Comparing Exploration Strategies for Q-learning in Random Stochastic Mazes

Conference Paper · December 2016

DOI: 10.1109/SSCI.2016.7849366

3 authors:

Arryon Tijsma
SoundAppraisal
3 PUBLICATIONS **56** CITATIONS
SEE PROFILE

Madalina M Drugan
ITLearns Online
91 PUBLICATIONS **1,159** CITATIONS
SEE PROFILE

Marco A. Wiering
University of Groningen
234 PUBLICATIONS **4,201** CITATIONS
SEE PROFILE

Comparing Exploration Strategies for Q-learning in Random Stochastic Mazes

A.D. Tijsma*, M.M. Drugan† and M.A. Wiering*
*Institute of Artificial Intelligence and Cognitive Engineering
University of Groningen
Email: a.d.tijsma@rug.nl, m.a.wiering@rug.nl
†Department of Mathematics and Computer Science
Technical University Eindhoven
Email: madalina.drugan@gmail.com

***Abstract*—Balancing the ratio between exploration and exploitation is an important problem in reinforcement learning. This paper evaluates four different exploration strategies combined with Q-learning using random stochastic mazes to investigate their performances. We will compare: UCB-1, softmax, ϵ-greedy, and pursuit. For this purpose we adapted the UCB-1 strategy to be used in the Q-learning algorithm. The mazes consist of a single optimal goal state and two suboptimal goal states that lie closer to the starting position of the agent, which makes efficient exploration an important part of the learning agent. Furthermore, we evaluate two different kinds of reward functions, a normalized one with rewards between 0 and 1, and an unnormalized reward function that penalizes the agent for each step with a negative reward. We have performed an extensive grid-search to find the best parameters for each method and used the best parameters on novel randomly generated maze problems of different sizes. The results show that UCB-1 outperforms the other strategies when using the unnormalized reward function. For the normalized reward function, the results show that softmax is the best performing exploration strategy. Finally, UCB-1 is the only exploration strategy that performs well using the initially found meta-parameters on a smaller maze, when tested on a larger maze problem.**

I. Introduction

One of the most challenging tasks in reinforcement learning (RL) [1], [2] is that of balancing the ratio between exploration and exploitation. Too much exploration yields a lower accumulated reward, while too much exploitation can lead to the agent being stuck in a local optimum. This problem is known as the exploration/exploitation dilemma [3]–[5]. Because exploration may waste time exploring an irrelevant part of the environment, exploitation must happen simultaneously. One of the most widely used exploration strategies is ϵ-greedy [6]. The advantage of this strategy is that it is not dependent on specific data such as counters [7] and as an allround exploration strategy, ϵ-greedy is often hard to beat [8].

One can distinguish between two types of exploration: *directed* and *undirected* exploration [7], [9]. Undirected exploration is driven by randomness, the most trivial example being the 'random walk' [10] where the agent completely ignores the reward function and always performs random actions. ϵ-greedy is another example of undirected exploration, and is a case of using *semi-uniform distributions* [11], as is action selection based on the utility of an action [6], [12], of which *Boltzmann* or softmax exploration is an example.

Directed exploration methods can be distinguished using three categories. *Counter-based exploration* [9], [13] keeps count of how many times a state-action pair has been visited, and evaluates actions based on a linear combination of an exploitation term and an exploration term. Another category relies on exploring parts of the environment where the errors were large during the last updates, so called *error-based exploration* methods [14], [15]. Finally, there exists a category of directed exploration techniques called *recency-based exploration techniques* that prefers to select the action in a state that has been selected the longest time ago [9]. It has been shown that directed strategies perform better than undirected strategies for solving particular difficult maze problems when model-based RL is used [9]. It has also been shown that for the multi-armed bandit problem [16], simple heuristics like undirected exploration sometimes outperform more advanced algorithms [17].

In multi-armed bandit problems, there exists an algorithm called UCB-1 (for Upper Confidence Bound), which is a smart directed counter-based exploration strategy that has been shown to perform well for multi-armed bandits [16]. In this paper, this strategy is transferred to a Markov decision process in a similar way as in [18]. The question is whether UCB-1 will perform better or worse than the compared undirected exploration techniques, namely ϵ-greedy, softmax, and pursuit.

Contributions. This paper contributes a new comparison of exploration strategies in a stochastic maze problem. We use Q-learning [6] as reinforcement learning algorithm and investigate the effect that four different exploration strategies have on the cumulative reward intake. For this purpose, we built a random stochastic maze generator in which there are one optimal goal state and two suboptimal goal states. The task of the agent is to learn to navigate to the optimal goal state using the least amount of steps. In the same vein as in [18], we adapted the UCB-1 exploration strategy, often used for the multi-armed bandit problem, to Q-learning with state-action pairs. We compare UCB-1, ϵ-greedy, softmax and pursuit as exploration strategies using two different reward functions, a normalized one with rewards between 0 and 1, and an unnormalized reward function that penalizes every step not going to a goal state with a negative value. We also explore the effect of optimistic initialization of the Q-values by comparing this initialization scheme to the use of initial Q-values of zero. Furthermore, heatmaps for the exploration strategies are computed to gain insight into the general behavior of the reward intake as a function of the learning rate and each strategy's tunable parameter. This is important because certain approximation algorithms benefit from a well-defined search space [19], and this gives insight into how easy it is to tune a particular exploration method in combination with Q-learning.. The best found parameters are tested on novel generated mazes of different sizes, to make it possible to investigate the direct use of the best found parameters on a smaller maze to a maze problem of a bigger size. Our research question is: when learning to navigate in a stochastic maze using different exploration strategies, which strategy yields the largest accumulated reward, and which is the most consistent over different parameter values?

Outline of this paper. Section II presents the principles of reinforcement learning, Markov Decision Processes (MDPs) and Q-learning explaining the details of the four exploration strategies. Section III describes the methods used for training and evaluating the performance of the strategies. Section IV shows the results obtained from examining the influence of the two tunable parameters for each method and test the exploration strategies with the optimal parameters. Finally, the conclusions are presented in Section V.

II. Reinforcement Learning

Reinforcement learning [1], [2], [20] is an area of machine learning where an agent is connected to an environment via perception and action. At each step, the agent receives an input, chooses and executes an action, and this changes the state of the environment. The reward of executing the action in the previous state is then given to the agent by a reward function. The agent should maximize the cumulative sum of obtained rewards by choosing preferred actions, learned by trial and error using a particular reinforcement learning algorithm. The underlying model of this sequential decision making problem is a Markov Decision Process (MDP) [21], defined by the following principles:

- A discrete set of n states $S = \{s_1, s_2, ..., s_n\}$, $s_t \in S$ describing the state of the environment at time step t.
- A discrete set of m actions $A = \{a_1, a_2, ..., a_m\}$, where a_t denotes the action selected by the agent at time t.
- A transition function $T(s_t = s, a, s_{t+1} = s')$ that maps state-action pair s, a to the next state s' with a given transition probability.
- A reward function $R(s, a, s')$ denotes the average reward the agent obtains when transiting from state s to state s' using action a. r_t is the reward obtained at time t.
- A discount factor $0 \leq \gamma \leq 1$ that assigns a higher importance to immediate rewards compared to future rewards.

In reinforcement learning, the agent uses its past experiences to learn the optimal policy π^* which maps states to optimal actions that optimize the cumulative reward intake. Learning this policy involves the estimation of a value-function using past experiences, called the Q-function [1]. The Q-function $Q^\pi(s, a)$ denotes the expected accumulated discounted future reward obtained by selecting action a in state s and following policy π afterwards:

$$Q^\pi(s,a) = E(\sum_{t=1}^{\infty} \gamma^{t-1} r_t | s_t = s, a_t = a, \pi) \tag{1}$$

Q-learning [6] is an off-policy *temporal difference* (TD) [22] learning technique. With an off-policy learning method, the agent follows a behavioral policy and at the same time learns about the optimal Q-function. If the agent visits all state-action pairs an infinite number of times, Q-learning converges to the optimal Q-function [23]. Therefore, Q-learning can be used to learn the optimal policy for a given MDP [24], [25]. When the optimal Q-function is known, the optimal policy selects the action with the highest Q-value in a state.

The Q-learning update rule of the Q-value of a state-action pair at time step $t+1$, $Q_{t+1}(s_t, a_t)$ after an experience s_t, a_t, s_{t+1}, r_t, is as follows:

$$Q_{t+1}(s_t,a_t) = Q_t(s_t,a_t) + \alpha\big[r_t + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t)\big]$$

Where $0 \leq \alpha \leq 1$ is the learning rate of the update rule, and $0 \leq \gamma \leq 1$ is the discount factor.

A. Exploration Strategies

An essential part of Q-learning in finding an optimal policy is the selection of action a_t in state s_t. In Q-learning, there exists a tradeoff between selecting the currently expected optimal action, or selecting a different action in the hope it will yield a higher cumulative reward sum in the future. To investigate the goal-finding abilities of an agent in a maze using Q-learning, we investigate four different exploration strategies: ϵ-greedy, Boltzmann (also called *softmax*) exploration, pursuit, and UCB-1. We will discuss them below.

1) ϵ-greedy: ϵ-greedy exploration is one of the most used exploration strategies. It uses $0 \leq \epsilon \leq 1$ as parameter of exploration to decide which action to perform using $Q_t(s_t, a)$. The agent chooses the action with the highest Q-value in the current state with probability $1 - \epsilon$, and a random action otherwise. Randomness is necessary for an agent navigating through a stochastic maze to learn the optimal policy. With the experiments, we will find out which value of ϵ is optimal for our maze setup and how ϵ-greedy compares to other exploration strategies.

2) Boltzmann (or softmax) exploration: One drawback of ϵ-greedy exploration is that the exploration action is selected completely randomly from the set of possible actions. Therefore, it is as likely to choose the worst appearing action as it is to choose the second-best appearing action. That is why Boltzmann or softmax exploration [26] uses the Boltzmann distribution function to assign a probability $\pi(s_t, a)$ to the actions in order to create a graded function of estimated value:

$$\pi(s_t, a) = \frac{e^{Q_t(s_t,a)/T}}{\sum_{i=1}^{m} e^{Q_t(s_t,a(i))/T}} \tag{2}$$

$\pi(s_t, a)$ denotes the probability the agent will select action a in state s_t and T is the temperature parameter used in the Boltzmann distribution. Using softmax exploration, the agent still most likely selects the best action, but other actions are ranked instead of randomly chosen.

3) Pursuit: The pursuit method is adapted from the multi-armed bandit problem [1]. A pursuit method maintains both an action-value estimate and an action preference for the current state. Let $\pi_t(s_t, a)$ be the probability of selecting action a at play t when in the state s_t. After each time step t and the update of the Q-value, the greedy action $a^*_{t+1} = \arg\max_a Q_{t+1}(s = s_t, a)$ is selected in state s. a^*_{t+1} has a probability of successively being selected that is incremented with a fraction β towards one:

$$\pi_{t+1}(s_t, a^*_{t+1}) = \pi_t(s_t, a^*_{t+1}) + \beta\big[1 - \pi_t(s_t, a^*_{t+1})\big] \tag{3}$$

The probabilities for all the other actions in state s_t, on the other hand, are decreased towards zero:

$$\pi_{t+1}(s_t, a_{t+1}) = \pi_t(s_t, a_{t+1}) + \beta\big[0 - \pi_t(s_t, a_{t+1})\big], \textit{ for all } a_{t+1} \neq a^*_{t+1}$$

4) UCB-1: The exploration strategy *UCB-1* was also proposed for the multi-armed bandit problem [16], and is being adapted here for use with Q-learning. In [18], the authors adapted discounted UCB-1-tuned to be combined with Q-learning. However, they did not compare this method to the exploration strategies we study in this paper. The UCB-1 strategy keeps a count of the number of times an action a is executed in state s, for all actions and all states. After selecting an action, the strategy increases the count of the action taken in a state by one. The UCB-1 strategy first selects all actions exactly one time when a state is visited, so their counters have been populated. Afterwards, UCB-1 calculates an *exploration bonus* using:

$$bonus(s_t, a_t(i)) = 100 \times C \times \sqrt{\big(2 \times \frac{logN(s_t)}{N(s_t, a_t(i))}\big)} \tag{4}$$

where $N(s_t) = \sum_i N(s_t, a_t(i))$ denotes how often an action has been selected in state s_t, and $N(s_t, a_t(i))$ counts the number of times action i has been selected in state $s = s_t$. The bonus becomes larger when the fraction of counts for all actions versus the count for action $a_t(i)$ increases. A larger value of C increases exploration. For action selection, the bonus is added to the Q-value, and the action with the highest value in state s_t is selected:

$$a_t = \arg\max_i (Q(s_t, a(i)) + bonus(s_t, a(i))) \tag{5}$$

In [16] and [18], the term 100 does not appear in the equation to compute the bonus. However, we found that when using the unnormalized reward function (see section III-C) this aided the strategy's performance.

III. Experimental setup

A. Stochastic Mazes

The agent walks around in a maze environment. Figure 1 shows a 10×10 square maze, with obstacles B and three separate goals, two suboptimal goal states denoted g and the optimal goal state G. Each goal defines an end state to the maze, and the agent should learn to navigate to the optimal goal state. The agent always starts in state A, as shown in Figure 1. The rewards in the maze problem are shown in Table I and will be further explained in section III-C. Walking outside the maze boundaries or against a block B resets the agent to the state it was previously in, and yields the regular reward of a single step.

[width=2in]maze

Fig. 1. A random 10×10 maze

TABLE I
Rewards within the stochastic maze

	Non-normalized	Normalized
step	-1	0
block	-5	0
small goal	50	0.5
large goal	200	1.0

We define a randomly generated maze by placing the agent and the goals always in the same places, but varying the placement of the obstacles in the maze. In each maze, regardless of size, ten percent of the available spaces is occupied by blocks. Their positions are randomly sampled from the available spaces. In this way, the agent will be forced to take a different route each time, but the absolute distances from the agent to the goals remain the same. A random maze is deemed solvable if the agent can find all goal states within 3000 steps, taking a random walk in the maze. We will only use solvable maze problems in the experiments.

B. Terminology

The terminology we use for the various parts of the experiment will now be explained:

- **Epoch**: a strategic walk through the maze, until the agent reaches a goal state. Each epoch yields a cumulative reward sum and the total number of steps needed.
- **Run**: A full run of e epochs for a given combination of parameters. A run yields vectors $\vec{r}$ and $\vec{s}$ where $\vec{r}_i$ and $\vec{s}_i$ denote the total cumulative reward and the total number of steps taken for epoch i.
- **Search**: One run for each combination of parameters, performed on a single maze. This yields two search result matrices $\mathbf{R}$ and $\mathbf{S}$ of size $M \times N$, where M equals the total number of tested parameter combinations, and N equals the number of epochs in that run. The value $\mathbf{R}_{m,n}$ is the total reward with parameter combination m at epoch n, and the value $\mathbf{S}_{m,n}$ is the total number of steps to reach the goal for parameter combination m at epoch n.
- **Simulation**: A search performed on each of the randomly generated mazes, yielding two search result matrices per maze.
- **Test**: Using the best performing parameter combination from the simulation, we perform a run on each maze from a newly generated set of random mazes, optionally with extra repetitions for each run. The set of mazes can have different dimensions than the set on which we computed the best parameters for each method. The result is two new vectors $\vec{r}$ and $\vec{s}$ for each run, which we can average across runs to obtain a final score of performance for the strategy.
- **Experiment**: a set of simulations plus tests, each using different general parameters such as the initial Q-values of the Q-function, the rewards, or the transition probabilities. Each experiment investigates which strategy performs best regarding our research question, given the constraints of the general parameters.

C. Normalized and Non-normalized Reward Functions

In the original paper from Auer et al., the upper confidence bound of UCB-1 is proven given that the support of the rewards is in $[0, 1]$ [16]. This means that for correct comparison of the UCB-1 algorithm, we must also define a maze in which the reward is at most 1, and at least 0. We achieve this by defining two types of experiments using normalized and non-normalized reward functions. In the normalized version of the experiment, support is defined within [0,1] by assigning a reward of 1 to the largest goal, a reward of 0.5 to the smallest goal, and removing the negative rewards from a step and a block-encounter, see also Table I.

Since we also want to study the use of optimistic initialization of the Q-values in each scenario, we ultimately define four different types of experiments, see also Table II:

1) Non optimistic, non-normalized initialization, where rewards are set according to column 1 in Table I and initial Q-values are set to zero. This experiment is called **¬O/¬N**
2) Optimistic, non-normalized initialization, where rewards are set according to column 1 in Table I and initial Q-values are set to the value of the highest valued goal from column 1. This experiment is called **O/¬N**
3) Non optimistic, normalized initialization, where rewards are set according to column 2 in Table I and initial Q-values are set to zero. This experiment is called **¬O/N**
4) Optimistic, normalized initialization, where rewards are set according to column 2 in Table I and initial Q-values are set to the value of the highest valued goal from column 2. This experiment is called **O/N**

TABLE II
FOUR EXPERIMENTS OF EVALUATING EXPLORATION STRATEGIES

	Optimistic	Non-optimistic
Normalized	O/N	¬O/N
Non-normalized	O/¬N	¬O/¬N

D. Experimental Setup

As explained in Sections I and III-B, we perform an exhaustive search on the parameter space of each exploration strategy. Table III shows the range of parameters over which a search for each experiment is performed. The number of linearly spaced intervals for both α and the dependent parameter of each exploration strategy are set to 15, resulting in 225 parameter combinations for each search. The low values for C in UCB-1 in the normalized experiments is because we multiply this value by 100 when using the strategy, see Equation 4.

TABLE III
GRID SEARCH PARAMETERS USED IN NORMALIZED AND NON-NORMALIZED EXPERIMENTS. NORM. = NORMALIZED

		UCB-1		ϵ-greedy		softmax		pursuit	
norm.		C	α	ϵ	α	T	α	β	α
no	low	.2	.1	0.0	.1	.01	.1	.1	.1
	high	2.5	.95	1.0	.95	1.5	.95	1.0	.95
yes	low	.002	.1	0.0	.1	.01	.1	.01	.1
	high	.025	.95	1.0	.95	1.5	.95	.5	.95

In all simulations, the discount factor remains fixed at a value of $\gamma = 0.98$, and the total number of epochs is set to 3000. Furthermore, the transition probabilities also remain the same. There is a 70% probability the agent ends up in the state resulting from the chosen action (north, west, south, east), and there is a 10% probability the agent goes to each of the other three adjacent states.

Each run is repeated and averaged three times for the same maze, as an additional method to produce more accurate results. For each experiment and each simulation, we use the same set of ten random mazes of size 10×10 that we generated beforehand. For each test, we generated two new [width=1]img/heatmaps1.jpg

Fig. 2. Heatmaps for the experiment with the non-normalized reward function, shown as function of parameter combinations (x and y axes) for the four different exploration strategies combined with Q-learning.

sets of ten mazes with sizes 10×10 and 20×20. An optimistic initialization means that the initial Q-values are set to the value of the highest goal, as seen in table I. A non-optimistic initialization sets all initial Q-values to zero.

IV. Results

For each simulation, we calculate: 1) the average number of steps needed to reach a goal, and 2) the average cumulative reward sum. In each simulation, these measurements are taken per parameter combination, for 3000 epochs in a run, averaged over three times 10 mazes. We created a plot per parameter combination and exploration strategy, for which the average obtained rewards for the last 100 epochs are computed. The resulting heatmaps for the non-normalized reward function are shown in Figure 2.

The heatmaps in Figure 2 show clearly defined gradients for UCB-1 and ϵ-greedy in the experiment with the non-normalized reward function. The parameter search space of these strategies seems well-defined, and can be exploited by maximum-finding heuristics to search the parameter space in order to find the optimal parameter combinations. The heatmaps of softmax and pursuit are less clearly defined. The softmax approach performs worse as is shown in the figure and pursuit requires small values for the parameter β. This means that it will be more difficult to find optimal parameters for these strategies for the non-normalized reward functions. We can therefore conclude that UCB-1 and ϵ-greedy are exploration strategies of which the optimal parameter combinations are easiest to obtain for this experiment.

The best found parameters for all exploration strategies with the four different experimental setups are shown in Table IV.

TABLE IV
Optimal parameters for each type of experiment from table II

	O/N		¬O/N		O/¬N		¬O/¬N	
UCB-1	α	C	α	C	α	C	α	C
	.16	.007	.10	.007	.16	.69	.10	1.19
ϵ-greedy	α	ϵ	α	ϵ	α	ϵ	α	ϵ
	.95	.29	.83	.36	.59	.20	.65	.15
Softmax	α	T	α	T	α	T	α	T
	.16	.12	.10	.12	.95	1.18	.95	1.50
Pursuit	α	β	α	β	α	β	α	β
	.95	.007	.95	.007	.65	.007	.59	.007

Figures 2 till 5 show these obtained results from the exhaustive search. Table VII shows the averages and standard deviations for all four types of experiments.

Using these formed a test run, consisting of Q-learning using optimal parameters, and then performing 100 test epochs, on new random mazes of sizes 10x10 and 20x20. To obtain a measure for obtained reward and consistency, we calculate the average rewards of ten random mazes obtained in the test, for each strategy, and rank them using a one-sided Student's t-test. The results can be seen in tables V until VIII .

TABLE V
Average steps needed in 10x10 mazes to reach goal. 'norm.' = normalized, 'opt.' = optimistic

norm.	opt.	UCB-1	ϵ-greedy	pursuit	softmax	
yes	yes	29	83	33	22	μ
		5.0	66.9	11.2	3.2	σ
	no	27	59	36	47	μ
		5.6	37.7	14.4	19.3	σ
no	yes	29	31	26	24	μ
		6.3	12.2	9.5	7.6	σ
	no	27	30	26	23	μ
		5.8	10.4	7.1	7.0	σ

TABLE VI
Average steps needed in 20x20 mazes to reach goal. 'norm.' = normalized, 'opt.' = optimistic

norm.	opt.	UCB-1	ϵ-greedy	pursuit	softmax	
yes	yes	41	119	78	42	μ
		5.4	39.7	31.7	8.0	σ
	no	36	81	75	33	μ
		4.4	24.5	28.3	3.4	σ
no	yes	44	66	75	68	μ
		7.3	26.6	29.3	29.6	σ
	no	44	60	65	85	μ
		8.0	19.4	20.8	33.6	σ

TABLE VII
Average reward during last 100 epochs for 10x10 mazes obtained at goal, higher is better. 'norm.' = normalized, 'opt.' = optimistic. Rank determined by students t-test with p=0.05

norm.	opt.	UCB-1	ϵ-greedy	pursuit	softmax	
yes	yes	0.989	0.974	0.998	**1.000**	μ
		0.03	0.06	0.01	**0.00**	σ
		3	4	2	**1**	rank
	no	0.961	0.980	**0.997**	**0.998**	μ
		0.08	0.05	**0.01**	**0.01**	σ
		3	2	**1**	**1**	rank
no	yes	171	157	**174**	85	μ
		10.7	32.4	**8.0**	46.5	σ
		2	3	1	4	rank
	no	**173**	158	**174**	84	μ
		7.0	22.1	**9.8**	42.0	σ
		1	2	**1**	3	rank

Whenever rewards are normalized, and no penalty is given to performing a step, all strategies need more steps on average [width=1]img/heatmaps2.jpg

Fig. 3. Heatmaps for normalized experiment rewards shown as function of parameter combinations (x and y axes). Note that the normalized heatmaps are not shown in the same dynamic range, as the reward of some strategies barely changes as function of the parameters

[width=1]img/rewards1.jpg

Fig. 4. Reward obtained at the goal from the last 100 training epochs for non-normalized (top two rows) and normalized (bottom two rows) experiments. Black line is average value for all parameter combinations. First contour area represents the standard deviation, second contour area represents the minimum and maximum score for all parameters for the given epoch

TABLE VIII
AVERAGE REWARD DURING LAST 100 EPOCHS FOR 20X20 MAZES OBTAINED AT GOAL, HIGHER IS BETTER. 'NORM.' = NORMALIZED, 'OPT.' = OPTIMISTIC. RANK DETERMINED BY STUDENTS T-TEST WITH P=0.05

norm.	opt.	UCB-1	ϵ-greedy	pursuit	softmax	
yes	yes	0.500	0.500	0.516	**0.716**	μ
		0.00	0.00	0.05	**0.18**	σ
		3	3	2	**1**	rank
	no	0.500	0.500	**0.517**	0.500	μ
		0.00	0.00	**0.05**	0.00	σ
		2	2	**1**	2	rank
no	yes	**62**	-2	21	-2	μ
		11.4	16.7	36.0	16.8	σ
		1	3	2	3	rank
	no	**42**	-2	14	-17	μ
		40.9	16.7	23.4	24.0	σ
		1	3	2	4	rank

to reach a goal. On a 10x10 maze, softmax and pursuit obtain the highest maximum reward during testing, softmax being the most consistent since it obtains the highest reward in both scenarios. On a 20x20 maze, softmax and pursuit again obtain the highest rewards.

When rewards are not normalized, UCB-1 performs best in three out of four experiment configurations, being beaten once by Pursuit.

Overall, we conclude that Pursuit is the most consistent strategy, because in all experiments with both 10x10 and 20x20 mazes, it performs best in four out of eight experiments.

UCB-1, part of the directed, count-based class of exploration strategies, performs well in non-normalized experiments, both on 10x10 mazes and on 20x20 mazes. In normalized experiments it does not perform as well, being beaten by softmax and pursuit.

Test results in case of UCB-1 sometimes exceed maximum values from training performance (compare the values from the normalized experiments in table I to those of the max trace for UCB-1 in the bottom two rows of figure 4a), indicating good generalization on the maze problem. This is further confirmed by the performance in the 20x20 maze size test.

Surprisingly, UCB-1 performs better when the support for its rewards is not normalized, even though according to literature, this is the range in which the upper bound of the strategy is proven. We conclude that UCB-1 with non-normalized rewards support benefits from a multiplication of the bonus constant C, as shown in section II-A4.

V. CONCLUSION

While this paper is a good primer on performance of exploration strategies, it is by no means complete. As mentioned in the introduction, there exist a wide variety of exploration/exploitation algorithms, both in the directed and undirected categories. Most strategies used here are in their basic form, having been improved and tuned in different ways. Creating a thorough comparison of a multitude of these strategies would give an even better overview of performance of exploration strategies for the given MDP.

The bonus term used in UCB-1 was applied after we saw poor performance of this strategy compared to others, while literature suggested this should not be the case. The value of 100 has the same order of magnitude as the largest reward in the maze, however for a good evaluation different values should be chosen to see how this reflects performance.

The constructed mazes took a certain amount of handmade precision in order to avoid a ceiling or floor effect when evaluating the different strategies. It will be interesting to see which type of convergence is achieved when different types of mazes are used, with more or with less goals, with dynamic placement of both goals and the agent.

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 1998.

[2] M. A. Wiering and M. van Otterlo, *Reinforcement learning: State-of-the-art*. Springer, 2012.

[3] S. B. Thrun, “The role of exploration in learning control,” *Handbook of intelligent control: Neural, fuzzy and adaptive approaches*, 1992.

[4] S. Yahyaa, “Explorations in reinforcement learning: Online action selection and online value function approximation,” Ph.D. dissertation, Free University of Brussels, 2015.

[5] M. A. Wiering, “Explorations in efficient reinforcement learning,” Ph.D. dissertation, University of Amsterdam, 1999.

[6] C. J. C. H. Watkins, “Learning from delayed rewards,” Ph.D. dissertation, King’s College, 1989.

[7] S. B. Thrun, “Efficient exploration in reinforcement learning,” Tech. Rep., 1992.

[8] J. Vermorel and M. Mohri, “Multi-armed bandit algorithms and empirical evaluation,” in *European conference on machine learning*. Springer, 2005, pp. 437–448.

[9] M. Wiering and J. Schmidhuber, “Efficient model-based exploration,” in *Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior (SAB98)*, 1998, pp. 223–228.

[10] M. C. Mozer and J. Bachrach, “Discovering the structure of a reactive environment by exploration,” *Neural computation*, vol. 2, no. 4, pp. 447–457, 1990.

[11] S. D. Whitehead and D. H. Ballard, “Learning to perceive and act by trial and error,” *Machine Learning*, vol. 7, no. 1, pp. 45–83, 1991.

[12] R. S. Sutton, “Integrated architectures for learning, planning, and reacting based on approximating dynamic programming,” in *Proceedings of the seventh international conference on machine learning*, 1990, pp. 216–224.

[13] M. Sato, K. Abe, and H. Takeda, “Learning control of finite markov chains with an explicit trade-off between estimation and control,” *IEEE transactions on systems, man, and cybernetics*, vol. 18, no. 5, pp. 677–684, 1988.

[14] J. Schmidhuber, “Adaptive confidence and adaptive curiosity,” in *Institut fur Informatik, Technische Universitat Munchen, Arcisstr. 21, 800 Munchen 2*. Citeseer, 1991.

[15] S. B. Thrun and K. Mller, “Active exploration in dynamic environments,” in *Advances in neural information processing systems*, 1992, pp. 531–538.

[16] P. Auer, N. Cesa-Bianchi, and P. Fischer, “Finite-time analysis of the multiarmed bandit problem,” *Machine Learning*, vol. 47, no. 2-3, pp. 235–256, 2002.

[17] V. Kuleshov and D. Precup, “Algorithms for multi-armed bandit problems,” *arXiv preprint arXiv:1402.6028*, 2014.

[18] K. Saito, A. Notsu, and K. Honda, “Discounted ucb1-tuned for q-learning,” in *Soft Computing and Intelligent Systems (SCIS), 2014 Joint 7th International Conference on and Advanced Intelligent Systems (ISIS), 15th International Symposium on*. IEEE, 2014, pp. 966–970.

[19] J. Kennedy, *Particle swarm optimization*, ser. Encyclopedia of machine learning. Springer, 2011, pp. 760–766.

[20] L. P. Kaelbling, M. L. Littman, and A. W. Moore, “Reinforcement learning: A survey,” *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[21] R. E. Bellman, *Dynamic Programming*. Courier Dover Publications, 1957.

[22] R. S. Sutton, “Learning to predict by the methods of temporal differences,” *Machine Learning*, vol. 3, no. 1, pp. 9–44, 1988.

[23] T. Jaakkola, M. I. Jordan, and S. P. Singh, “On the convergence of stochastic iterative dynamic programming algorithms,” *Neural computation*, vol. 6, no. 6, pp. 1185–1201, 1994.

[24] R. Bellman, “A markovian decision process,” *Journal of Mathematics and Mechanics 6*, 1957.

[25] R. A. Howard, *Dynamic Programming and Markov Processes*. Technology Press of Massachusetts Institute of Technology, 1960.

[26] A. G. Barto, S. J. Bradtke, and S. P. Singh, *Real-time learning and control using asynchronous dynamic programming*. University of Massachusetts at Amherst, Department of Computer and Information Science, 1991.