(s'\sim\mathcal{P}(\cdot|s,a))}[r + \gamma \max_{a'} Q(s', a', \theta_{i-1})]$. Such Q-networks are popularly known as DQNs [34].

However, such networks are unstable or even to diverge due to various causes such as the correlations between the sequence of observations, changing data distribution because of the update and the correlations between action values and target values [35]. These issues are mitigated using experience replay (ER) buffers and target networks [35]. Experience replay buffers are memory buffers which store the agent's experiences $e_t = (s_t, a_t, r_{t+1}, s_{t+1})$ as a dataset $D = (e_1, e_2, ...)$. The training of the Q-network is performed with random minibatches of data from this experience replay buffer. Then the parameters θ are updated by

$$\theta_i \leftarrow \theta_i + \alpha \nabla_{\theta_i} L_i \tag{4}$$

$$\begin{aligned} L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim U(D)} \Big[& r + \gamma \max_{a'} Q(s', a', \theta_i^-) \\ & - Q(s, a; \theta_i) \Big]^2 \end{aligned} \tag{5}$$

$$\begin{aligned} \nabla_{\theta_i} L_i = \mathop{\mathbb{E}}_{(s,a,r,s')\sim U(D)} \Big[& (r + \gamma \max_{a'} Q(s', a'; \theta_i^-) \\ & - Q(s, a; \theta_i)) \nabla_{\theta_i} Q(s, a; \theta_i) \Big] \end{aligned} \tag{6}$$

where α is the learning rate, $U(D)$ is a uniform distribution over the experiences stored in the ER buffer. Here uniform distribution is not necessary. Other work such as Prioritized Experience Replay [36] and Distributed Prioritized Experience Replay [37] have proposed different sampling methods from ER buffer and achieved better performance. θ_i^- are parameters of the target network. The target network is essentially a copy of the Q-network and it updates its parameters by copying from the Q-network periodically with a hyper-parameter T. The parameters of the target network keep constant between the updates.

3. Exploration Strategies

In this section, several exploration strategies that try to balance exploration and exploitation are presented.

3.1. Greedy

The simplest strategy is greedy selection which follows the problem-solving heuristic of making the locally optimal choice at each step. Greedy selection is a pure exploitation approach and is the commonly used learning policy that is associated with the vanilla Q-learning [38]. The strategy could be formally defined by

$$\pi(a \mid s) = \underset{a}{argmax}\, Q(s, a) \tag{7}$$

Following the greedy strategy, the agent selects action based on the highest action value.

3.2. ϵ-greedy

On the basis of greedy selection, a simple approach that comes up with to encourage exploration is ϵ-greedy. The percentage of time the agent deviates from greedy behaviour is regulated by a single parameter $\epsilon \in [0, 1]$. When the agent chooses which action to take, there is a ϵ possibility of exploration, and a $1 - \epsilon$ possibility of exploitation. The strategy could be formally defined by

$$\pi(a \mid s) = \begin{cases} rand(A), & rand(0,1) < \epsilon \\ \underset{a}{argmax}\, Q(s, a), & otherwise \end{cases} \tag{8}$$

It can be easily derived that the higher the value of ϵ, the more frequently the agent will choose exploration. And the lower the value of ϵ, the more frequently the agent will choose exploitation. One disadvantage of ϵ-greedy action selection is that it forces to try non-greedy actions, but chooses them indiscriminately, and does not have a preference for those that are nearly greedy or particularly uncertain, which is probably not reasonable for an exploration strategy [20]. Due to the randomness, it is possible that the agent ends up exploring a bad action which has been confirmed in the past [39].

Another problem for the general ϵ-greedy algorithm is that the random sampling parameterϵ is a constant value. So when the exploration is sufficient, the action selection strategy will still have the same degree of exploration, which is actually a waste since we hope that the agent can make full use of the past experience to get the most benefit after the full exploration, and the subsequent exploration will no longer be meaningful. An intuitive method to solve this problem is to make ϵ decrease over time and the method is called decaying ϵ-greedy. A common implementation is that the ϵ decreases exponentially over time, that is, at each time step t, ϵ is updated by

$$\epsilon = \epsilon * decay \tag{9}$$

Since ϵ-greedy is easy to implement, it is one of the most popular exploration strategies [40] and many works such as the famous vanilla DQN[34] took this approach as the learning policy.

3.3. Boltzmann Distribution

Boltzmann distribution, also called soft-max distribution or Gibbs distribution, is originally a probability distribution or measure that describes the likelihood of a system being in a certain state as a function of the energy and temperature of the system [41]-[42]. In the reinforcement learning setting, Boltzmann distribution provides a probability distribution based on action values and the probability that an action is selected depends on this distribution. The strategy could be formally defined by

$$\pi(a \mid s) = \frac{e^{Q(s,a)/T}}{\sum_b e^{Q(s,b)/T}} \tag{10}$$

where T is a hyper-parameter called temperature. Exploration increases as T increases. Similar to the ϵ-greedy method, the parameter T can also decrease over time. When exploration is sufficient, a very small T can allow the agent to make full use of past experience and conduct exploitation.

3.4. Pursuit

Pursuit Algorithms (PA) was first introduced by Thathachar et al. [43]. It utilizes the maximum likelihood method for estimating reward probabilities and maintain an explicit strategy. The method for the most basic pursuit algorithm is as follows. To begin, in a certain state s, the probability of each action being chosen is set to be uniform, that is $p_a(0) = 1/N_A$ where N_A is the number of actions. Then at each time step t, the probability of each action is updated by

$$p_a(t) = \begin{cases} p_a(t) + \alpha(1 - p_a(t)), & a = \underset{a}{argmax}\, Q(s, a) \\ p_a(t) + \alpha(0 - p_a(t)), & otherwise \end{cases} \tag{11}$$

where $\alpha \in (0, 1)$ is the learning rate. It can be seen that when an action is rewarded and has a higher Q value, the probability of choosing this action will increase, while other actions still have a probability of being selected, which ensures that the agent can keep exploring. It can also be seen that one disadvantage of the pursuit algorithm is that the performance is probably worse in a problem with a large action space. Because the probability of each action being selected will be very small in this case, more time will be required to ensure that each action is taken enough times and find out the optimal action.

3.5. Thompson Sampling

Thompson Sampling, also called posterior sampling was proposed by Thompson in 1933 [44]. The first application to MDP was in 2000 [45] and a thorough modern treatment of Thompson sampling is provided by Russo et al. [46]. At each time step, The expected action value $Q(a)$ are sampled from a prior distribution such as Beta distribution $Beta(\alpha, \beta)$ for each action. The agent selects greedily from these action values. Then, the agent updates the parameters of the distribution based on the reward obtained in the environment. These updates help improve the distribution and after a large amount of trials, the distribution could give expected action values that are close to the true action values. Thompson sampling strategy could be formally defined by

$$\begin{aligned} \pi\left(a \mid h_t\right) &= \mathbb{P}\left[Q(a) > Q\left(a'\right), \forall a' \neq a \mid h_t\right] \\ &= \mathbb{E}_{\mathcal{R}|h_t}\left[\mathbb{I}\left(a = \arg\max_{a \in \mathcal{A}} Q(a)\right)\right] \\ = \int \left[\mathbb{I}\left(\mathbb{E}\left(r \mid a^*, \theta\right) = \max_{a \in A} \mathbb{E}(r \mid a, \theta)\right)\right. & \left. P(\theta \mid h_t) d\theta \right. \end{aligned} \tag{12}$$

where h_t denotes the history at time step t, and $\mathbb{I}$ denotes the indicator function. For $Beta$ distribution and for a certain action a_i, the parameters α_i and β_i are simply updated by

$$\begin{aligned} \alpha_i &\leftarrow \alpha_i + r_t \\ \beta_i &\leftarrow \beta_i + 1 - r_t \end{aligned} \tag{13}$$

and each action in each state follows its own $Beta$ distribution.

3.6. Simulated Annealing

Simulated Annealing is a probabilistic method for approximating global optimization for an optimization problem in a large search space. Guo et al. introduced SA strategy into Q learning and achieved good performance [2]. The SA-Q-learning algorithm is as follows: First, all action values are initialized arbitrarily. Then at each time step t, in a randomly chosen state s, the agent selects an action a_r arbitrarily and selects an action a_p according to the policy. Then a random value $\epsilon \in (0, 1)$ is generated. If

$$\epsilon < exp((Q(s, a_r) - Q(s, a_p))/T), \tag{14}$$

where T is a hyper-parameter called temperature, then a_r is finally selected, else a_p is finally selected. If the policy is a greedy policy, then the strategy could be formally defined by

$$\pi(a \mid s) = \begin{cases} \text{rand}\left(A\right), & condition(14) \\ \text{argmax}_a\, \mathcal{Q}(s, a), & otherwise \end{cases} \tag{15}$$

Like ϵ-greedy exploration strategy, the SA method encourages the agent to explore by increasing the chance of selecting actions other than the greedy action under current action values. Instead of giving a probability distribution to select action like Boltzmann selection strategy, the SA method compares the value of a small ϵ value and the exponential value of the difference between a random action value and a max action value over parameter T and if ϵ is smaller, then the agent explores. It can be derived that higher T or lower ϵ leads to more exploration.

3.7. Intrinsic Reward-Based Exploration

As mentioned in the MDP section that under the problem definition of MDP, the environment provides feedback by rewarding the agent and the goal of the agent is to maximize such a cumulative return. Thus the return provided by the environment is an important signal for the agent to learn. Such a kind of return could be called an extrinsic reward and be denoted by r_t^e. An intrinsic reward r_t^i by the agent itself can be added to encourage extra exploration. Essentially, the method based on intrinsic reward can be regarded as a way of interval estimation, and the intrinsic rewards are based on the potential of the action to be the optimal action.

One intuitive method is to count the number of times that a state-action pair has been encountered, and uses the count value to calculate the intrinsic reward. For states and actions that have been experimented with many times, the intrinsic reward is small. On the contrary, for rarely visited states and actions, the intrinsic reward is large. This method is called count-based exploration.

The count-based intrinsic reward (CBIR) exploration algorithms are optimistic about unexplored states and actions, thus more intrinsic reward is given to them. Wiering et al. [47] classified such exploration strategies that use special exploration-specific knowledge to guide the search through alternative policies into directed exploration and classified exploration strategies, such as $\epsilon-$greedy strategy, that use randomized action selection methods to guess useful experiences into undirected exploration strategies. Wiering et al. proposed two count-based intrinsic rewards. The first one is recency-based, that is, the agent selects the action which

has been selected least recently. It can be formally defined by

$$r^i(s_t, a_t, *) := \frac{-t}{K_T} \tag{16}$$

where t is the time step and K_T denotes a scaling constant. The second one is frequency-based, that is, the agent selects the action which has been executed least frequently. It can be formally defined by

$$r^i(s_t, a_t, *) := \frac{-C_{s_t}(a_t)}{K_C} \tag{17}$$

where K_C is a scaling constant.

Another simple and common choice of the count-based intrinsic reward is $r_t^i = N(s_t, a_t)^{-1/2}$ [48]. Then the strategy could be formally defined by

$$\pi(a \mid s) = \underset{a}{argmax}[Q_t(s,a) + \alpha N(s_t, a_t)^{-1/2}] \tag{18}$$

where α is the learning rate.

3.8. Upper-Confidence-Bound

The UCB algorithms are also a kind of count-based exploration. Considering the disadvantage of ϵ-greedy, it is more reasonable and preferable to select non-greedy actions based on their potential for being optimal, while considering the closeness of their estimated values to the maximum estimated values and the uncertainty of these estimated values [20]. The UCB methods quantify unexplored states and actions' potential of being optimal by measuring the uncertainty or variance in the estimate of action's value $U_t(s,a)$ so that

$$Q(s,a) \leq Q_t(s,a) + U_t(s,a) \tag{19}$$

with high probability. Here $Q(s,a)$ denotes the true action value for an agent choosing action a in the state s, $Q_t(s,a)$ denotes the obtained action value at time step t. The inequality can be established when $U_t(s,a)$ is set to some specific values, which could be proved by Hoeffding's Inequality [49] The uncertainty term could be viewed as a bonus for state-action pairs that have a high potential of being optimal or rarely selected.

An early UCB algorithm UCB1 was proposed by Auer et al.[50]. The strategy could be formally defined by

$$\pi(a \mid s) = \underset{a}{argmax}[Q_t(s,a) + c\sqrt{2\ln t / N_t(s,a)}] \tag{20}$$

where $N_t(s,a)$ denotes the number of times the agent chooses action a in the state s prior to time t, and the number $c > 0$ is the parameter that refers to the confidence level and controls the degree of exploration. If $N_t(s,a) = 0$, then the action is greedily selected. It can be easily derived that in the state s, each time the action a is selected, as $N_t(s,a)$ increases, the measure of uncertainty of the action a in the state s will decrease. On the other hand, each time an action other than action a is selected, as $\ln t$ increases, the measure of uncertainty of the action a in the state s will increase. With the trial and error of the agent over time, the upper bound will gradually approach the true action value. The $\ln$ operator indicates the increase of the numerator term gets smaller over time. All actions will be selected, but actions with lower action value estimate and actions that have already been selected frequently will be selected with lower frequency [20].

The UCB approach is also widely used in industry. The Monte Carlo Tree Sort (MCTS) algorithm that applied in the famous AlphaGo [17] uses UCB1 as its first step to estimate action values.

4. Experiments

Experiments were performed to test different exploration strategies as discussed above. Detailed experimental setup including environments tested, as well as detailed parameter settings for the network and each exploration strategy, will be presented in this section.

4.1. Environment

All experiments were conducted in two environments. Both environments are grid worlds, and their sizes are 5x5 and 10x10 respectively. The aim of setting up a smaller 5x5 environment is to evaluate solutions in a smaller environment first with a faster speed and eliminate strategies that aren't working. A parameter N_{obs} controls the number of obstacles in the environment and a parameter T_r controls the frequency of the environment changes, that is, after T_r steps, the environment will clear the current obstacles and generate new obstacles at random locations. Although obstacles are randomly generated, there are still several limitations. First, obstacles will not appear at the current position of the agent and the endpoint. Second, a breadth-first search(BFS) algorithm is applied to ensure that there is always at least one feasible path from the current position of the agent to the endpoint during the generation of obstacles.

The initial position of the agent is in the upper left corner of the grid world, and the destination is in the lower right corner. At each time step t, the agent chooses an action according to its exploration strategy. The agent can move in four directions: up, left, down, right. If the place the agent moves to is an obstacle or the boundary of the grid world, the agent will stay in the original place. The reward for each time step t is -1 to encourage the agent to find the target as soon as possible. And because of this reward setting, the final obtained reward in each episode is actually the negative value of the number of steps plus one(the step to the end

will not be rewarded).

4.2. Network

A two-layered fully connected neural network with 32 hidden units in each layer and ReLU non-linearity is used as the Q-network and target network. The neural network weights are initialized using the Xavier initialization [51] and the initialization of biases are from a standard normal distribution. The size of the experience replay buffer is 4000 and the network is trained with a learning rate of 0.001 and with batches of data from the buffer of size 32. Adam [52] is the optimizer with $tolerance = 10^{-8}, \beta_1 = 0.99$ and $\beta_2 = 0.999$. The target network is updated after every environment step.

4.3. Exploration strategies settings

4.3.1. GREEDY

There is no hyperparameter for Greedy strategy.

4.3.2. ϵ-GREEDY

ϵ is set to 0.1.

4.3.3. DECAYING ϵ-GREEDY

Initial ϵ_i is set to 0.5, $decay$ is set to 0.995, and the final ϵ_f is set to 0.1.

4.3.4. BOLTZMANN DISTRIBUTION

The initial temperature T_i is set to 0.5, $decay$ is set to 0.9999, and the final T_f is set to 0.001.

4.3.5. SIMULATED ANNEALING

The temperature parameter T of the SA algorithm follows the Boltzmann Distribution strategy's setting.

4.3.6. PURSUIT

The learning rate α is set to 0.1.

4.3.7. THOMPSON SAMPLING

$Beta$ distribution is used and a normalized action value is used as the reward to update parameters α and β of $Beta$ distribution.

4.3.8. SIMPLE COUNT BASED REWARD EXPLORATION

The final reward to update the policy is a normalized q value $\bar{Q}(s_t, a_t)$ plus $r_t^i = N(s_t, a_t + 0.01)^{-1/2}$.

4.3.9. UPPER CONFIDENCE BOUND

The final reward to update the policy is a normalized q value $\bar{Q}(s_t, a_t)$ plus $\sqrt{2 \ln t / N_t(s, a)}$.

4.4. Experimental Settings

In the 5x5 grid world environment, the performance of different exploration strategies is tested when N_{obs} is set to 0,5,10,15 respectively and the environment refresh parameter T_r is set to 0,10,100 respectively. In the 10x10 grid world environment, N_{obs} is set to 10,20,30,40,50,60 respectively and the environment refresh parameter T is set to 10,100 respectively. The setting of N_{obs} to 0 and T_r to 0 in a 5x5 grid world is to test the performance of different exploration strategies in an obstacle-free environment and a stationary environment. It is expected to see whether each strategy will perform differently if the update does not occur, the update occurs within one episode, and the update occurs within several episodes.

The number of runs is set to 20 and the number of episodes is set to 100. The maximum number of steps for each episode is set to 1000. Random behaviours occur during the generation of the environment, action selection and network initialization. The value of the seed for the random number generator is the number of the run. All experimental results are derived from the mean and variance of reward obtained over 20 runs.

5. Result

Result graphs are obtained rewards versus number of episodes averaged over 20 runs for each exploration strategy. In order to evaluate the results, in addition to the intuitive judgment from the graph, the average value of the rewards obtained in the last 20 episodes will also be considered. Results of experiments that T_r =100 are shown in the Appendix.

5.1. Results of Experiments in the 5x5 Grid World

Fig. 1 shows obtained rewards in a 5x5 grid world with N_{obs} = 0,5,10,15 and T_r = 0.

Table 1 and 2 show the average value of the rewards obtained in the last 20 episodes for each exploration strategy under this experimental setting. It can be concluded that when N_{obs} = 0, Greedy method and Count-based intrinsic reward method reach the optimal value. Since ϵ-greedy and decaying ϵ-greedy will always maintain a probability of exploration in the end, thus they cannot converge to the optimal solution. Also, Boltzmann distribution method, and UCB reach near optimal values. It can be seen from Fig. 1(b) that the pursuit algorithm is unstable during the training process, but it eventually converges to a near op-

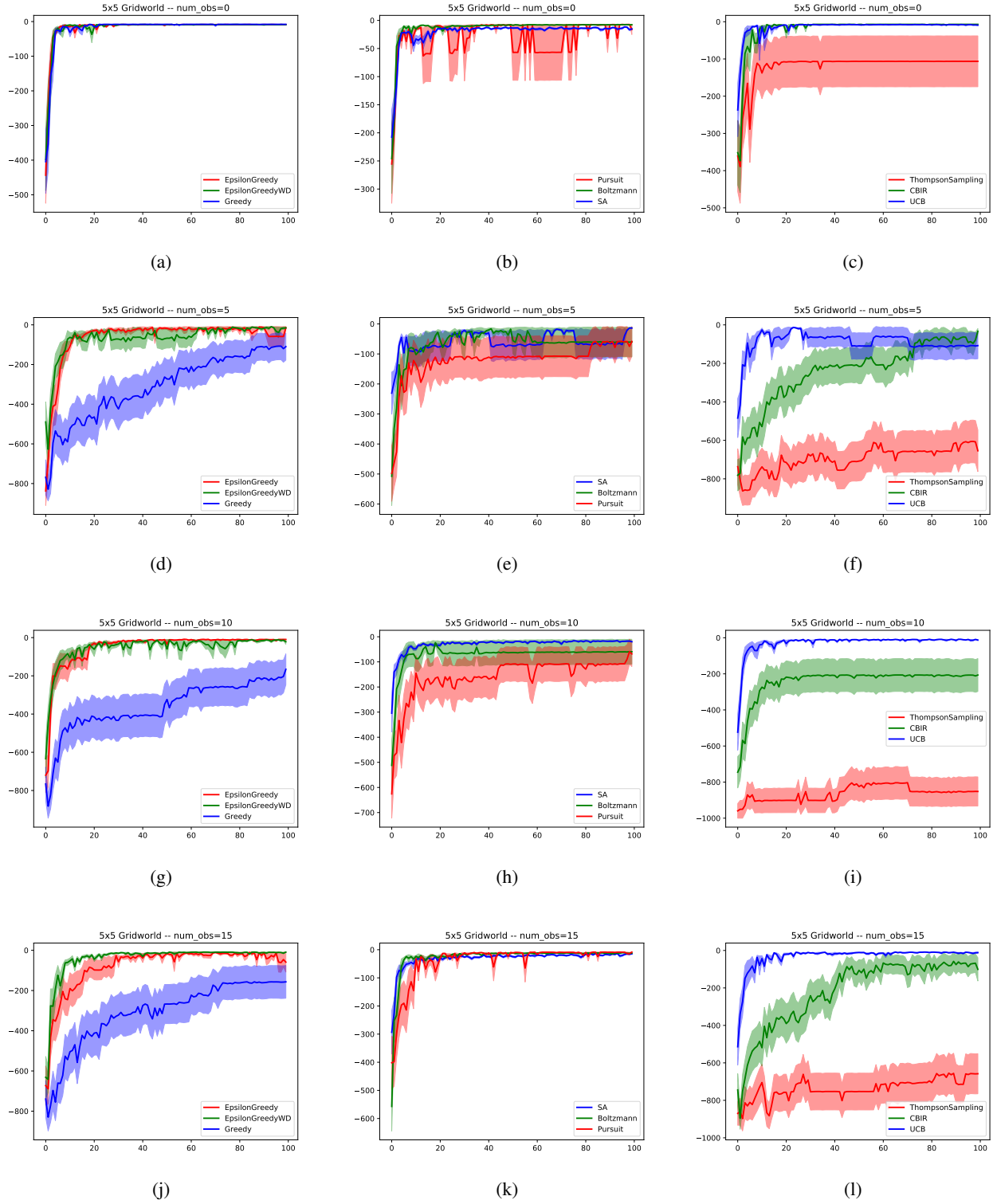

Figure 1. Obtained rewards vs number of episodes in a 5x5 grid world with N_{obs} = 0,5,10,15 and T_r = 0, averaged over 20 runs for each exploration strategy. (a)(d)(g)(j) compares the performance of ϵ-greedy, decaying ϵ-greedy, and Greedy methods. (b)(e)(h)(k) compares the performance of Pursuit, Boltzmann distribution and Simulated Annealing methods. (c)(f)(i)(l) compares the performance of Thompson Sampling, Count-based intrinsic reward exploration and UCB method.

Table 1. Average rewards obtained in the last 20 episodes, when T_r=0, N_{obs}=0(left), 5(right) in a 5x5 grid world

Exploration strategy	Value
Greedy	-7.0
ϵ-greedy	-7.1
Decaying ϵ-greedy	-7.1
Pursuit	-7.8
Boltzmann distribution	-7.9
SA	-14.0
Thompson Sampling	-106
CBIR	-7.0
UCB	-8.8

Exploration strategy	Value
Greedy	-159.45
ϵ-greedy	-27.95
decaying ϵ-greedy	-17.9
Pursuit	-106.75
Boltzmann	-59.45
SA	-65.9
Thompson Sampling	-655.9
CBIR	-74.2
UCB	-106.35

Table 2. Average rewards obtained in the last 20 episodes, when T_r=0, N_{obs}=10(left), 15(right) in a 5x5 grid world

Exploration strategy	Value
Greedy	-258.85
ϵ-greedy	-9.1
decaying ϵ-greedy	-13.85
Pursuit	-128.1
Boltzmann	-59.45
SA	-15.25
Thompson Sampling	-856.25
CBIR	-205.6
UCB	-10.8

Exploration strategy	Value
Greedy	-161.25
ϵ-greedy	-17.4
decaying ϵ-greedy	-11.6
Pursuit	-8.5
Boltzmann	-10.95
SA	-15.25
Thompson Sampling	-705.2
CBIR	-107.0
UCB	-8.15

timal value. Simulated Annealing converges to an acceptable value and Thompson Sampling shows the worst performance. When N_{obs} = 5, the performance of each exploration strategy starts to differentiate. The performance of decaying ϵ-greedy outperforms other methods. When N_{obs} = 10,15, there is a significant improvement in the performance of UCB and SA methods and UCB outperforms other methods. Fig. 1(d)(g)(j) shows such a situation that greedy may fall into a local optimal solution due to the lack of exploration, which lowers its average performance. All exploration strategies except Thompson Sampling work in the stationary environment under this project's experimental settings.

Fig. 2 shows obtained rewards in a 5x5 grid world with N_{obs} = 5,10,15 and T_r = 10. Table 3 shows the average value of the rewards obtained in the last 20 episodes for each exploration strategy under this experimental setting. It can be concluded that all strategies except Thompson Sampling converge to a good result. When there are more obstacles, the average number of steps required to reach the goal is more, except that pursuit and Thompson Sampling perform better when N_{obs}=15 than when N_{obs}=10. the Boltzmann distribution method outperforms all other methods under this setting.

Fig. 5 shows obtained rewards in a 5x5 grid world with N_{obs} = 5,10,15 and T_r = 100. Table 6 shows the average value of the rewards obtained in the last 20 episodes for each exploration strategy under this experimental setting. It can be concluded that all strategies perform worse when T_r is 100 than when T_r is 10 and the number of steps required to reach the goal is generally doubled. SA performs the best when N_{obs} = 5,10 and Boltzmann distribution performs the best when N_{obs} = 15.

It can be seen from the graph that Greedy, Pursuit and CBIR take more episodes to converge. The performance of Thompson Sampling is poor in all settings, and it will not be tested in a 10x10 environment.

5.2. Results of Experiments in the 10x10 Grid World

Because the 10x10 environment has a larger state space, the agent needs more steps to reach the goal.

Fig. 3 and Fig. 4 show the obtained rewards in a 10x10 grid world with N_{obs} = 10,20,30,40,50,60 and T_r = 10. Table 4 and Table 5 show the average value of the rewards obtained in the last 20 episodes for each exploration strategy under this experimental setting. Each exploration strategy can converge to a result that the agent could reach the goal within 100 steps. The general trend is that each exploration strategy becomes less effective as obstacles increase. The same as count-based methods, UCB converges faster than CBIR that takes $r_t^i = N(s_t, a_t)^{-1/2}$ as the intrinsic reward, while they finally reach a similar result. The performance of Pursuit and SA method are unstable during training as

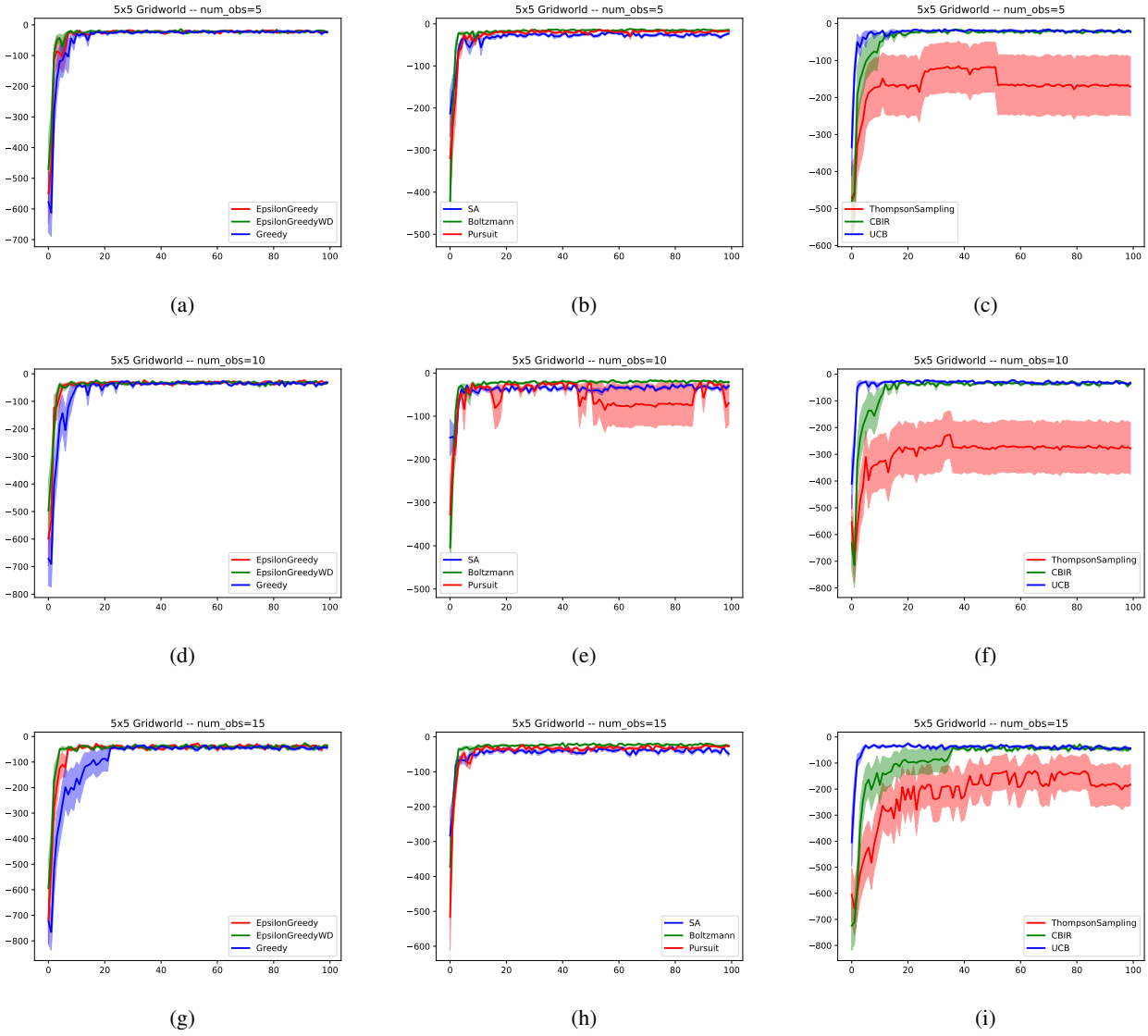

Figure 2. Obtained rewards vs number of episodes in a 5x5 grid world with N_{obs} = 5,10,15 and T_r = 10, averaged over 20 runs for each exploration strategy. (a)(d)(g) compares the performance of ϵ-greedy, decaying ϵ-greedy, and Greedy methods. (b)(e)(h) compares the performance of Pursuit, Boltzmann distribution and Simulated Annealing methods. (c)(f)(i) compares the performance of Thompson Sampling, Count-based intrinsic reward exploration and UCB method.

Table 3. Average rewards obtained in the last 20 episodes, when T_r=10, N_{obs}=5(left), 10(middle), 15(right) in a 5x5 grid world

Exploration strategy	Value
Greedy	-22.8
ϵ-greedy	-17.75
decaying ϵ-greedy	-24.65
Pursuit	-20.75
Boltzmann	-14.9
SA	-25.3
ThompsonSampling	-165.65
CBIR	-18.05
UCB	-22.8

Exploration strategy	Value
Greedy	-35.7
ϵ-greedy	-38.25
decaying ϵ-greedy	-32.25
Pursuit	-70.25
Boltzmann	-16.35
SA	-31.1
ThompsonSampling	-273.85
CBIR	-35.85
UCB	-38.05

Exploration strategy	Value
Greedy	-46.95
ϵ-greedy	-40.45
decaying ϵ-greedy	-42.7
Pursuit	-34.0
Boltzmann	-22.2
SA	-40.15
ThompsonSampling	-139.85
CBIR	-42.6
UCB	-43.2

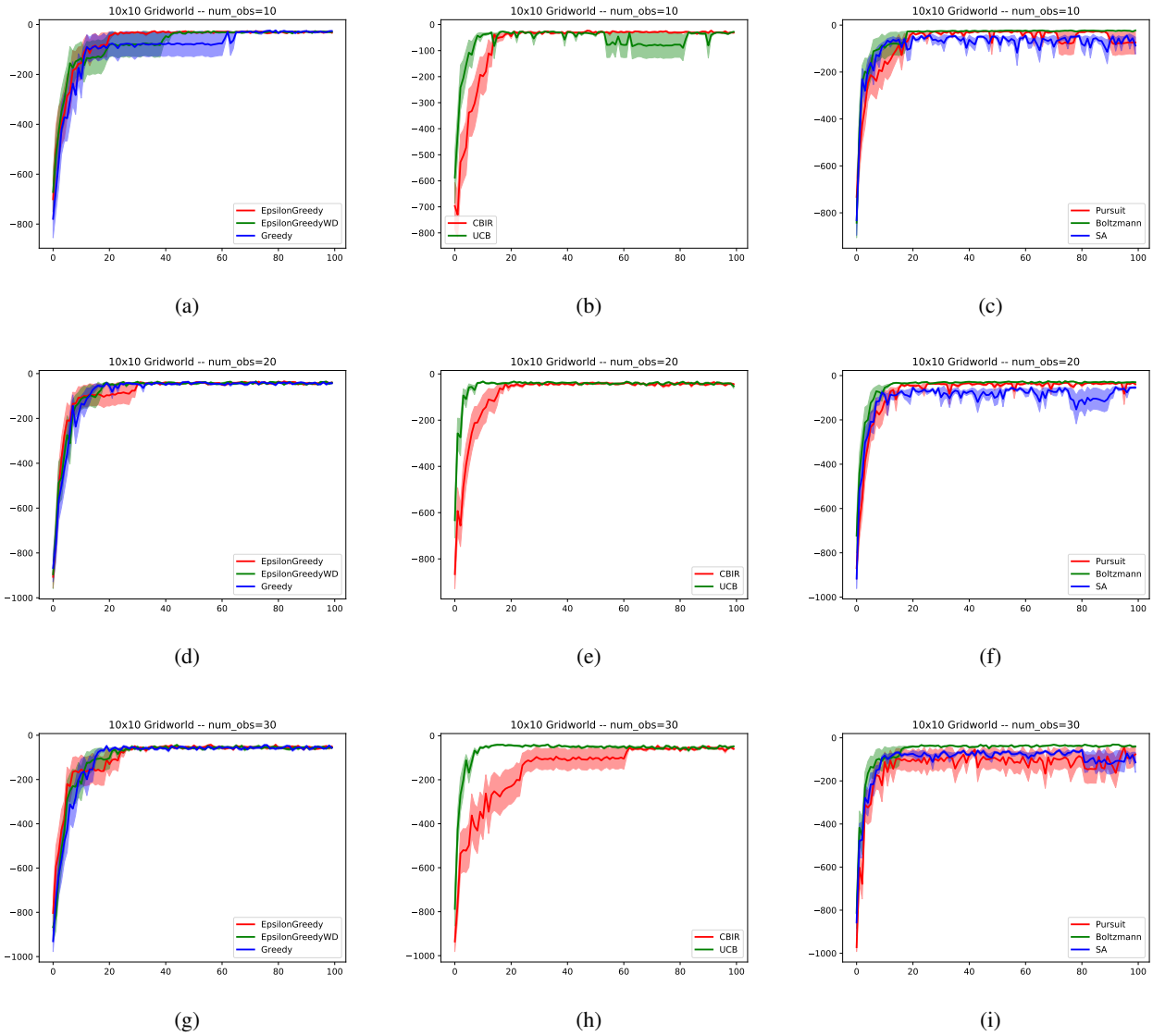

Figure 3. Obtained rewards vs number of episodes in a 10x10 grid world with N_{obs} = 10,20,30 and T_r = 10, averaged over 20 runs for each exploration strategy. (a)(d)(g) compares the performance of ϵ-greedy, decaying ϵ-greedy, and Greedy methods. (b)(e)(h) compares the performance of Count-based intrinsic reward exploration and UCB method. (c)(f)(i) compares the performance of Pursuit, Boltzmann distribution and Simulated Annealing methods.

Table 4. Average rewards obtained in the last 20 episodes, when T_r=10, N_{obs}=10(left), 20(middle), 30(right) in a 10x10 grid world

Exploration strategy	Value
Greedy	-33.15
ϵ-greedy	-31.1
decaying ϵ-greedy	-31.35
Pursuit	-28.6
Boltzmann	-22.1
SA	-79.5
CBIR	-29.85
UCB	-76.3

Exploration strategy	Value
Greedy	-38.2
ϵ-greedy	-34.85
decaying ϵ-greedy	-36.8
Pursuit	-43.1
Boltzmann	-33.75
SA	-107.9
CBIR	-45.8
UCB	-37.05

Exploration strategy	Value
Greedy	-55.2
ϵ-greedy	-55.55
decaying ϵ-greedy	-56.75
Pursuit	-91.4
Boltzmann	-37.25
SA	-55.0
CBIR	-67.25
UCB	-60.65

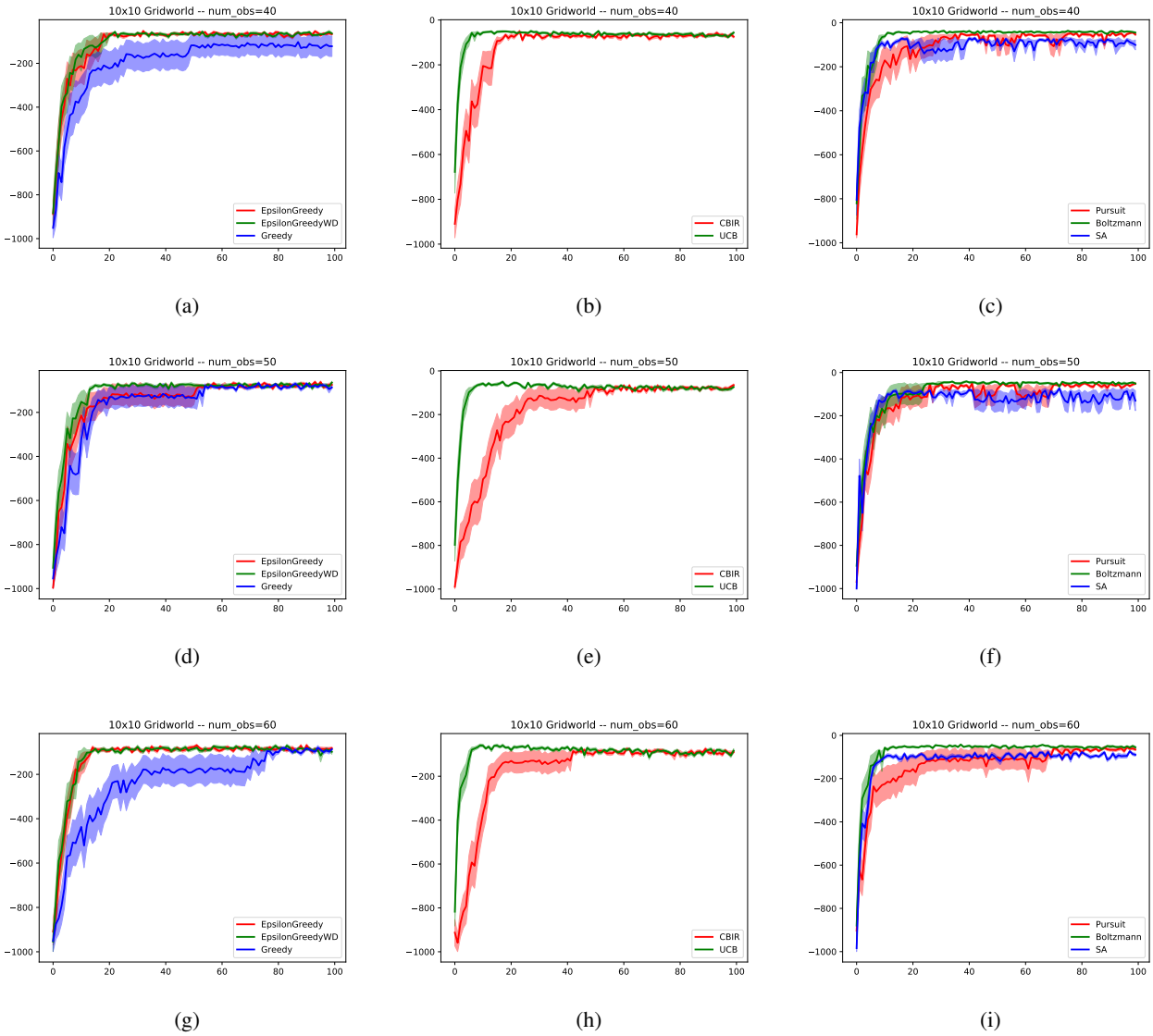

Figure 4. Obtained rewards vs number of episodes in a 10x10 grid world with N_{obs} = 40,50,60 and T_r = 10, averaged over 20 runs for each exploration strategy. (a)(d)(g) compares the performance of ϵ-greedy, decaying ϵ-greedy, and Greedy methods. (b)(e)(h) compares the performance of Count-based intrinsic reward exploration and UCB method. (c)(f)(i) compares the performance of Pursuit, Boltzmann distribution and Simulated Annealing methods.

Table 5. Average rewards obtained in the last 20 episodes, when T_r=10, N_{obs}=40(left), 50(middle), 60(right) in a 10x10 grid world

Exploration strategy	Value
Greedy	-113.3
ϵ-greedy	-57.4
decaying ϵ-greedy	-62.25
Pursuit	-50.95
Boltzmann	-41.45
SA	-94.85
CBIR	-65.05
UCB	-62.65

Exploration strategy	Value
Greedy	-84.25
ϵ-greedy	-76.7
decaying ϵ-greedy	-78.05
Pursuit	-54.1
Boltzmann	-47.15
SA	-140.1
CBIR	-76.5
UCB	-72.85

Exploration strategy	Value
Greedy	-99.6
ϵ-greedy	-79.75
decaying ϵ-greedy	-81.9
Pursuit	-78.2
Boltzmann	-53.35
SA	-76.0
CBIR	-97.7
UCB	-98.75

seen in Fig. 3(c)(f)(i) and Fig. 4(c)(f). The performance of the Boltzmann distribution method surpasses other methods, both in terms of convergence speed and final convergence result.

Fig. 6 and Fig. 7 show the obtained rewards in a 10x10 grid world with N_{obs} = 10,20,30,40,50,60 and T_r = 100. Table 7 and Table 8 show the average value of the rewards obtained in the last 20 episodes for each exploration strategy under this experimental setting. When N_{obs} = 10, the Boltzmann distribution method, decaying ϵ-greedy and SA show good performance that the agent could reach the goal within 100 steps with these strategies. When N_{obs} = 20, all methods except Boltzmann distribution and SA require the agent to take more than 100 steps to reach the goal. When N_{obs} is greater than 30, SA, ϵ-greedy and decaying ϵ-greedy become the best three strategies and Boltzmann falls behind under this setting. It can be clearly seen from Fig. 6(h) and Fig. 7(b)(e)(h) that UCB will reach a good result first around episode 20, and then it will perform worse and worse over episodes. The reason may due to the fact that UCB can learn a good policy at the beginning. Since T_r is 100, UCB will maintain its internal rewards for a longer period of time, so that after the environment is updated, the exploration ability of UCB becomes weak, and it is unable to make effective exploration to adapt to the new environment within 100 steps. While the environment keeps changing, the performance of UCB continues to decline over episodes.

6. Conclusion

This work has focused on the performance of various exploration strategies in a non-stationary environment. The performance of a Q-learning agent using the reported strategies namely Greedy, ϵ-greedy, Decaying ϵ-greedy, Boltzmann distribution method, Pursuit method, Thompson sampling, Simulated Annealing, Count-based intrinsic reward exploration, and Upper-Confidence-Bound have been studied in a grid world with refreshing obstacles. Through the experiments conducted, Boltzmann distribution method outperforms other strategies in most cases. Therefore, the Boltzmann distribution method could be a preferred choice to try for the exploration strategy of a Q-learning agent in a non-stationary environment.

References

[1] Y. Shen and C. Zeng, "An adaptive approach for the exploration-exploitation dilemma in non-stationary environment," vol. 1, 01 2008, pp. 497–500.

[2] M. Guo, Y. Liu, and J. Malec, "A new q-learning algorithm based on the metropolis criterion," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 34, no. 5, pp. 2140–2143, 2004.

[3] M. Yogeswaran and S. Ponnambalam, "Reinforcement learning: exploration–exploitation dilemma in multi-agent foraging task," *Opsearch*, vol. 49, no. 3, pp. 223–236, 2012.

[4] M. Juliá, A. Gil, and O. Reinoso, "A comparison of path planning strategies for autonomous exploration and mapping of unknown environments," *Autonomous Robots*, vol. 33, no. 4, pp. 427–444, 2012.

[5] A. Xie, J. Harrison, and C. Finn, "Deep reinforcement learning amidst lifelong non-stationarity," *arXiv preprint arXiv:2006.10701*, 2020.

[6] S. Padakandla, P. K. J., and S. Bhatnagar, "Reinforcement learning in non-stationary environments," *CoRR*, vol. abs/1905.03970, 2019. [Online]. Available: http://arxiv.org/abs/1905.03970

[7] S. Thrun, *Lifelong Learning Algorithms*. Boston, MA: Springer US, 1998, pp. 181–209. [Online]. Available: https://doi.org/10.1007/978-1-4615-5529-2_8

[8] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, 1989.

[9] T. M. Mitchell *et al.*, "Machine learning," 1997.

[10] S. Marsland, *Machine Learning: An Algorithmic Perspective, Second Edition*, 2nd ed. Chapman Hall/CRC, 2014.

[11] I. Kononenko, "Machine learning for medical diagnosis: history, state of the art and perspective," *Artificial Intelligence in medicine*, vol. 23, no. 1, pp. 89–109, 2001.

[12] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[13] X. Wang, "Intelligent multi-camera video surveillance: A review," *Pattern recognition letters*, vol. 34, no. 1, pp. 3–19, 2013.

[14] M. J. Pazzani and D. Billsus, "Content-based recommendation systems," in *The adaptive web*. Springer, 2007, pp. 325–341.

[15] J. Li, L. Deng, R. Haeb-Umbach, and Y. Gong, *Robust automatic speech recognition: A bridge to practical applications*, 01 2015.

[16] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016. [Online]. Available: https://doi.org/10.1038/nature16961

[17] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, Oct. 2017. [Online]. Available: https://doi.org/10.1038/nature24270

[18] O. Simeone, "A very brief introduction to machine learning with applications to communication systems," *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 4, pp. 648–664, 2018.

[19] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques," *Emerging artificial intelligence applications in computer engineering*, vol. 160, no. 1, pp. 3–24, 2007.

[20] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[21] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[22] P. Comon, "Independent component analysis, a new concept?" *Signal processing*, vol. 36, no. 3, pp. 287–314, 1994.

[23] S. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[24] S. C. Johnson, "Hierarchical clustering schemes," *Psychometrika*, vol. 32, no. 3, pp. 241–254, 1967.

[25] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in

large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[26] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 eighth ieee international conference on data mining*. IEEE, 2008, pp. 413–422.

[27] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *arXiv preprint arXiv:1406.2661*, 2014.

[28] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[29] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," 2017.

[30] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[31] J. Hu, H. Niu, J. Carrasco, B. Lennox, and F. Arvin, "Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 14 413–14 423, 2020.

[32] B. C. Da Silva, E. W. Basso, A. L. Bazzan, and P. M. Engel, "Dealing with non-stationary environments using context detection," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 217–224.

[33] R. Bellman, "A markovian decision process," *Journal of mathematics and mechanics*, vol. 6, no. 5, pp. 679–684, 1957.

[34] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[35] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[36] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[37] D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. Van Hasselt, and D. Silver, "Distributed prioritized experience replay," *arXiv preprint arXiv:1803.00933*, 2018.

[38] B. Price and C. Boutilier, "Accelerating reinforcement learning through implicit imitation," *Journal of Artificial Intelligence Research*, vol. 19, pp. 569–629, 2003.

[39] L. Weng, "The multi-armed bandit problem and its solutions," *lilianweng.github.io/lil-log*, 2018. [Online]. Available: http://lilianweng.github.io/lil-log/2018/01/23/the-multi-armed-bandit-problem-and-its-solutions.html

[40] I. Szita and A. Lőrincz, "The many faces of optimism: A unifying approach," in *Proceedings of the 25th International Conference on Machine Learning*, ser. ICML '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 1048–1055. [Online]. Available: https://doi.org/10.1145/1390156.1390288

[41] M. Derthick, "Variations on the boltzmann machine learning algorithm," 1984.

[42] E. L. L D Landau, *Statistical Physics*, 01 1980.

[43] M. Thathachar and P. Sastry, "A class of rapidly converging algorithms for learning automata," 1984.

[44] W. R. Thompson, "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples," *Biometrika*, vol. 25, no. 3/4, pp. 285–294, 1933.

[45] M. Strens, "A bayesian framework for reinforcement learning," in *ICML*, vol. 2000, 2000, pp. 943–950.

[46] D. Russo, B. V. Roy, A. Kazerouni, and I. Osband, "A tutorial on thompson sampling," *CoRR*, vol. abs/1707.02038, 2017. [Online]. Available: http://arxiv.org/abs/1707.02038

[47] M. Wiering and J. Schmidhuber, "Efficient model-based exploration," 1998.

[48] A. L. Strehl and M. L. Littman, "An analysis of model-based interval estimation for markov decision processes," *Journal of Computer and System Sciences*, vol. 74, no. 8, pp. 1309–1331, 2008.

[49] W. Hoeffding, "Probability inequalities for sums of bounded random variables," in *The Collected Works of Wassily Hoeffding*. Springer, 1994, pp. 409–426.

[50] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, vol. 47, no. 2, pp. 235–256, 2002.

[51] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks,"

in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[52] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

A. Appendix

Extended results when T_r = 100.

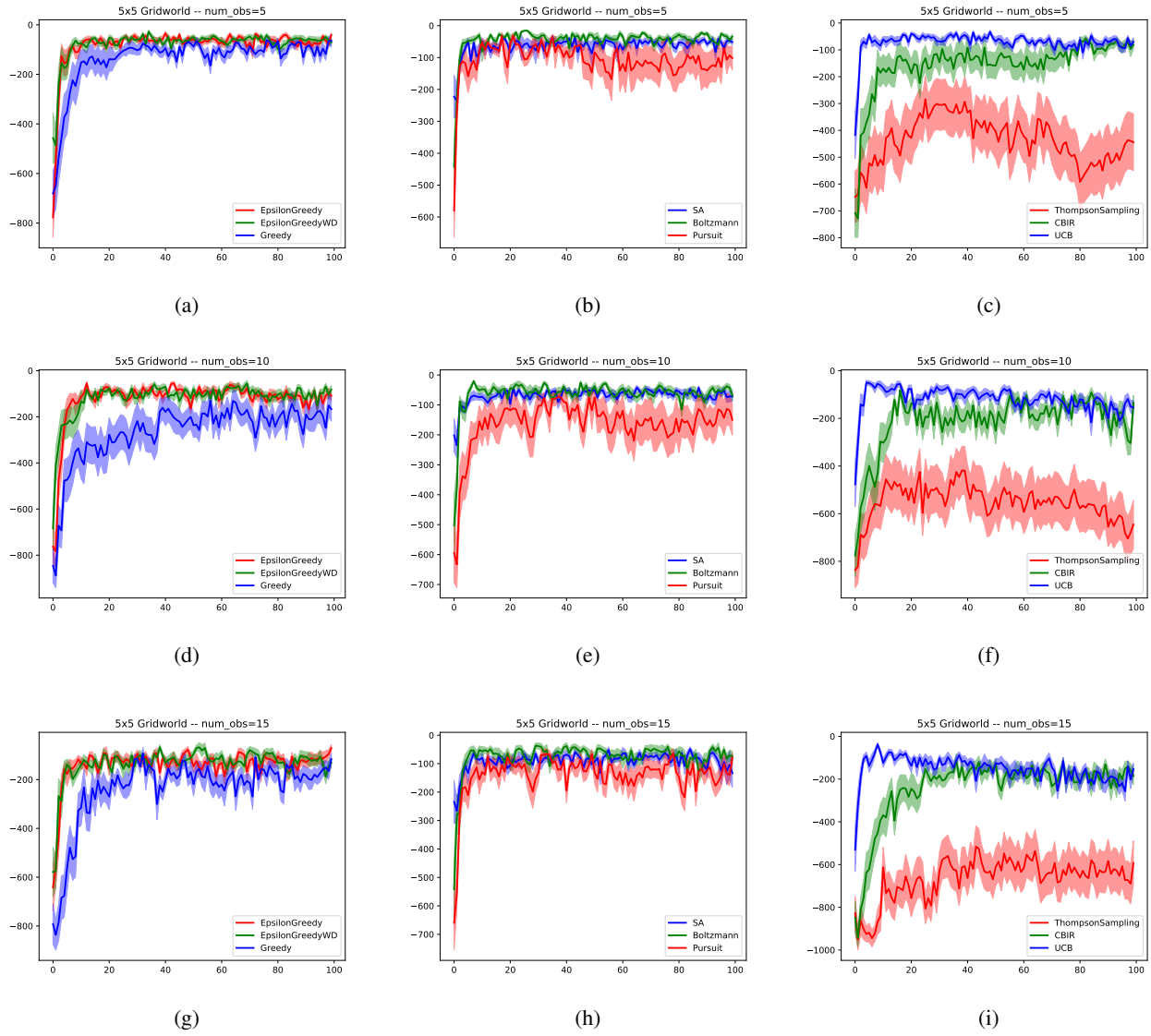

Figure 5. Obtained rewards vs number of episodes in a 5x5 grid world with N_{obs} = 5,10,15 and T_r = 100, averaged over 20 runs for each exploration strategy. (a)(d)(g) compares the performance of ϵ-greedy, decaying ϵ-greedy, and Greedy methods. (b)(e)(h) compares the performance of Pursuit, Boltzmann distribution and Simulated Annealing methods. (c)(f)(i) compares the performance of Thompson Sampling, Count-based intrinsic reward exploration and UCB method.

Table 6. Average rewards obtained in the last 20 episodes, when T_r=100, N_{obs}=5(left), 10(middle), 15(right) in a 5x5 grid world

Exploration strategy	Value
Greedy	-64.8
ϵ-greedy	-70.75
decaying ϵ-greedy	-97.3
Pursuit	-189.15
Boltzmann	-71.05
SA	-54.15
ThompsonSampling	-591.35
CBIR	-64.45
UCB	-61.85

Exploration strategy	Value
Greedy	-211.3
ϵ-greedy	-123.2
decaying ϵ-greedy	-102.25
Pursuit	-189.15
Boltzmann	-71.05
SA	-54.15
ThompsonSampling	-577.25
CBIR	-129.3
UCB	-118.7

Exploration strategy	Value
Greedy	-168.6
ϵ-greedy	-103.05
decaying ϵ-greedy	-131.25
Pursuit	-145.35
Boltzmann	-54.85
SA	-108.2
ThompsonSampling	-660.8
CBIR	-206.15
UCB	-219.05

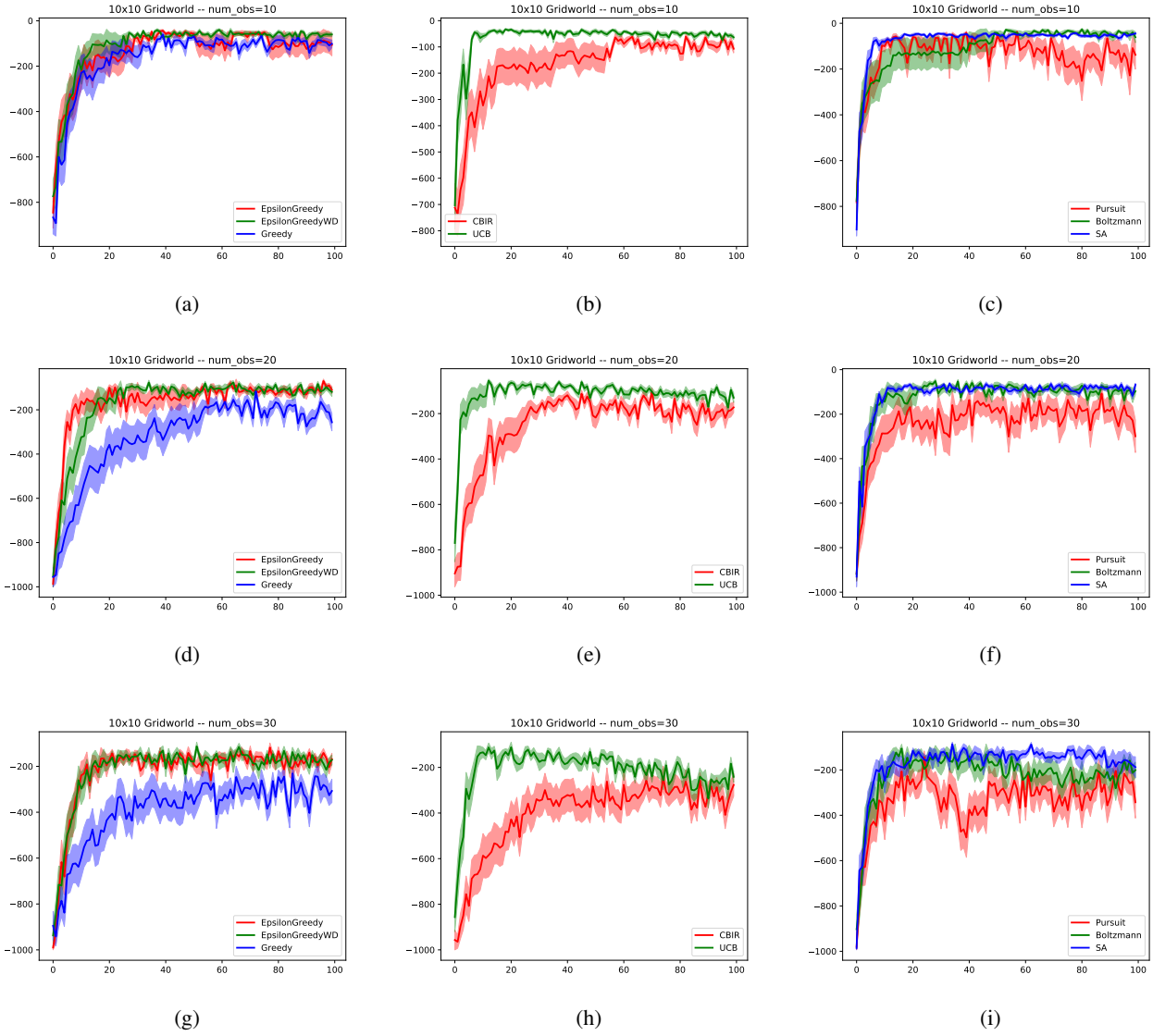

Figure 6. Obtained rewards vs number of episodes in a 10x10 grid world with N_{obs} = 10,20,30 and T_r = 100, averaged over 20 runs for each exploration strategy. (a)(d)(g) compares the performance of ϵ-greedy, decaying ϵ-greedy, and Greedy methods. (b)(e)(h) compares the performance of Count-based intrinsic reward exploration and UCB method. (c)(f)(i) compares the performance of Pursuit, Boltzmann distribution and Simulated Annealing methods.

Table 7. Average rewards obtained in the last 20 episodes, when T_r=100, N_{obs}=10(left), 20(middle), 30(right) in a 10x10 grid world

Exploration strategy	Value
Greedy	-83.15
ϵ-greedy	-115.1
decaying ϵ-greedy	-47.7
Pursuit	-251.85
Boltzmann	-26.7
SA	-55.4
CBIR	-124.75
UCB	-56.55

Exploration strategy	Value
Greedy	-170.7
ϵ-greedy	-118.4
decaying ϵ-greedy	-118.1
Pursuit	-185.75
Boltzmann	-70.3
SA	-88.05
CBIR	-217.8
UCB	-108.6

Exploration strategy	Value
Greedy	-246.05
ϵ-greedy	-132.95
decaying ϵ-greedy	-175.7
Pursuit	-377.65
Boltzmann	-209.6
SA	-151.8
CBIR	-310.3
UCB	-195.8

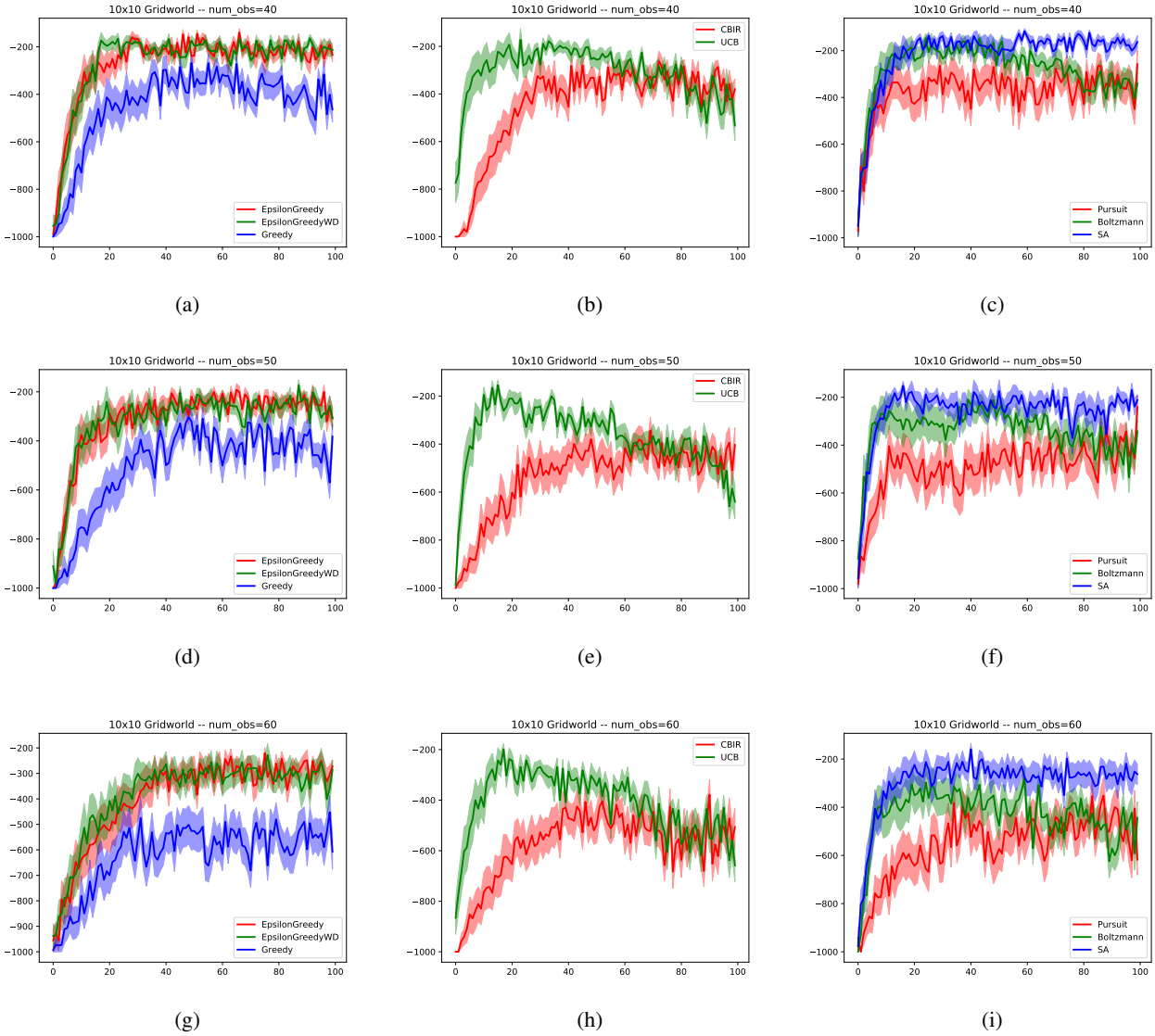

Figure 7. Obtained rewards vs number of episodes in a 10x10 grid world with N_{obs} = 40,50,60 and T_r = 100, averaged over 20 runs for each exploration strategy. (a)(d)(g) compares the performance of ϵ-greedy, decaying ϵ-greedy, and Greedy methods. (b)(e)(h) compares the performance of Count-based intrinsic reward exploration and UCB method. (c)(f)(i) compares the performance of Pursuit, Boltzmann distribution and Simulated Annealing methods.

Table 8. Average rewards obtained in the last 20 episodes, when T_r=100, N_{obs}=40(left), 50(middle), 60(right) in a 10x10 grid world

Exploration strategy	Value
Greedy	-349.25
ϵ-greedy	-199.25
decaying ϵ-greedy	-192.95
Pursuit	-426.35
Boltzmann	-378.8
SA	-156.65
CBIR	-314.0
UCB	-279.6

Exploration strategy	Value
Greedy	-453.6
ϵ-greedy	-271.05
decaying ϵ-greedy	-263.5
Pursuit	-384.9
Boltzmann	-350.1
SA	-218.45
CBIR	-314.0
UCB	-279.6

Exploration strategy	Value
Greedy	-543.3
ϵ-greedy	-314.6
decaying ϵ-greedy	-287.75
Pursuit	-496.65
Boltzmann	-435.25
SA	-260.0
CBIR	-635.35
UCB	-573.2