MDPs, Policy Iteration, Q-Learning and SARSA

Kirsten Odendaal
College of Computing
Georgia Institute of Technology

1 Introduction

Reinforcement learning (RL) provides an effective and practical framework for solving real-world decision-making problems, such as optimizing traffic intersections and navigating warehouse robots. These tasks require balancing many conflicting constraints, namely reducing congestion while maintaining smooth operations or ensuring safe, efficient deliveries. RL achieves this by enabling agents to learn optimal behaviours through interactions with the environment. This study explores Markov Decision Process (MDP) models for traffic control and warehouse navigation, applying both planning methods (*Value Iteration, Policy Iteration*) and model-free methods (*Q-Learning, SARSA*). A detailed analysis and comparison is conducted to investigate the strengths and limitations of each and also the trade-offs in terms of computational efficiency and learning performance.

2 Technical Background

RL is a computational approach under the artificial intelligence umbrella, where agents learn optimal behaviours through interaction with an environment. This section outlines the technical foundations of RL, including MDPs, algorithms, and exploration strategies.

2.1 Markov Decision Processes

MDPs provide the mathematical framework for modelling sequential decision-making in RL. An MDP is defined by a tuple (S, A, R, P, γ):

- *State Space* (S): Represents all possible configurations of the environment.
- *Action Space* (A): Represents all actions available to the agent.
- *Reward Function* ($R(s, a, s')$): The immediate reward received after a state-action transition.
- *Transition* ($P(s'|s, a)$): Probability of reaching state (s') from state (s), after action (a).
- *Discount Factor* (γ): Balances the importance of immediate and future rewards.

MDPs assume the Markov property, meaning the next state depends only on the current state and action, not the sequence of past states (Sutton & Barto, 2018; Morales, 2020). This critical assumption allows for computational tractability.

2.2 Reinforcement Learning Algorithms

RL algorithms aim to find an optimal policy (π^*), mapping states to actions that maximize cumulative rewards. These algorithms fall into two categories:

1. *Model-Based Algorithms:* Require knowledge of the environment's dynamics to plan optimal policies. These methods often involve computing state values or policies directly before execution.
2. *Model-Free Algorithms:* Learn optimal policies iteratively through interaction without prior knowledge. They update value functions or policies iteratively based on observed rewards.

Model-Based (Planning) Methods

1. *Value Iteration:* Computes the optimal policy by iteratively updating state values $V(s)$ based on the Bellman optimality equation:

$$V(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V(s')]$$

After convergence, the policy is extracted by selecting the action that maximizes the expected value. It serves as a baseline due to its guaranteed convergence for fully-defined MDPs. (Sutton & Barto, 2018).

2. *Policy Iteration:* Alternates between *Policy Evaluation* and *Policy Improvement*. Policy Evaluation computes $V(s)$ for a fixed policy π.

$$V_\pi(s) = \sum_a \pi(a|s) \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V_\pi(s')]$$

   Policy Improvement updates the policy $\pi(s)$ to greedily maximize the expected value.

$$\pi'(s) = \arg\max_a \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V_\pi(s')]$$

   The process repeats until the policy converges to the optimal π^*. Policy iteration often converges in less policy updates than value iteration, although each iteration usually needs more computation for policy evaluation (Sutton & Barto, 2018).

Model-Free Methods

1. *Q-Learning:* An off-policy algorithm that updates the action-value function $Q(s,a)$ using the maximum estimated future reward, independent of the agent's current policy:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[R + \gamma \max_a Q(s',a') - Q(s,a)]$$

   By learning from exploratory actions while optimizing a greedy policy, Q-Learning converges to the optimal policy (Sutton & Barto, 2018).

2. *SARSA (State-Action-Reward-State-Action):* An on-policy algorithm that updates $Q(s,a)$ using the action actually taken under the current policy:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[R + \gamma Q(s',a') - Q(s,a)]$$

   Since SARSA follows the policy being executed, it tends to produce smoother and more stable policies in stochastic environments (Morales, 2020; Sutton & Barto, 2018).

2.3 Exploration versus Exploitation

RL faces the challenge of balancing exploration (trying new actions) and exploitation (choosing known optimal actions to maximize reward). The ϵ-greedy with exponential decay method addresses this by selecting random actions with probability ϵ, which decays exponentially over time to favour exploitation as learning continues.

$$\epsilon_t = \epsilon_{min} + (\epsilon_0 - \epsilon_{min}) \cdot e^{-kt}$$

where ϵ_0 and ϵ_{min} is the initial and minimum exploration rates and k controls the decay rate. This approach allows for exploration early in training while gradually shifting toward exploitation as the agent learns the environment. Implementation was done considering the methodology outlined by Morales (2020).

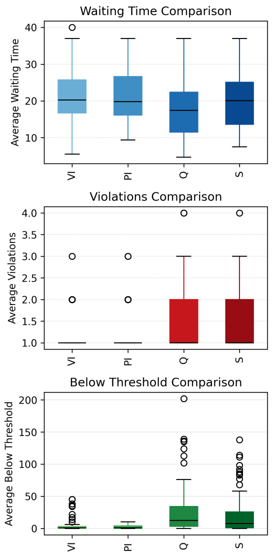
3 Problem Definitions

This section defines two simplified problems modelled as Markov Decision Processes: traffic intersection control and warehouse robot navigation. These problems give a basis for evaluating the RL algorithms. The traffic intersection problem explores optimizing traffic light timings to reduce congestion and maintain flow. The warehouse robot navigation problem focuses on enabling autonomous robots to transport items while avoiding collisions and conserving energy. Table 1 outlines each MDP problem in further detail.

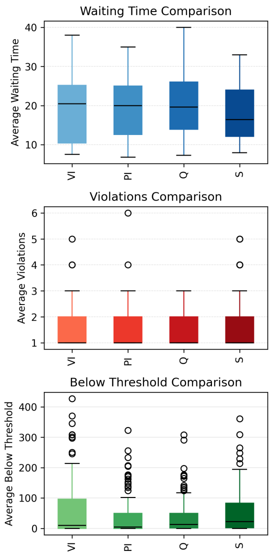
It should be noted that the environment assumes simplified dynamics, such as no left/right turns, fixed maximum car movements for traffic intersections, and a grid-world representation with stochastic movement for the warehouse robot. These simplifications are required for manageable computations but do not completely capture the complexity of real-world scenarios.

Component	Traffic Intersection Problem	Warehouse Robot Problem
State Space (S)	Encodes the number of cars queued in each direction (N-S and E-W) and the current traffic light configuration. (c_{NS}, c_{EW}, l) where: • c_{NS}: Number of cars queued in N-S direction. • c_{EW}: Number of cars queued in E-W direction. • l: Traffic light status (0 for N-S green, 1 for E-W green).	Represents the robot's position in a grid world, locations of static obstacles, and nearby moving workers. (x, y, b) where: • r_x, r_y: Robot's coordinates in the grid. • b: Binary variable for *bump* status of the robot: *wall, equip., worker, target, step*. • t_x, t_y: target coordinates in the grid.
Action Space (A)	• Maintain current light configuration. • Switch the light configuration.	• Move up, Move down. • Move left, Move right.
Transition Dynamics ($P(s'\|s, a)$)	Determined by current light configuration, car arrivals (Poisson process), and cars passing through the intersection. • $l = 0$ for Green N-S, cars pass in that direction; cars queue in other direction (E-W).	Stochastic transitions due to environ. hazards. • Probability of deviating intended direction. • Collisions lead to bouncing back to the previous state. • Collisions with workers terminate the episode.
Reward Function ($R(s, a)$)	• $(+)$ clear cars from the intersection. • $(-)$ total car count exceeds critical thresholds.	• $(+)$ delivering a box. • $(-)$ collisions with objects. • $(-)$ each step to incentivize efficiency.
Discount Factor (γ)	Balances trade-offs between immediate and long-term congestion reduction.	Balances trade-offs between minimized collisions and optimized deliveries.

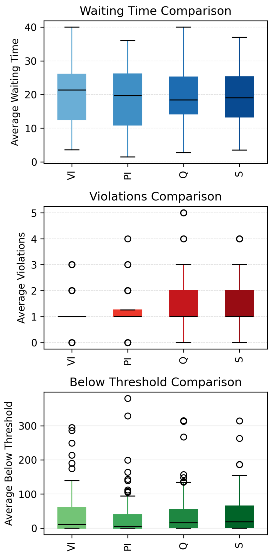
Table 1: MDP Components for Traffic Intersection and Warehouse Robot Problems



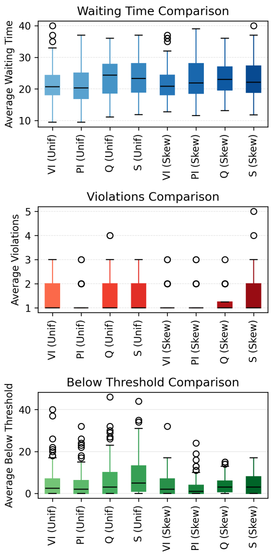
(a) E1



(b) E2



(c) E3



(d) E4

Figure 1: Traffic experimental results for each metric of interest

4 Methodology

The study systematically evaluates various RL algorithms across the two domain problems. The experiments conducted are detailed in Table 2. Baseline evaluations establish performance benchmarks, followed by experiments exploring variations in reward shaping, terminal conditions, and system parameters. The metrics assessed are chosen to capture task-specific objectives, such as reducing congestion or maximizing deliveries, and general reinforcement learning goals, like policy convergence and robustness.

5 Experiments and Results

5.1 Traffic Intersection Analysis

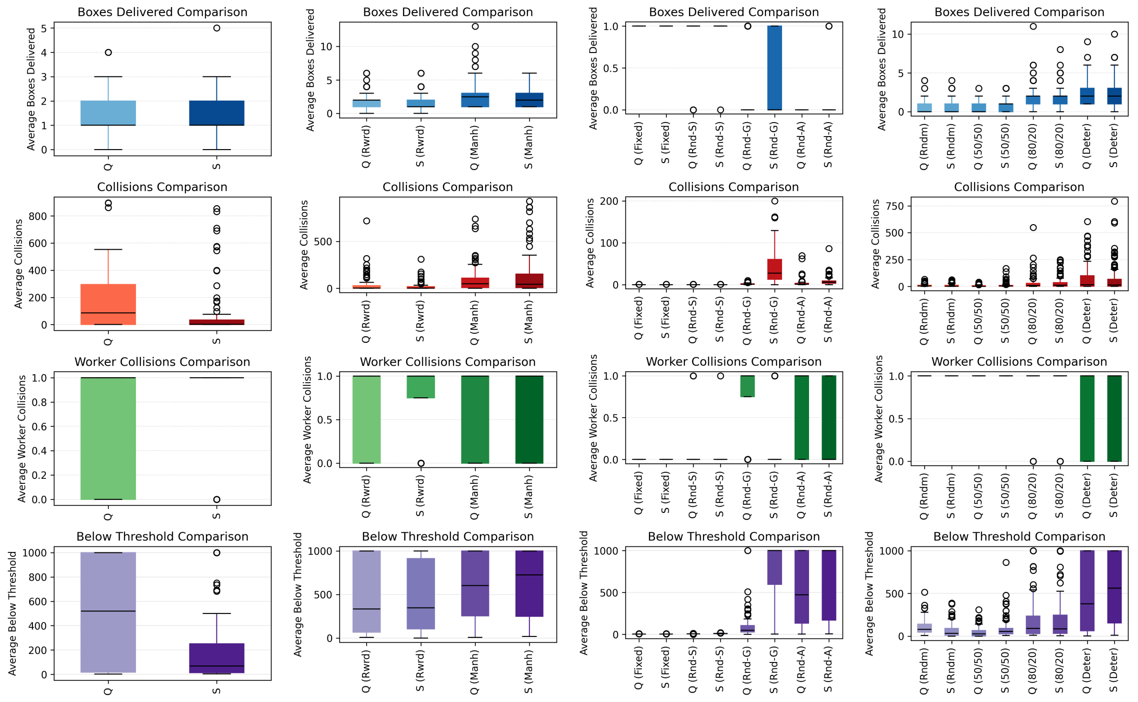
This section explores the traffic problem and how different RL algorithms respond to reward structures, terminal conditions, and traffic patterns. A summary of the collected results for each experiment can be seen visualized in Figure 1. Learning curve visualizations can be found in Appendix C.

Exp.	Description	Variations	Metrics Evaluated
	Traffic Intersection		
E1	Baseline evaluations using default parameters: • Penalize number of waiting cars • Terminate on exceeding directional ($M = 20$) or total ($N = 30$) limits • ϵ-greedy exponential decay function implemented	• max evals: 100 • episodes: 2000 • max steps: 1000 • gamma: 0.90 • alpha: 0.1 • epsilon: 1.0 • epsilon decay: 0.99 • theta: 1e-10 • $r_{exceed} = -50$ • $r_{wait} = -1 * c_{tot}$	1. **Average queue length:** Average amount of cars waiting at the intersection. 2. **Congestion events:** Number of instances in which the car count exceeds the total number of N cars or M in a single direction. 3. **Max clearance time:** maximum continuous time during which the the intersection remains below the critical thresholds.
E2	Reward shaping for congestion and queue reduction: • Penalize high congestion • Reward queue reduction • Incentivize green light with the largest queued direction	• $r_{golight} = 10$ • $r_{nolight} = -5$ • $r_{c_{tot}<=5} = 50$ • $r_{c_{tot}<=15} = 25$ • $r_{c_{tot}<=25} = 10$ • $r_{c_{tot}>25} = -10$	
E3	Terminal condition exploration: • Episode ends on clearance with high reward	• $r_{cleared} = 100$	
E4	Traffic Poisson distribution exploration: • Uniform distribution • Highly skewed distribution	• $\lambda_{NS} = 3, \lambda_{EW} = 3$ • $\lambda_{NS} = 5, \lambda_{EW} = 2$	
	Warehouse Robot		
E1	Baseline evaluations using default parameters: • Penalize number of steps per episode • Reward box delivery • ϵ-greedy exponential decay function implemented • Terminate on colliding with a worker • Probability of successful action is always achieved	• max evals: 100 • episodes: 2000 • max steps: 1000 • gamma: 0.90 • alpha: 0.1 • epsilon: 1.0 • epsilon decay: 0.99 • theta: 1e-10 • $r_{step} = -0.1$ • $r_{deliver} = 100$	1. **Deliveries per episode:** Average number of boxes successfully delivered. an episode. 2. **Collision rates:** Number of instances the robot collides with obstacles. 3. **Worker collision rates:** Number of instances the robot causes harm to a worker. 4. **Uninterrupted operation time:** Longest uninterrupted duration the robot functions without any collision.
E2	Reward shaping for safety and efficiency: • Penalize collisions with increasing severity • Penalize extreme distances using manhattan (L_1) • Reward proximity to drop-off	• $r_{wall} = -1$ • $r_{equipment} = -5$ • $r_{worker} = -20$ • $r_{man} = -0.1 * d_{L1}$	
E3	Terminal condition exploration: • Episode ends on delivery or worker impact for: (1) fixed start-fixed goal, (2) random start-fixed goal, (3) fixed start-random goal, (4) random start-random goal	• same rewards as E2	
E4	Reliability under stochastic actions: • Random action • Coin-toss action • Almost always action • Deterministic action	• $P_{random} = 0.1$ • $P_{50/50} = 0.5$ • $P_{80/20} = 0.8$ • $P_{determ.} = 1.0$	

Table 2: Experiment summary for both Traffic and Warehouse Robot problems

Impact of Reward Shaping

Reward shaping improved agent behaviour across all algorithms by encouraging congestion reduction and intersection clearing. Model-based algorithms (Value Iteration, Policy Iteration) adapted quickly, resulting in higher cumulative rewards and more efficient traffic flow. These algorithms consistently prioritized clearing the intersection, as shown by the large increase in time spent below congestion thresholds. Whereas the Model-free algorithms (Q-Learning and SARSA) showed improved performance under reward shaping, but their learning curves revealed greater variability. The tiered reward system gave these agents clearer guidance, balancing short-term rewards for congestion reduction with long-term goals for sustained flow. Ultimately, reward shaping acted as a sort-of guide or curriculum, simplifying learning and aligning exploration with desired goals, particularly for model-free algorithms (Sutton & Barto, 2018). Without reward shaping, agents struggled to develop consistent policies, often performing strange light-switching phenomena. Default rewards often failed to differentiate between subtle but critical policy decisions. By introducing structured rewards, the agents are guided toward desirable behaviours, effectively shrinking required exploration and accelerating learning of the agents.



(a) E1 (b) E2 (c) E3 (d) E4

Figure 2: Robot experimental results for each metric of interest

Effect of Terminal Conditions

Introducing terminal conditions, such as ending episodes upon clearing the intersection, revealed large performance differences. The early termination reduced cumulative rewards, as episodes were trimmed before reaching their full potential for maximizing returns. Under these constraints, model-based algorithms excelled in minimizing waiting times and showcasing their ability to prioritize short-term goals. Meanwhile, model-free algorithms adapted less effectively, although they did show slight reductions in the number of violations. These observations highlight the advantages and limitations of treating intersection clearing as a terminal condition. On the one hand, terminal conditions simplify objectives, allowing agents to focus on immediate tasks like reducing congestion. On the other hand, they may potentially restrict the optimization of long-term optimization as policies become overly biased on the resets (Morales, 2020).

Intersection with Varying Traffic Patterns

Experiments with balanced and skewed traffic patterns tested the adaptability of each RL algorithm. In the uniform traffic case, the lack of natural directional bias made it difficult for agents to prioritize actions, resulting in slower convergence and higher average waiting times. This phenomenon increases the difficulty of learning clear policies. Meanwhile, for the highly skewed traffic distribution, the strong directional bias overwhelmed agents, particularly model-free ones, leading to skewed policies that neglected the less busy direction, leading to poor generalization. Additionally, the unbounded nature of Poisson arrivals increases this issue, as extreme traffic bursts often overwhelmed learned strategies of the agents.

5.2 Warehouse Robot Analysis

The warehouse robot problem analyzed only Q-Learning and SARSA algorithms. The section highlights key findings into the algorithm performance and adaptability, while presenting challenges specific to varying reward structures, stochasticity, and terminal conditions. A summary of the collected results for each experiment can be seen visualized in Figure 2. Furthermore, learning curve visualizations can be found in Appendix C.

Impact of Reward Shaping

Reward shaping influenced agent behaviour by encouraging goal-directed movement and penalizing collisions. This is clearly evidenced by introducing a Manhattan distance reward, helping to encourage movement toward the goal while penalizing collisions with walls, equipment, and workers. Q-Learning's off-policy updates allowed for broader exploration, enabling around three deliveries per episode while also reducing the amount of collisions. SARSA became overly cautious, appearing to prioritize penalty avoidance over goal-reaching. This is due to its on-policy nature, where incremental updates and penalties are immediately felt during exploration (Sutton & Barto, 2018). Therefore, the newly shaped rewards highlighted a trade-off: Q-Learning benefited from exploration, achieving efficiency gains, while SARSA's cautious updates restricted its ability to explore robust policies. While the manhattan distance metric showed positive contributions, further reworking of the function to a more potential-driven function (ψ) could be explored (Russell & Norvig, 2022).

Effect of Terminal Conditions

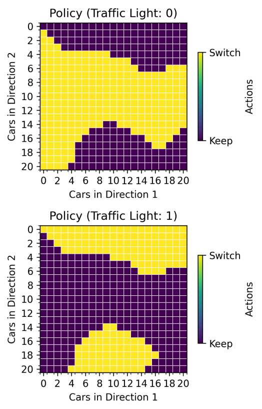
Both algorithms performed effectively in the *Fixed Start & Goal* scenario as the deterministic environment stopped the need for exploration. Both models also showed strong performance in the *Random Start & Fixed Goal* case by quickly learning the task. However, when the goal position was randomized, the algorithms struggled. SARSA frequently timed out due to insufficient exploration, failing to adapt to the changing goals. Q-Learning had higher collision rates as it struggled with the enlarged state-action space. *Random Start & Goal* was the most challenging, where both agents experienced significant performance degradation due to the substantial expansion of the state-action space. SARSA's step-by-step updates struggled to make immediate progress, while Q-Learning's strategies were also ineffective. However, given sufficient learning opportunities (more episodes), model-free methods can still converge to effective policies even in complex environments (see Section 6.1). This highlights the importance of balancing adaptability and caution (Morales, 2020).

Reliability in Deterministic vs. Stochastic Settings

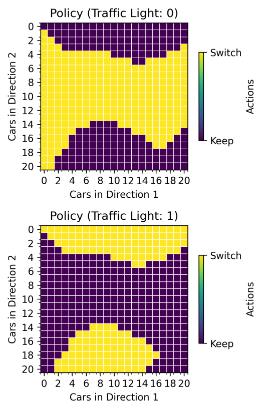
Varying stochasticity in the environment showed interesting effects on performance. While both algorithms made more box deliveries in the deterministic settings, they also showed a significant increase in time-outs. This suggests that the deterministic environment led models to adopt suboptimal policies, often resulting in infinite loops, as they consistently executed the same actions cycles without success (Morales, 2020). Introducing moderate randomness (80/20) slightly decreased deliveries but greatly reduced time-outs by forcing exploration, helping agents escape unproductive cycles and lowering collisions. However, increasing randomness beyond this threshold proved damaging, as the models struggled to reliably handle excessive variability. This lead to very poor performance, highlighting the difficulty to produce optimal policies in overly stochastic environments.

6 Policy Comparisons

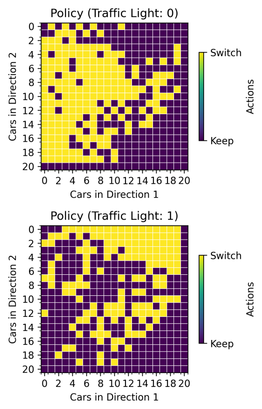
Extracted policies from traffic and warehouse experiments show clear distinctions in smoothness, adaptability, and consistency between model-based and model-free algorithms. Figure 3 visualizes the policies learned by Value Iteration, Policy Iteration, Q-Learning, and SARSA for the traffic intersection problem. The model-based algorithms (Value Iteration and Policy Iteration) produce smooth and consistent policies. This smoothness stems from their direct computation of policies over the entire MDP (Sutton & Barto, 2018). Q-Learning and SARSA produce noisier, less structured policies due to their experience-driven Q-value updates. Model-free methods need extensive exploration and longer training to refine policies, particularly in environments with sparse rewards or stochastic transitions (Morales, 2020). Terminal states in model-free policies appear as blank regions, as exploration stops once terminal conditions are reached, whereas model-based algorithms provide continuity by defining actions in terminal states. Environmental factors, such as traffic



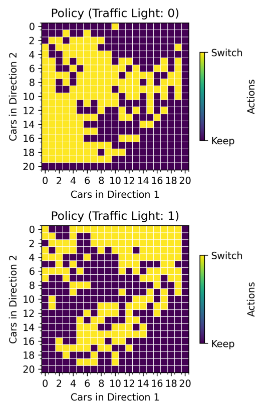
(a) Value Iteration



(b) Policy Iteration

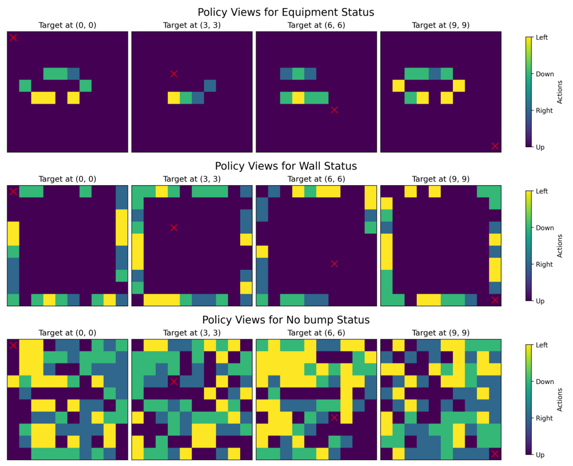


(c) Q-Learning

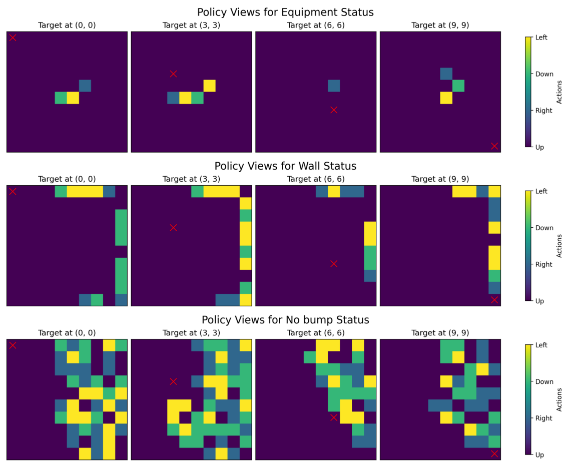


(d) SARSA

Figure 3: Reconstructed policy using learned Q-table for Traffic experiment #2



(a) Q-Learning



(b) SARSA

Figure 4: Reconstructed policy using learned Q-table for Robot experiment #2

distributions, shape the policies. Imbalanced traffic flows result in prioritized actions for busier directions, while uniform distributions produce more evenly distributed patterns.

Figure ?? shows the policies for the warehouse robot problem, demonstrating the navigation strategies learned by Q-Learning and SARSA for various conditions. Each target generates unique policy distributions influenced by localized rewards and constraints such as obstacles and stochastic deviations. The grid world's exponential growth of the state-action space shows the challenges of scaling complexity. Q-Learning's off-policy method supports broader exploration, allowing for denser policy coverage, while SARSA's on-policy updates neglect states not visited under its current policy (Sutton & Barto, 2018). This gap highlights why SARSA can become overly cautious, especially with strict terminations or sparse rewards. Therefore, tuning exploration parameters is crucial for preventing incomplete coverage and balancing efficiency against risk.

6.1 Hyperparameter Optimization

The hyperparameter optimization process revealed important trade-offs between computational efficiency and training complexity, as summarized in Table 3. The discount factor (γ) showed a significant impact, where lower values ($\gamma = 0.5$) encouraged pursuit of short-term rewards, while higher values ($\gamma = 0.9$) favoured long-term planning (Sutton & Barto, 2018). For the traffic problem, lower γ globally improved cumulative rewards by prioritizing near-term objectives, suggesting the

Traffic Intersection									
Parameters		γ		θ		ϵ		*Eps*	
Ranges		**0.5**	**0.9**	**1.00E-02**	**1.00E-10**	**0.5**	**0.99**	**5000**	**10000**
VI	reward	750.7	640.8	562.7	640.8				
	time	3.1	15.7	6.1	15.7				
PI	reward	535.4	510.6	362	510.6				
	time	3.5	17.4	4.7	17.4				
Q	reward	618.9	365.8			462.2	463	463	510.2
	time	9.1	7.3			6.5	8.2	8.2	37.4
S	reward	631.4	495.3			614.3	533.6	556.6	552.8
	time	5.2	4.0			2.8	4.1	11.3	15.5
Warehouse Robot									
Parameters		γ		θ		ϵ		*Eps*	
Ranges		**0.5**	**0.9**	**1.00E-02**	**1.00E-10**	**0.5**	**0.99**	**5000**	**10000**
Q	delivery	2.6	3.1			1.8	2.3	13.2	47.7
	time	492.5	442.1			372.8	477.9	1720.4	4384.2
S	delivery	2.2	2.1			2.1	2.4	24.5	57.4
	time	395.4	413.7			407.7	540.6	2967.1	9314.2

Table 3: Hyperparameter results summary

importance of short-term decision-making in such an environment. Convergence thresholds (θ) for Value Iteration and Policy Iteration balanced precision and efficiency, with lower θ values ($1E-02$) accelerating convergence but producing less optimal policies. For model-free methods, the exponential decay factor (ϵ) controls the exploration-exploitation balance. Faster decay ($\epsilon = 0.5$) stabilized policies quickly, while slower decay ($\epsilon = 0.99$) supported broader exploration, particularly beneficial in dynamic environments like the warehouse robot problem. The number of training episodes (*Eps*) had a massive impact on Q-Learning and SARSA. Extended training (10,000 episodes) dramatically improved performance for the warehouse robot by enabling exploration of sparse state-action spaces at a consequence of significant computational cost. While obvious for the robot scenario, such improvements were not observed for the traffic problem. This suggests that further exploration provided diminishing returns.

Ultimately, model-based algorithms demonstrated faster convergence and efficiency but relied on complete transition and reward models, making larger problem intractable. Meanwhile, despite their higher computational demands, model-free methods were essential for exploring and adapting to environments with incomplete information. These outcomes provide an indication of the importance of systematic hyperparameter tuning to balance efficiency and effectiveness across diverse problem domains. This experiment focused on single hyperparameter variations while keeping others fixed at defaults. Therefore, any potential cross-interactions and their impacts were not explored.

7 Conclusions and Future Work

This study compared model-based (*Value Iteration, Policy Iteration*) and model-free (*Q-Learning, SARSA*) reinforcement learning algorithms to traffic intersection control and warehouse robot navigation. Model-based methods demonstrated quicker convergence and globally consistent policies by using complete transition and reward models, but proved less scalable for high-dimensional problems. Model-free methods required more extensive computation and exploration but were critical for environments with unknown dynamics. This highlighted their adaptability to real-world scenarios where perfect models do not exist. In both problems, hyperparameter choices shaped policy effectiveness, convergence speed, and the balance between short- and long-term objectives. Key parameters such as the discount factor, convergence thresholds, and exploration decay had noticeable impacts. For example, shorter horizons benefited near-term traffic optimization, whereas extended training greatly improved performance in the more complex warehouse setting. These results emphasize the importance of systematic hyperparameter tuning and reward design to achieve robust policies. Looking ahead, advanced exploration strategies and further hyperparameter tuning will be crucial for handling increased complexity of large-scale traffic systems and collaborative warehouse tasks.

References

M. Morales. *Grokking Deep Reinforcement Learning*. Manning, Shelter Island, NY, 1st edition, 2020.

Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, 4th edition, 2022.

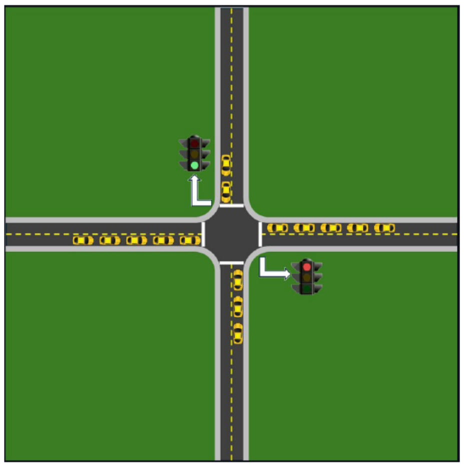
Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, MA, 2nd edition, 2018.

A Available Repository

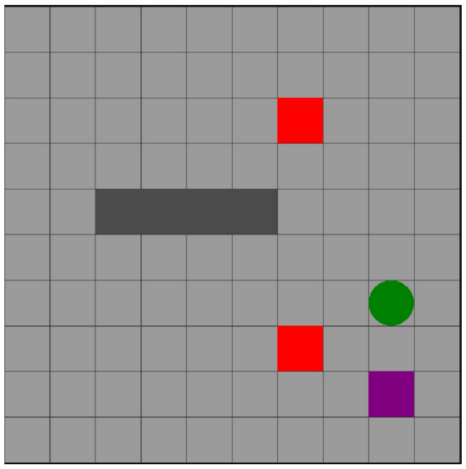
All code and supplementary materials used in this research are available in a GitHub repository. The repository can be accessed at: `https://github.gatech.edu/gt-omscs-rldm/7642RLDMSpring2025kodendaal3` where the latest commit hash is:

B Problem Environments

Two examples of the graphical environment renderings.



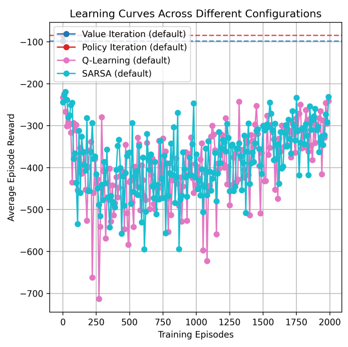
(a) Illustration of the traffic intersection problem simulation. The light above the intersection indicates if it is green or red for North/South traffic, while the light on the right shows the red or green status for East/West traffic. Vehicles enter and leave in accordance to a probability distribution



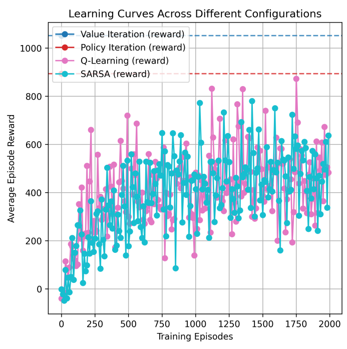
(b) Depiction of the warehouse robot simulation. Light gray squares are open areas on the floor grid, dark gray squares indicate equipment, red squares mark a worker's presence, the green circle symbolizes the robot, and the purple square signifies the target for box delivery.

C Learning Curves

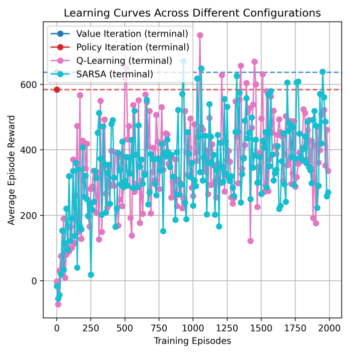
The learning curves for each experiments is visualized in Figure 6. Considering the amount of time it took to generate these plots for each case, it would be a shame to exclude them due to a lack of writing space. As such, for each 10th episode increment, the models are evaluated and performance metrics (cumulative rewards) are collected and analyzed. Please note that only a single point evaluation is necessary for the model-based methods .



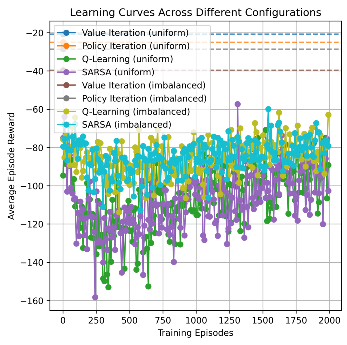
(a) Traffic: E1



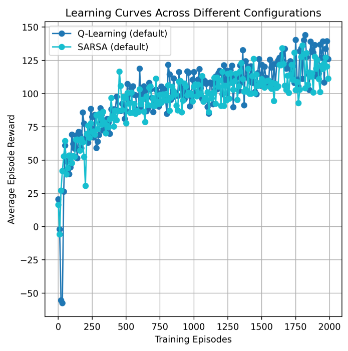
(b) Traffic: E2



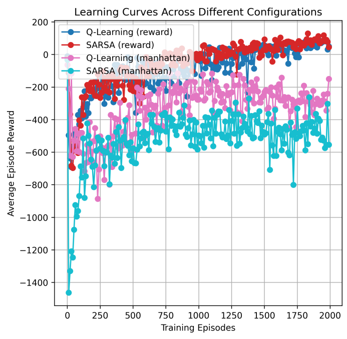
(c) Traffic: E3



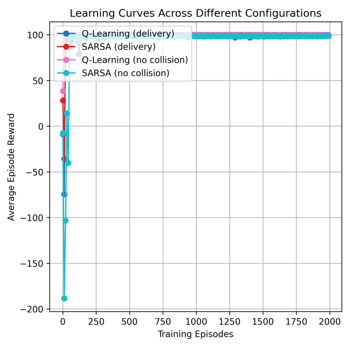
(d) Traffic: E4



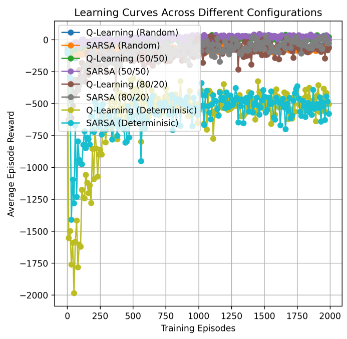
(e) Robot: E1



(f) Robot: E2



(g) Robot: E3



(h) Robot: E4

Figure 6: Traffic Intersection and Warehouse Robot Experiment Learning Curves