

A Fast Reinforcement Learning Technique via Multiple Lookahead Levels

H. S. Al-Dayaa

CMINDS Research Center
Electrical and Computer Engineering Department
University of Massachusetts Lowell
Lowell MA, USA

D. B. Megherbi

CMINDS Research Center
Electrical and Computer Engineering Department
University of Massachusetts Lowell
Lowell MA, USA

Abstract – Reinforcement learning techniques have contributed and continue to tremendously contribute to the advancement of machine learning. In this paper, we introduce a technique for reinforcement learning using multiple lookahead levels that grants an autonomous agent more visibility in its environment and helps it learn faster. This technique extends the Watkins’s Q -Learning algorithm by using the Multiple-Lookahead-Levels equation we developed in this paper. We analyze the convergence of the MLL equation and prove its effectiveness. We also propose a method to compute the improvement rate of the agent’s learning speed between different lookahead levels. Here, both the time and space complexities are examined. Results show that the number of steps, required to achieve the goal, per learning path exponentially decreases with the learning path number (time). Results also show that the number of steps per learning path is less at any time when the number of lookahead levels is higher (space).

Keywords: Reinforcement Learning, Machine Learning, Intelligent Control.

1 Introduction

We learn by interacting with our environment [12]. There is a learning process in most aspects of life like how to eat, walk, speak, write, read, and drive, to name a few. The environment may be different for each of them, but it mainly consists of the same elements: the starting state, one or more obstacles, and the goal. To achieve the goal, we take actions that avoid obstacles and lead us closer to it. For every action, there is a reaction; we adopt the actions that result in more rewards and avoid the ones that result in less rewards (or punishments). After we reach the goal, we use the information we learned about the environment to improve, and we keep improving until we learn the optimal way to that goal. The more useful information we have, the faster we can learn that optimal way.

Reinforcement learning applies the same concept on machines. It addresses how an autonomous agent can learn and choose optimal actions to achieve a determined goal in an uncertain environment based on reactions (return values in type of rewards) received after each action [15]. In traditional reinforcement learning, the agent uses the return

values of the next possible states to decide its next action from the state it is in; hence, it uses a single lookahead level. Figure 1 illustrates the agent-environment interaction in reinforcement learning as presented by Sutton and Barto [12], i.e. single-lookahead-level learning.

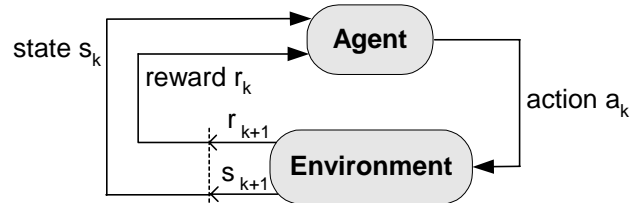


Figure 1 – Agent-Environment Interaction in Reinforcement Learning Using Single-Lookahead-Level Techniques

The single-lookahead-level reinforcement learning methods are widely used in the literature for various applications and architecture [4][12][13][14][15] as in the ‘Integrated Architecture for Learning, Planning, and Reacting Based on Approximating Dynamic Programming’ by Sutton [14]. One of the main limitations of such methods is that they provide the agent with only one level of visibility, which may limit its learning capacity and speed. The purpose of this work is to enhance the single-lookahead-level concept by adding more levels of visibility. This can be done by developing a way to *reward the agent using not only the reward values of the current and next states but also those of the after-next states and so on.*

In this paper, we introduce, analyze, and demonstrate the Multiple-Lookahead-Levels (MLL) technique. Section 2 presents the basis of the work in this paper, and section 3 depicts the proposed solution and includes the main contribution theorems and their proofs. Section 4 presents the proposed algorithm. In section 5, we show and analyze experimental results illustrating the performance and value of the proposed technique. Section 6 constitutes the conclusion and future work.

2 Basis of this Work

The aim of the proposed reinforcement learning technique using multiple lookahead levels is to allow the agent to look further ahead and use return values from possible states that will occur after the next one to decide its next action. With more visibility of the future states and

their return values, the agent can learn faster as we show in this paper. Figure 2 illustrates the agent-environment interaction in reinforcement learning using multiple lookahead levels.

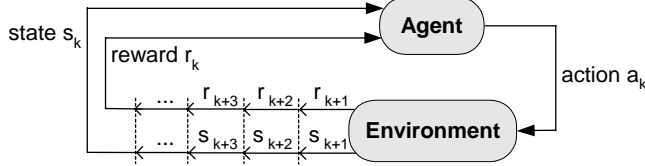


Figure 2 – Agent-Environment Interaction in Reinforcement Learning Using the MLL Technique

In particular, the proposed MLL technique combines Watkins’s Q-Learning with the λ -return equation used by Sutton and Barto [12] in their forward-view algorithm and produces, as we show in this paper, a complete MLL equation where the return is a function of different rewards from different lookahead levels weighted by a lookahead weight factor λ .

3 Proposed Solution: Main Theorems

The following proposition and theorems constitute some of the main contributions of this paper. Proposition 1 relates to the MLL equation, Theorem 1 relates to its learning convergence, and Theorem 3 relates to its convergence speed.

Let E be an environment, as depicted in Figure 2, where the agent is always in a known state s_k from which it can take a known action a_k and receive an updated return (or reward) $Q(s_k, a_k)$ and where the index k represents the space coordinates that are i for vertical and j for horizontal in the case of a two-dimensional environment (maze). In E , let $Q(s_k, a_k)$ be the return received from executing the action a_k from the state s_k (the current state), $r(s_k, a_k)$ the immediate return (0 or 1) received from executing the action a_k from the state s_k , $Q(s_{k+1}, a_{k+1})$ the reward that will be received after executing the action a_{k+1} from the state s_{k+1} (the next state), α ($0 < \alpha < 1$) the step-size parameter, γ ($0 < \gamma < 1$) the discount rate, λ ($0 \leq \lambda \leq 1$) the lookahead weight factor, and N the number of lookahead levels. N is a user-defined variable to be chosen as a tradeoff between agent environment space size and desired learning speed. The methodology behind choosing an optimal N is beyond the scope of this paper and will be published elsewhere.

Proposition 1: Given an agent in the environment E and given $R(s_{k+n}, a_{k+n})$ being the actual return following action n , for all values of n , the return $Q(s_k, a_k)$, which is $Q(s_k, a_k)$ at time $t+1$, is given by the MLL equation as depicted in (1), which implements the visibility of forward states and uses the return values from those states in combination with the return value from the current state.

$$\hat{Q}(s_k, a_k) \leftarrow Q(s_k, a_k) + \alpha \left[(1 - \lambda) \sum_{n=1}^N [\lambda^{n-1} R(s_{k+(n-1)}, a_{k+(n-1)})] + \lambda^N R(s_{k+(N-1)}, a_{k+(N-1)}) - Q(s_k, a_k) \right] \quad (1)$$

Proof:

We start with the regular Q-Learning reinforcement-learning equation given in (2) used to generate the return following each action.

$$\hat{Q}(s_k, a_k) \leftarrow Q(s_k, a_k) + \alpha \left[r(s_k, a_k) + \gamma \max_{a_{k+1}} Q(s_{k+1}, a_{k+1}) - Q(s_k, a_k) \right] \quad (2)$$

This equation only supports one lookahead level. It is originated from a simple every-step Monte Carlo method [3] shown in (3), where R_k is equivalent to $R(s_k, a_k)$ and is the actual return following action a_k from state s_k . $V(s_k)$ and $Q(s_k, a_k)$ are equivalent to $Q(s_k, a_k)$ and $Q(s_k, a_k)$, respectively.

$$\hat{V}(s_k) \leftarrow V(s_k) + \alpha [R_k - V(s_k)] \quad (3)$$

By comparing equations (2) and (3), we obtain equation (4).

$$R(s_k, a_k) = r(s_k, a_k) + \gamma \max_{a_{k+1}} Q(s_{k+1}, a_{k+1}) \quad (4)$$

A general equation of forward propagation [12] is shown in (5). λ is the lookahead weight factor such that $0 \leq \lambda \leq 1$. The larger λ is the more weight is given to future states on the expense of the current one and vice versa.

$$R_k^\lambda = (1 - \lambda) \sum_{n=1}^{N-k-1} \lambda^{n-1} R_k^{(n)} + \lambda^{N-k-1} R_k \quad (5)$$

Substituting R_k in (4) with its general value in (5) results in the equation shown in (6).

$$\hat{V}(s_k) \leftarrow V(s_k) + \alpha \left[(1 - \lambda) \sum_{n=1}^{N-k-1} \lambda^{n-1} R_k^{(n)} + \lambda^{N-k-1} R_k - V(s_k) \right] \quad (6)$$

We then write the equation in (6) using the Q-Learning format and taking into consideration the definition $R(s_k, a_k)$ in (4). The resulting equation shown in (7) is the MLL equation shown (1) as well.

$$\hat{Q}(s_k, a_k) \leftarrow Q(s_k, a_k) + \alpha \left[(1 - \lambda) \sum_{n=1}^N [\lambda^{n-1} R(s_{k+(n-1)}, a_{k+(n-1)})] + \lambda^N R(s_{k+(N-1)}, a_{k+(N-1)}) - Q(s_k, a_k) \right] \quad (7)$$

The general MLL equation shown in (7) is what is going to be used in the Watkins’s Q-Learning algorithm for multiple lookahead levels. The equations in (8) and (9) show the MLL equation for 2 lookahead levels and 3 lookahead levels, respectively.

$$\hat{Q}(s_k, a_k) \leftarrow Q(s_k, a_k) + \alpha \left[(1-\lambda) [R(s_k, a_k) + \lambda R(s_{k+1}, a_{k+1})] + \lambda^2 R(s_{k+1}, a_{k+1}) - Q(s_k, a_k) \right] \quad (8)$$

$$\hat{Q}(s_k, a_k) \leftarrow Q(s_k, a_k) + \alpha \left[(1-\lambda) \left[R(s_k, a_k) + \lambda R(s_{k+1}, a_{k+1}) + \lambda^2 R(s_{k+2}, a_{k+2}) \right] + \lambda^3 R(s_{k+2}, a_{k+2}) - Q(s_k, a_k) \right] \quad (9)$$

Note that if $\lambda = 0$, the MLL equation in (7) is the original Q-Learning equation, which considers a single lookahead level. As λ increases, the MLL technique places increasing emphasis on discrepancies based on more lookahead levels. Q.E.D.

Theorem 1: The updated return $\hat{Q}(s_k, a_k)$ computed by the MLL equation in (7) converges to the maximum (optimal) return $Q(s, a)$ as $t \rightarrow \infty$ for all states s and actions a .

Proof:

$\hat{Q}(s_k, a_k)$ in (2) is known to converge to $Q(s, a)$, with probability 1, as $k \rightarrow \infty$ for all states s and actions a [12][14][15]. $R(s_k, a_k)$ is also proven to converge to $R(s, a)$ as time $\rightarrow \infty$ for all s and a [15]. The conditions under which they converge are present in the MLL technique: The step-size parameter α is chosen such that $0 < \alpha < 1$, the discount rate γ is such that $0 < \gamma < 1$, and the immediate reward $r(s_k, a_k)$ is bounded such that $|r(s_k, a_k)| \leq 1$ for all s and a . The choice of the lookahead weight factor λ such that $0 \leq \lambda \leq 1$ is also necessary for convergence, so that λ^n converging measure. The following is a mathematical proof based on the fact that $\hat{Q}(s_k, a_k)$ converges to $Q(s, a)$ and $R(s_k, a_k)$ converges to $R(s, a)$, as mentioned earlier in this paragraph.

Equation (7) can be written as follows in (10).

$$\hat{Q}(s_k, a_k) \leftarrow Q(s_k, a_k) + \alpha [F(\lambda, N, R) - Q(s_k, a_k)] \quad (10)$$

where

$$F(\lambda, N, R) = (1-\lambda) \sum_{n=1}^N [\lambda^{n-1} R(s_{k+(n-1)}, a_{k+(n-1)})] + \lambda^N R(s_{k+(N-1)}, a_{k+(N-1)}) \quad (11)$$

As $R(s_k, a_k)$ converges to $R(s, a)$, $F(\lambda, N, R)$ becomes a series shown in (12).

$$\begin{aligned} F(\lambda, N, R) &\rightarrow \\ &(1-\lambda)(1 + \lambda + \dots + \lambda^{N-2} + \lambda^{N-1}) R(s, a) \\ &\quad + \lambda^N R(s, a) \\ &= R(s, a) \end{aligned} \quad (12)$$

Hence, the equation in (10) becomes as shown in (13).

$$\hat{Q}(s_k, a_k) \leftarrow Q(s_k, a_k) + \alpha [R(s_k, a_k) - Q(s_k, a_k)] \quad (13)$$

The equation in (13) is the same as the Q-Learning equation in (2), which is proven to converge as mentioned earlier. Therefore,

$\hat{Q}(s_k, a_k)$ still converges to $Q(s, a)$ as time $\rightarrow \infty$ for all states s and actions a . Q.E.D.

Theorem 2: The larger the number of lookahead levels N is, the faster the MLL equation converges to the maximum return value $Q(s, a)$.

Proof:

As stated in Proposition 1, $\hat{Q}(s_k, a_k)$ is $Q(s_k, a_k)$ at time $t+1$. Hence, the equation in (10) can be converted to the time domain and written in equation (14) that leads to the first-order differential equation in (15) where Δt is a time-step positive constant.

$$q(t+1) = q(t) + \alpha [f(\lambda, N, r) - q(t)] \quad (14)$$

$$\dot{q}(t) = \Delta t \alpha [f(\lambda, N, r) - q(t)] \quad (15)$$

As proven in Theorem 2, $\hat{Q}(s_k, a_k)$ converges to the maximum return value $Q(s, a)$ at the time $t \rightarrow \infty$ for all states s and actions a . This fact is illustrated in the time domain and shown in the exponential trend of the form $e^{Ct} = q(t)$ (where C is a positive constant) in Figure 3. Hence, it is necessary to prove that the larger the number of lookahead levels N is, the faster $q(t)$ ascends from its initial return value of 0 toward the maximum return value $Q(s, a)$.

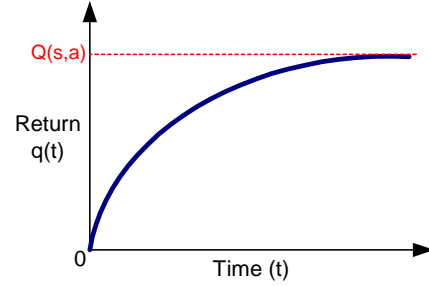


Figure 3 – Convergence of $q(t)$ to Maximum Return $Q(s, a)$

For all states s_k except the one that leads directly to the goal, the immediate return $r(s_k, a_k)$ in equation (4) is 0. Let θ be a space factor (a real number), such as $\theta > 1$. Note that here θ multiplies the actual return not reduces it toward 0. The actual-return equation in (4) in the time domain can be written as shown in (16) where γ is the discount rate ($0 < \gamma < 1$) as described in the environment E characteristics and k is the space index.

$$r(t) = \theta^k \gamma q(t) \quad (16)$$

Therefore, using equation (16), $F(\lambda, N, R)$ from (11) can be written in the time domain as shown in equation (17).

$$f(\lambda, N, r) = (1-\lambda) \left(\sum_{n=1}^N \lambda^{n-1} \theta_n \right) \gamma q(t) + \lambda^N \theta_N \gamma q(t) \quad (17)$$

The differential equation in (15) becomes of the form shown in equation (18).

$$\dot{q}(t) = \Delta t \alpha \gamma \left[\left[(1-\lambda) \left(\sum_{n=1}^N \lambda^{n-1} \theta_n \right) + \lambda^N \theta_N \right] - 1 \right] q(t) \quad (18)$$

The solution for this differential equation is $q(t)$ of the form shown in (19) where C is the series given in (20).

$$\begin{aligned} q(t) &= e^{\Delta t \alpha \gamma \left[\left[(1-\lambda) \left(\sum_{n=1}^N \lambda^{n-1} \theta_n \right) + \lambda^N \theta_N \right] \gamma - 1 \right] t} \\ &= e^{\Delta t \alpha \gamma \left[\left[(1-\lambda) (\theta + \lambda \theta^2 + \dots + \lambda^{N-2} \theta^{N-1} + \lambda^{N-1} \theta^N) + \lambda^N \theta^N \right] - 1 \right] t} \\ &= e^{\Delta t \alpha \gamma \left[\left[\theta (1-\lambda) (1 + \lambda \theta + \dots + (\lambda \theta)^{N-2} + (\lambda \theta)^{N-1}) + (\lambda \theta)^N \right] - 1 \right] t} \\ &\equiv e^{Ct} \end{aligned} \quad (19)$$

$$C = \Delta t \alpha \gamma \left[\left[\theta (1-\lambda) (1 + \lambda \theta + (\lambda \theta)^2 + \dots + (\lambda \theta)^{N-2} + (\lambda \theta)^{N-1}) + (\lambda \theta)^N \right] - 1 \right] \quad (20)$$

Having $0 \leq \lambda \leq 1$ and $\theta > 1$, the value C is positive or zero but never negative, and it increases with the number of lookahead levels N . Hence, the larger N is, the faster $q(t)$ converges to the maximum return value. Q.E.D.

4 Proposed Algorithm

The resulting proposed MLL reinforcement learning technique algorithm consists of the steps shown below. Moreover, these steps should be executed in the order they are listed.

1. Do an initial, usually random, physical path until goal is reached.
2. Compute the immediate ($r(s_k, a_k)$) and discounted reward values ($Q(s_k, a_k)$). The immediate reward = 0 for all state-action transitions and 1 for that leading to the goal. The discounted reward = γ^i ; where γ , $0 < \gamma < 1$, is the discount rate, and i is the number of time steps into the future (e.g. $i = 1$ for the state-action transition that directly leads to the goal; $i = 2$ for the state-action transition just before it in the path, etc.).
3. Update the world model in the agent's brain with the visited states, detected obstacles, and goal. Store the immediate and discounted reward values in their corresponding state-action transitions.
4. Do L learning (or hypothetical) paths (or trials), where L is the number of learning paths per physical path specified by the user, and each learning path consists of the following steps:
 - 4.a. Start with a known state s_k , which is usually the starting state.
 - 4.b. Choose an action $a_k \leftarrow \text{Policy}(s_k, a_k)$, where the policy used in this work is the Boltzmann distribution.
 - 4.c. Choose actions $a_{k+j} \leftarrow \text{Policy}(s_{k+j}, a_{k+j})$, where j is incremented from 1 to N with N being the number of lookahead levels specified by the user.
 - 4.d. Update $Q(s_k, a_k)$ for time $t+1$ using the MLL equation.
 - 4.e. Update the learning model in the agent's brain with the new value of $Q(s_k, a_k)$.

- 4.f. Update current state $s_k \leftarrow s_{k+1}$. Go to Step 4.a and repeat until goal is reached.
5. Do a learned physical path, where the agent moves based on the policy taking in consideration the updated $Q(s_k, a_k)$ values in its learning model. A physical path ends when the goal is reached.
6. Update the world model in the agent's brain with the newly visited states, detected obstacles, and goal.
7. Go to Step 4 and repeat until $P-1$ learned physical paths are done, where P is the total number of physical paths specified by the user and includes the initial path.

5 Experimental Results

As an illustration to the MLL technique, consider an autonomous agent's task T of navigating a maze to find the optimal (shortest) path to the goal. The maze is a two-dimensional matrix of possible states (or locations), one of which is the starting state, marked with 'S', and one is the goal state, marked with 'G'. The states marked with 'B' are barriers and cannot be entered by the agent. All the states are distinct and distinguishable by their x-y coordinates in the matrix. From each state, the agent can take four possible actions: UP, RIGHT, DOWN, and LEFT. Each action changes the location accordingly, except where such a movement would take the agent into a barrier or outside the maze, in which case the location is not changed. Since the experiment targets the learning performance of the agent with a variable number of lookahead levels, the Q-Learning parameters are kept constant for all instances: The step-size parameter $\alpha = 0.1$, and the discount rate $\gamma = 0.9$. The lookahead-weight factor is also constant and set to a moderate value between 0 and 1, $\lambda = 0.5$. The number of lookahead levels is variable: $N = 1, 2, 3$, and 4.

To exercise the MLL technique in different environments, experiments were done with two different mazes. Both mazes are 12 by 11 but have different barrier layouts: The first maze (Environment 1) in Figure 4 has an arbitrary barrier layout, and the second one (Environment 2) in Figure 7 has an occluded barrier layout.

Results for each maze are illustrated in two different instances. Each instance shows a plot of the number of steps per learning path versus the learning path number for each of the four values of N . The agent's initial path that results in the initial knowledge of the environment is the same for all four values of lookahead levels in each instance. Having the same initial path in the same maze ensures that any changes in the learning performance are due to the change in the number of lookahead levels. However, the agent's initial path is different among the different instances to ensure a different learning experience in each instance.

The plots of the actual experimental data show the number of steps per learning path decreases as the agent does more learning paths. They also show the greater N is, the faster the number of steps per learning path reaches the

minimum (optimal path). Least-Squares Exponential curve fitting was then applied on the experimental data to create more legible plots and be able to do a statistical analysis. The exponential curve is of the form $f(x) = Ae^{Bx}$.

We introduce here a new statistical analysis measure using a method that calculates the area under curve for a value of N and compares it to the area for another value of N. Since all curves have a fixed range of learning paths ($0 \leq x \leq L$, where L is the maximum number of learning paths), the area represents the overall magnitude of the number of learning steps per learning path ($f(x)$). Hence, a smaller area means a better performance. The equations in (21) and (22) illustrate the described method.

$$f_N(x) = A_N e^{B_N x}$$

$$area_N = \int_0^L A_N e^{B_N x} dx = \frac{A_N}{B_N} (e^{B_N L} - 1) \quad (21)$$

$$\text{Improvement Rate (from N1 to N2)} = 1 - \frac{area_{N2}}{area_{N1}} \quad (22)$$

Performance statistics are done for each set of data, using the method described above. The area and improvement rate values are in tables corresponding to the plots directly above them. Each table shows the improvement rate between consecutive values of N.

Figures 5.a and 5.b and Figures 6.a and 6.b correspond for Environment 1. Figures 8.a and 8.b and Figures 9.a and 9.b correspond for Environment 2. The results show the more lookahead levels used, the faster the agent is learning. Each graph shows that in any learning path, the agent needed less learning steps to reach the goal when $N = 2$ than when $N = 1$, when $N = 3$ than when $N = 2$, and when $N = 4$ than when $N = 3$. The improvement rates vary from an instance to another because of differences in the initial random path as the latter affects the initial exploration of the agent and its learning processes afterwards.

4.1 Environment 1: Arbitrary Barrier Layout

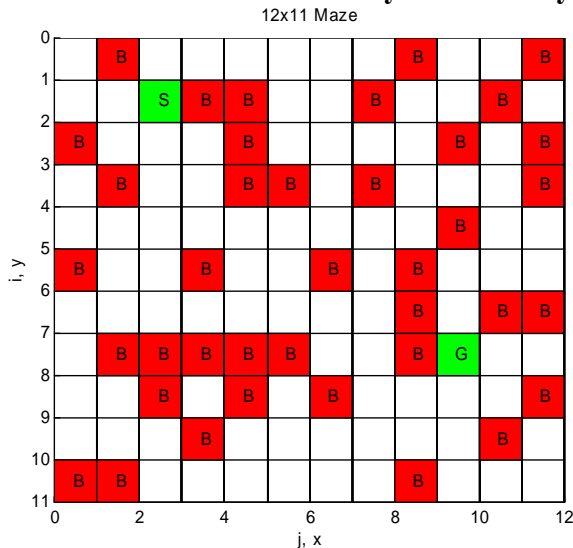


Figure 4 – Environment 1: Arbitrary Barrier Layout

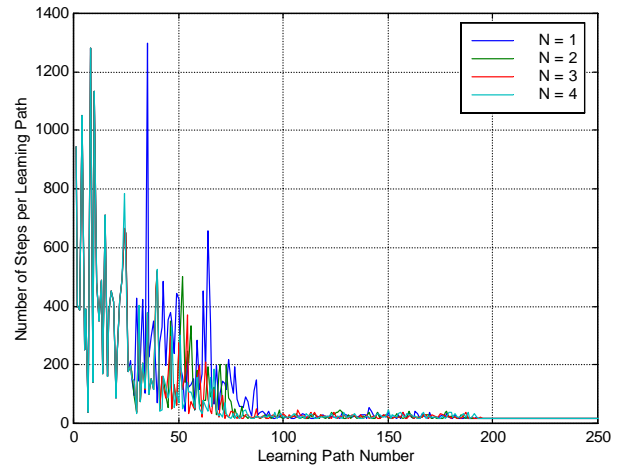


Figure 5.a – Actual Data for N=1,2,3,4 (1st Instance)

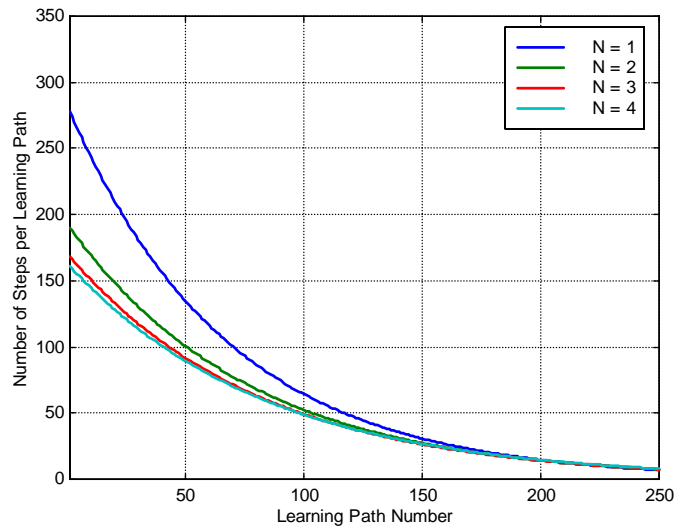


Figure 5.b – Curve-Fit Data for N=1,2,3,4 (1st Instance)

Table 1 – Performance Statistics (1st Instance)

| | N=1 → N=2 | N=2 → N=3 | N=3 → N=4 |
|------------------------|---------------|---------------|---------------|
| Area ($\times 10^4$) | 1.859 → 1.422 | 1.422 → 1.310 | 1.310 → 1.286 |
| Improvement Rate | 23.507 % | 7.876 % | 1.832 % |

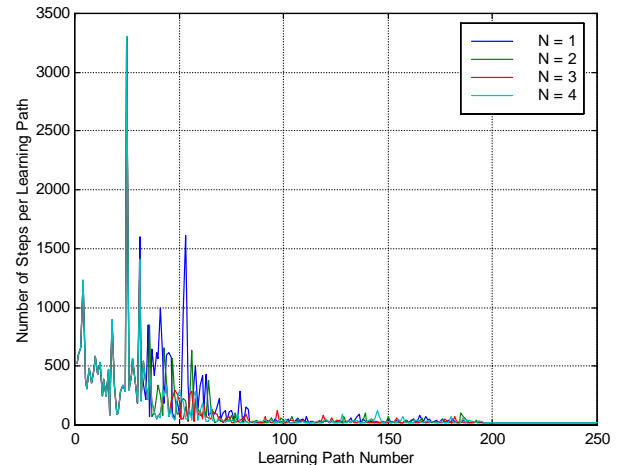


Figure 6.a – Actual Data for N=1,2,3,4 (2nd Instance)

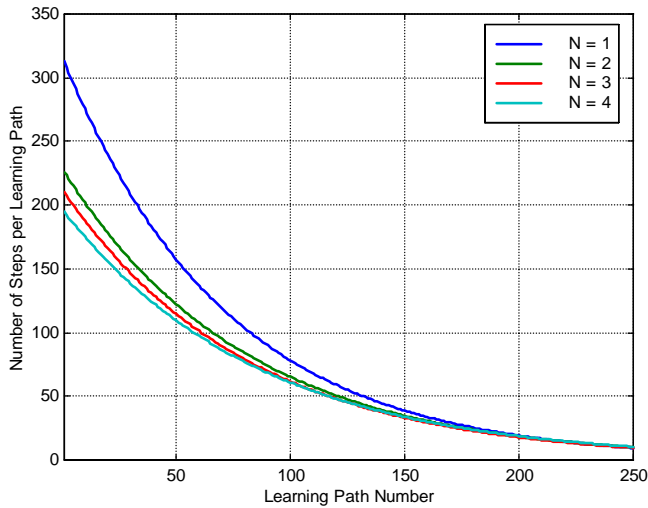


Figure 6.b – Curve-Fit Data for N=1,2,3,4 (2nd Instance)

Table 2 – Performance Statistics (2nd Instance)

| | N=1 → N=2 | N=2 → N=3 | N=3 → N=4 |
|------------------------|---------------|---------------|---------------|
| Area ($\times 10^4$) | 2.196 → 1.744 | 1.744 → 1.641 | 1.641 → 1.588 |
| Improvement Rate | 20.583 % | 5.906 % | 3.230 % |

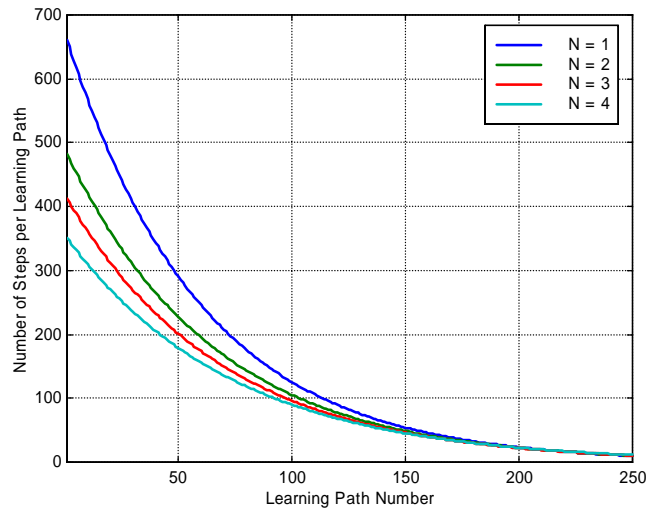


Figure 8.b – Curve-Fit Data for N=1,2,3,4 (3rd Instance)

Table 3 – Performance Statistics (3rd Instance)

| | N=1 → N=2 | N=2 → N=3 | N=3 → N=4 |
|------------------------|---------------|---------------|---------------|
| Area ($\times 10^4$) | 3.948 → 3.124 | 3.124 → 2.783 | 2.783 → 2.510 |
| Improvement Rate | 20.871 % | 10.916 % | 9.810 % |

4.2 Environment 2: Occluded Barrier Layout

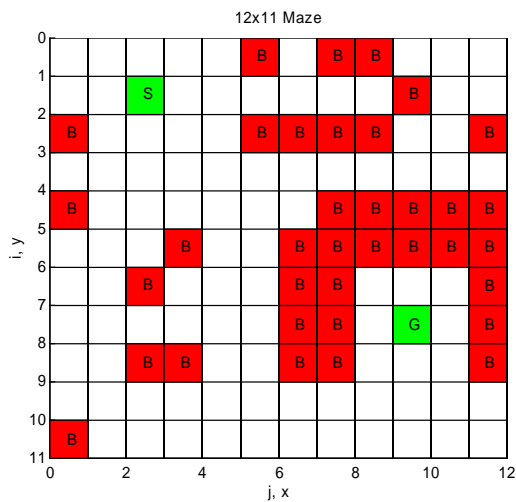


Figure 7 – Environment 2: Occluded Barrier Layout

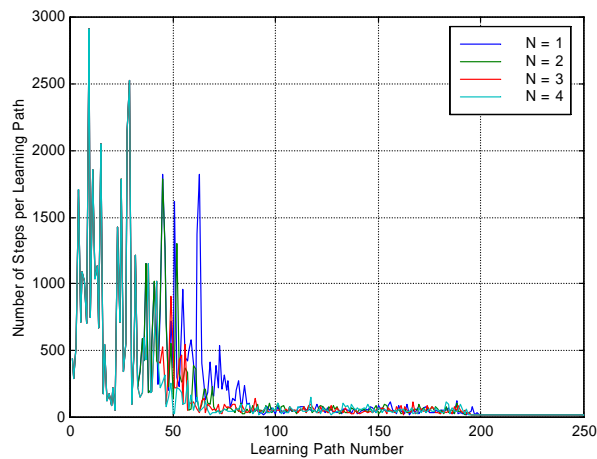


Figure 8.a – Actual Data for N=1,2,3,4 (3rd Instance)

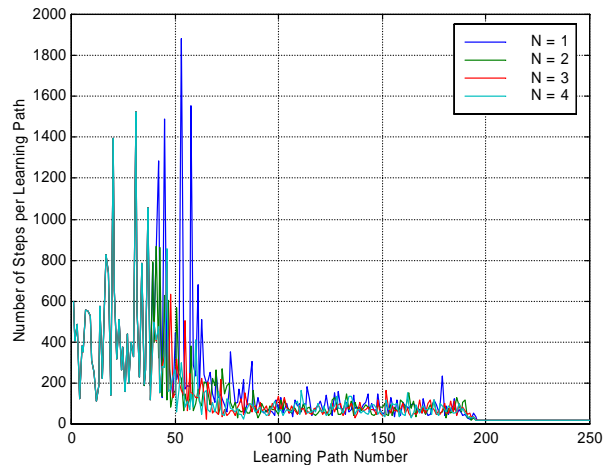


Figure 9.a – Actual Data for N=1,2,3,4 (4th Instance)

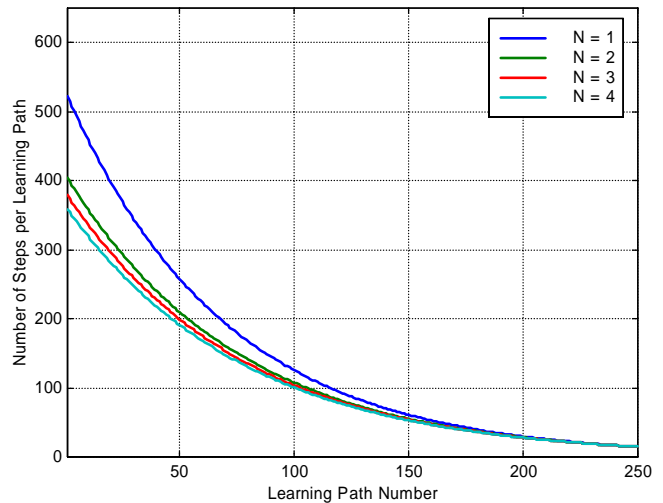


Figure 9.b – Curve-Fit Data for N=1,2,3,4 (4th Instance)

Table 4 – Performance Statistics (4th Instance)

| | N=1 → N=2 | N=2 → N=3 | N=3 → N=4 |
|-------------------------------|-------------|-------------|-------------|
| Area (×10⁴) | 3.571→2.949 | 2.949→2.816 | 2.816→2.709 |
| Improvement Rate | 17.418 % | 4.510 % | 3.800 % |

6 Conclusion and Future Work

In this paper, the focus was on presenting the MLL technique and proving its faster convergence. The experimental results showed the potential of this proposed technique and how the agent learns faster when more lookahead levels are used regardless of the initial exploration in the environment.

The state and action spaces in the environment are small, allowing tables or matrices to be used for all learning processes. Such environment does not require a lot of computation resources, and high numbers of lookahead levels can be used. As environments increase in size, gigantic computation resources may be required to a point where the benefit of higher numbers of lookahead levels may be offset by the amount of time and memory consumed by the autonomous agent or system. In the MLL reinforcement learning technique, the number of lookahead levels N is to be chosen as a tradeoff between available computation resources and desired learning speed. It is recommended to further study this matter and develop an algorithm or methodology to choose the optimal N. That value of N should vary depending on a number of variables, most important of which are the amount of available resources, the desired learning speed, and the type of environment the application is dealing with.

Future work will include further study and analysis of the MLL technique and demonstrate how it can be implemented in an intelligent chip.

7 References

- [1] A. Angelo, D. Florence, et al., “Efficient Learning of Variable-Resolution Cognitive Maps for Autonomous Indoor Navigation”, *IEEE Transactions on Robotics and Automation*, 1999.
- [2] A. G. Barto, R. S. Sutton, & C. J. C. H. Watkins, “Learning and Sequential Decision Making”, *Learning and Computational Neuroscience*, pp. 539-602, 1990.
- [3] C. J. C. H. Watkins, “Learning with Delayed Rewards”, *PhD Thesis Cambridge University Psychology Department*, 1989.
- [4] C. Watkins & P. Dayan, “Q-Learning”, *Machine Learning*, vol. 8, pp. 279-292, 1992.
- [5] D. B. Megherbi, A. Teirelbar, & A. J. Boulenouar, “A Time-Varying-Environment Machine Learning technique for Autonomous Agent Shortest Path Planning”, *Proceedings of the SPIE International Conference on Defense Sensing*, pp. 419-428, Unmanned Ground vehicle Technology, Orlando, Florida, April, 2001.
- [6] Ernst, D., Geurts, E., & Wehenkel, L., “Tree-Based Batch Mode Reinforcement Learning”, *Journal of Machine Learning Research*, April 2005.
- [7] Erwin Kreyszig, “Advanced Engineering Mathematics”, *John Wiley & Sons Inc.*, Seventh Edition, 1993.
- [8] Murphy, S. A., “A Generalization Error for Q-Learning”, *Journal of Machine Learning Research*, July 2005.
- [9] P. Dayan, “The Convergence of TD(λ) for General λ ”, *Machine Learning*, vol. 8, pp. 341-362, 1992.
- [10] R. Maclin & J. W. Shavlik, “Creating Advice-taking Reinforcement Learners”, *Machine Learning*, vol. 22, 251-281, 1996.
- [11] R. Riolo, “Lookahead Planning and Latent Learning in a Classifier System”, *Proceedings of the Int. Conf. on the Simulation of Adaptive Behavior*, 1991.
- [12] R. S. Sutton & A. G. Barto, “Reinforcement Learning: an Introduction”, *MIT Press*, 1998.
- [13] R. S. Sutton, “Dyna, an Integrated Architecture for Learning, Planning, and Reacting”, *Working Notes of 1991 AAAI Spring Symposium*, pp. 151-155, 1991.
- [14] R. S. Sutton, “Integrated Architectures for Learning, Planning, and Reaction Based on Approximating Dynamic Programming”, *Proceedings of the Seventh Int. Conf. on Machine Learning*, pp. 216-224, 1990.
- [15] T. M. Mitchell, “Machine Learning”, *McGraw-Hill*, 1997.