Networked Restless Multi-Arm Bandits with Reinforcement Learning

Hanmo Zhang[1], **Kai Wang**[1]

[1] Georgia Institute of Technology
hanmo@gatech.edu, kwang692@gatech.edu

Abstract

Restless Multi-Armed Bandits (RMABs) are a powerful framework for sequential decision-making, widely applied in public health challenges such as resource allocation and intervention optimization. However, traditional RMABs assume independence among arms, limiting their ability to account for interactions between individuals, which can be common and significant in a real-world environment. This paper introduces Networked RMAB, a novel framework that integrates the RMAB model with the independent cascade model to capture interactions between arms in networked environments. We establish the submodularity of Bellman's Equation for that model, enabling efficient policy design and proposing a Q-learning algorithm to account for the networked setting. Initial experimental results demonstrate that our network-aware approach outperforms a network-blind approach, highlighting the importance of capturing and leveraging network effects where they exist.

1 Introduction

Public health challenges such as infectious disease control, vaccination strategies, and the management of chronic illnesses increasingly require sophisticated sequential decision-making under uncertainty. These challenges involve allocating limited resources to interventions whose outcomes are uncertain and where timely decisions can significantly impact population health outcomes (World Health Organization 2019).

The Restless Multi-Armed Bandit (RMAB) framework has emerged as a powerful tool for addressing such sequential decision-making problems under resource constraints. Prior work has successfully applied variations of RMABs to various public health settings, including optimizing treatment strategies for infectious diseases (Mate et al. 2020) and designing treatment policies for tuberculosis patients in Mumbai, India (Mate, Perrault, and Tambe 2021).

However, a significant limitation of traditional RMAB models is the assumption of independence among arms. In many real-world applications, especially in public health, the state of one individual can directly affect others due to social interactions or network effects. For instance, the spread of infectious diseases is inherently a networked process where an individual's health status influences the infection risk of their contacts (Pastor-Satorras and Vespignani 2001). To model such interactions, the Independent Cascade (IC) model has been widely used to capture the probabilistic spread of influence through a network (Kempe, Kleinberg, and Tardos 2003).

In this paper, we introduce the *Networked Restless Multi-Armed Bandit (NRMAB)* framework, which integrates RMABs with the IC model to capture interactions between arms in a networked environment. By incorporating network effects, our model allows the action on one arm to influence not only its own state transitions but also those of neighboring arms through cascades. We formulate Bellman's Equation for this networked setting and prove its submodularity. This property enables us to design efficient greedy algorithms with performance guarantees close to the optimal policy (Nemhauser, Wolsey, and Fisher 1978).

Building on this theoretical foundation, we develop a Q-learning algorithm tailored for NRMABs. Our algorithm approximates the optimal policy without the need to compute the exact value function – which is computationally infeasible in large networks – while maintaining the submodularity of the value function to guide the learning process. We validate our approach through experiments on synthetic networks, demonstrating that our network-aware algorithm significantly outperforms network-blind baselines, including the traditional Whittle Index policy (Whittle 1988). These results highlight the importance of capturing network effects in sequential decision-making problems and suggest that NRMABs can provide more effective intervention strategies in public health and other domains where such networked interactions are significant.

2 Related Works

Restless multi-armed bandits: Restless Multi-Armed Bandits (RMABs), first introduced by Whittle (Whittle 1988), extend the classic Multi-Armed Bandit framework to scenarios where each arm evolves over time regardless of whether it is selected. This extension makes RMABs a powerful tool for modeling decision-making problems in uncertain and evolving environments. However, finding optimal policies for RMABs is PSPACE-hard (Papadimitriou and Tsitsiklis 1999), leading researchers to develop various

approximation algorithms and heuristics. The Whittle index policy (Whittle 1988) is a well-known heuristic that provides near-optimal solutions under certain conditions, particularly when the problem is indexable. RMABs have been applied in domains such as machine maintenance (Glazebrook, Mitchell, and Ansell 2005), healthcare (Mate et al. 2020), and communication systems (Liu and Zhao 2010).

Independent Cascade Model: The Independent Cascade model, introduced by Kempe et al. (Kempe, Kleinberg, and Tardos 2003), captures the probabilistic spread of influence through networks and has become a fundamental framework for studying diffusion processes in social networks. In the IC model, active nodes have a single chance to activate each inactive neighbor with a certain probability, modeling phenomena such as information spread, and epidemic propagation. The problem of influence maximization—selecting a set of initial nodes to maximize the expected spread—is NP-hard but benefits from the property of submodularity, which allows for efficient approximation algorithms with provable guarantees (Nemhauser, Wolsey, and Fisher 1978). Submodular function maximization has been extensively studied and applied to various network optimization problems (Leskovec et al. 2007; Chen, Wang, and Wang 2010).

Networked Bandits: In the domain of health monitoring and intervention, prior work has explored extending RMABs to account for network effects. For example, Ou et al. (2022) introduced a Networked Restless Multi-Armed Bandits framework for location-based by accounting for movement of people between locations. While this networked approach enhances resource allocation effectiveness in this scenario, it is particular to this narrow problem and difficult to generalize. In contrast, our work advances this foundation by modeling a wider range of interactions, enabling more nuanced and comprehensive resource allocation strategies in more complex and interconnected environments.

3 Problem Setting

3.1 Motivation

Efficient distribution of health resources is a critical challenge faced by public health authorities, particularly during outbreaks and pandemics. Traditional resource allocation strategies often rely on static models that assume independence among individuals, neglecting the complex interactions inherent in social and contact networks. For instance, during the COVID-19 pandemic, the effectiveness of interventions such as vaccination or quarantine depends not only on targeting specific individuals but also on how these interventions influence the broader network of interactions (World Health Organization 2020). Ignoring these network effects can lead to suboptimal allocation of limited resources, resulting in higher transmission rates and increased morbidity and mortality (Pastor-Satorras and Vespignani 2001).

To address this limitation, our work introduces the **Networked Restless Multi-Armed Bandit (NRMAB)** framework, which integrates the RMAB model with the Independent Cascade model to account for interactions between individuals in a networked environment. This integration allows for a more realistic representation of how interventions on one individual can propagate through the network, influencing the health states of others. The novelty of our approach lies in the formal incorporation of network effects into the RMAB framework and the demonstration of submodular properties in this combined model. These theoretical advancements facilitate the development of scalable algorithms capable of handling large and complex networks, which are common in public health applications. Additionally, our reinforcement learning-based solutions provide practical tools for decision-makers to dynamically adapt resource allocation strategies in response to evolving network dynamics.

3.2 Network RMAB Problem Formulation

An instance of the network RMAB problem is composed of a graph $G = (\mathcal{V}, \mathcal{E})$, where each node $v \in \mathcal{V}$ represents an arm that can transition between different states $s \in \mathcal{S}$. In this paper, we assume each node can only be in one of two states: inactive ($s = 0$) or active ($s = 1$). Each arm v can be targeted with an action $a \in \{0, 1\}$ corresponding to passive and active intervention, respectively. We assume each iteration we can only pick k arms to provide actions. This can be expressed as a budget constraint on the actions: $\sum_{i\in[n]} a_i \leq k$.

Independent arm transition: Given the state s and the action a of node v, the state transitions to the next state u based on the transition probability $P_v(s, a, u)$. We assume that active actions yield higher probabilities of beneficial transitions compared to passive actions. Specifically, for each node $v \in \mathcal{V}$:

$$P_v(s = 0, a = 1, u = 1) \geq P_v(s = 0, a = 0, u = 1), \quad (1)$$
$$P_v(s = 1, a = 1, u = 1) \geq P_v(s = 1, a = 0, u = 1). \quad (2)$$

In short, we can write the independent transition of the current state of all nodes $\boldsymbol{s} = [s_v]_{v\in\mathcal{V}}$ under action $\boldsymbol{a} = [a_v]_{v\in\mathcal{V}}$ to a temporary next state $\boldsymbol{u} = [u_v]_{v\in\mathcal{V}}$ by:

$$P(\boldsymbol{u}|\boldsymbol{s}, \boldsymbol{a}) = \prod\nolimits_{v\in\mathcal{V}} P_v(s_v, a_v, u_v) \quad (3)$$

Independent cascade: Nodes are interconnected through undirected edges $e \in \mathcal{E}$, where each edge has a weight $0 < w_e < 1$, representing the probability that an active node activates its neighbor via a cascade. This is motivated by the independent cascade model that disease or information can cascade from active nodes to their neighbors. We use a function P_G to denote the probability that the temporary state $\boldsymbol{u}$ cascades to the next state $\boldsymbol{s}' = [s'_v]_{v\in\mathcal{V}}$ through the graph G and the cascade probability of each edge by:

$$P_G(\boldsymbol{s}'|\boldsymbol{u}) \quad (4)$$

Reward objective: Our goal is to select the optimal k nodes at each timestep to maximize the cumulative reward over multiple timesteps t. This objective is formalized using the discounted return:

$$\sum\nolimits_{t=0}^{\infty} \gamma^t R(\boldsymbol{s_t}, \boldsymbol{a_t}), \quad (5)$$

where γ is the discount factor ($0 \leq \gamma < 1$) that prioritizes immediate rewards over distant future rewards. The reward function $R(s, a)$ is the immediate reward received after taking action a in state s, which is defined as:

$$R(\boldsymbol{s}, \boldsymbol{a}) = \sum\nolimits_{v \in \mathcal{V}} r(v), \tag{6}$$

where $\mathcal{V}$ represents the set of all nodes in the graph, and $r(v)$ is the value associated with node v if it is active ($s(v) = 1$), or zero otherwise.

4 Methodology

Our objective is to develop an algorithm capable of consistently selecting near-optimal actions in a scalable manner to maximize the accumulated value in each state. To represent action optimality, we use Bellman's Equation:

$$V(\boldsymbol{s}) = \max_{\boldsymbol{a} \in \mathcal{A}^n} Q(\boldsymbol{s}, \boldsymbol{a}), \tag{7}$$

$$Q(\boldsymbol{s}, \boldsymbol{a}) = R(\boldsymbol{s}, \boldsymbol{a}) + \gamma \sum\nolimits_{u} P(\boldsymbol{u} \mid \boldsymbol{s}, \boldsymbol{a}) \sum\nolimits_{s'} P_G(\boldsymbol{s'} \mid \boldsymbol{u}) V(\boldsymbol{s'}),$$

where u denotes the temporary state that each arm transitions independently based on their active and passive transition probabilities, and P_G further transitions from the temporary state u based on the independent cascade model. However, one challenge in the Bellman equation is the exponential number of actions to consider in Equation 7, which can be computationally challenging to optimize.

4.1 Submodularity of the Q-Function

In our algorithm, we first prove that $Q(\boldsymbol{s}, \boldsymbol{a})$ is submodular with respect to $\boldsymbol{a}$. Our problem aligns with the influence maximization problem, where a greedy hill-climbing strategy can achieve a performance guarantee of $1 - \frac{1}{e}$, approximately 63% of the optimal solution (Kempe, Kleinberg, and Tardos 2003). We proceed with a proof of the submodularity of $Q(\boldsymbol{s}, \boldsymbol{a})$.

Theorem 1. *Given a known $V(\boldsymbol{s})$ function and a constant state $\boldsymbol{s}$, we show that $Q(\boldsymbol{s}, \boldsymbol{a})$ is submodular with respect to a.*

Proof sketch. We show that for any $A \subseteq B \subseteq N$ and $t \notin B$:

$$Q(s, A \cup \{t\}) - Q(s, A) \geq Q(s, B \cup \{t\}) - Q(s, B)$$

In $Q(s, a)$, the reward function $R(s, a)$ is inherently submodular. We focus on the expected future value component:

$$\sigma(a) = \sum\nolimits_{s', u} P_G(s' \mid u) P(u \mid s, a) V(s')$$

We model the state transitions and cascades using coupled probabilistic simulations. For each node $v \in \mathcal{V}$, we simulate two coin flips: x_v, which represents the node's outcome in the transition step under a passive action, and y_v, the same node's outcome under an active action. We couple these coinflips such that if x results in activation, the corresponding y must also result in activation. For each edge $e \in \mathcal{E}$, we simulate a coin flip z_e to determine if an active node activates its neighbor via a cascade. With a full set of coinflips X, Y, Z, we deterministically know the set of active nodes after applying an action. Let $\sigma_{XY}(A)$ denote the set of active nodes after the Transition Step, and $\sigma_Z(\sigma_{XY}(A))$ denote the set of active nodes after both Transition and Cascade Steps.

We know that $\sigma_{XY}(A) \subseteq \sigma_{XY}(B)$ for $A \subseteq B$ and $\sigma_{XY}(A \cup \{t\}) \setminus \sigma_{XY}(A) = \sigma_{XY}(B \cup \{t\}) \setminus \sigma_{XY}(B)$. Using these properties, the submodularity inequality can be rewritten for the independent cascade as:

$$\begin{aligned} &\sigma_Z(\sigma_{XY}(A \cup \{t\})) - \sigma_Z(\sigma_{XY}(A)) \\ &\geq \sigma_Z(\sigma_{XY}(B \cup \{t\})) - \sigma_Z(\sigma_{XY}(B)). \end{aligned}$$

Since $V(s)$ depends directly on the total weighted node value and each active node contributes positively, $V(\sigma_{X,Y,Z}(A))$ is also submodular. Consequently, our expected future value can be formulated as:

$$\sigma(A) = \sum\nolimits_{X,Y,Z} P(XYZ) \cdot V(\sigma_{X,Y,Z}(A))$$

which is a non-negative linear combination of submodular functions, maintaining submodularity. Therefore, $Q(s, a)$, being a sum of submodular functions, is itself submodular. □

4.2 Deep Q-Learning with Hill Climbing

Deep Q Network To solve a NRMAB problem, we propose a Deep Q-Network using Hill Climbing. Our neural network takes in a representation of the state and action and pass it through three fully connected hidden layers to producing a Q-value for each state-action pair $Q(\boldsymbol{s}, \boldsymbol{a})$. However, if we have a large number of nodes and a large action size, then combinations of possible actions quickly becomes infeasible to calculate.

Incorporating Hill Climbing Thus, we iterate through the list of all possible single actions and utilize a neural network to predict the Q value of each single action a. Then, we greedily select the k actions with the highest Q-values. This should leverage the submodular properties of Bellman's Equation to achieve the $1 - \frac{1}{e}$ performance guarantee. We combine DQN and hill climbing to design a scalable Q-learning algorithm to solve the network RMAB problems, where the algorithm can be found in Algorithm 1.

Algorithm 1: Hill climbing DQN

Initialization: Neural network $Q(s, a; \theta)$, replay buffer
while until θ converges **do**
  Hill climbing action: intervention set $A = \emptyset$.
  while $|A| < k$ (budget for intervention) **do**
    Solve $v^* = \arg\max_{v \in V} Q(s, 1_{A \cup \{v\}})$
    Update $A \leftarrow A \cup \{v\}$
  end while
  Execute $a = 1_A$ and collect experience
  DQN Updates: Sample mini-batches from the replay buffer and run gradient descent to update θ.
end while
Output: Q network parameter θ

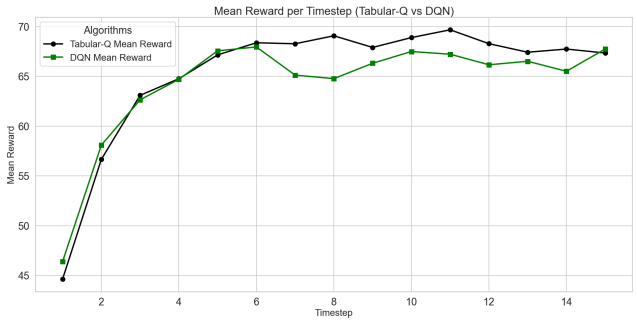

Figure 1: Mean Cumulative Reward Per Timestep for Tabular Q-Learning compared and DQN. Max timestep reward is around 70 and max cumulative reward is around 980.

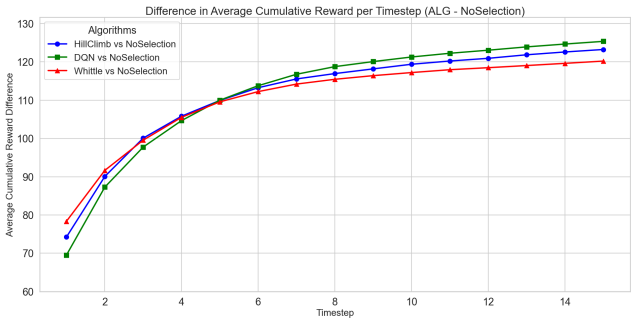

Figure 2: Reward difference between DQN, Hill Climbing, and Whittle Index versus no selection. Max timestep reward is around 430 and max cumulative reward is around 6100.

5 Experiments

5.1 Algorithms

To evaluate the effectiveness of our DQN algorithm, we compare it with three other algorithms:

1. **Tabular Q-Learning**: Solves the full Bellman's Equation for each state-action pair for small state-action sizes.
2. **Naive Hill-Climbing**: performs hill-climbing by calculating the value of activating a node in a state, taking into account network effect but ignoring future states.
3. **Whittle Index**: A traditional method for solving RMABs without considering network effects

We evaluate our algorithms on synthetic networks of varying sizes to assess their scalability and performance. The DQN is independently defined and trained using TianShou and PyTorch for the neural network, and Gymnasium for the simulation environment. The neural network is trained over three epochs of 1000 steps.

Simulation Protocol After training, each algorithm is evaluated through 100 simulations, each spanning 15 timesteps. The following metrics are collected:

1. **Mean Cumulative Reward**: The average total reward gathered at the end of all simulations
2. **Mean Reward Per Timestep**: The average reward gained at each timestep

For a baseline comparison, we also collect the above data on an algorithm that does not select any actions (no-select). We compare the difference in mean cumulative reward per timestep and mean reward per timestep of each algorithm against the no-select algorithm; this is calculated by dividing the cumulative reward at timestep t by t.

6 Results & Discussion

Overall, DQN performs comparably to the Hill Climbing algorithm, and both outperform the whittle index.

Submodularity Verification Figure 1, shows the performance of DQN compared to Tabular Q-Learning, an approach that gives near-optimal solutions at every timestep. We see the optimality guarantee of submodularity: DQN does not perform at less than a 63% threshold compared to Tabular Q-Learning.

DQN Algorithmic Effectiveness Figure 2 displays the difference in average cumulative reward between DQN, Hill Climbing, and Whittle Index compared to no selection on a graph with 300 nodes and 300 edges, with data collected over the same simulation parameters. Although the difference is small, we see a clear and consistent average outperformance of DQN against the other two methodologies.

6.1 Discussion

We observe several key results:

1. **Performance of Q-Learning Approaches**: In most cases, DQN exhibits better performance than Hill Climbing and Whittle Index, and does not perform notably worse than an optimal algorithm for small graph sizes. This similarity can be attributed to the submodular nature of the reward function, which allows both algorithms to achieve the $1 - \frac{1}{e}$ performance guarantee.
2. **Importance of Network Effects**: The Whittle Index, which does not account for network cascades, underperforms compared to network-aware approaches. This finding underscores the critical role of modeling network interactions in resource allocation problems. However, this performance difference decreases as the range of node values increases.
3. **Activation Dynamics**: There appears to be a saturation point in the number of active nodes, where the rate of activation balances with the deactivation rate. This suggests a theoretical maximum activation level achievable given the network structure and resource constraints.
4. **Ratio of Graph Sizes to Actions Taken** Performance differences between the algorithms decrease as the ratio between graph and action size increases. This is likely due to more passive transitions decreasing the impact of active actions and the existence of many near-optimal actions making them easier to find for all algorithms.

The superior performance of network-aware algorithms over the Whittle Index emphasizes the necessity of considering interdependencies in real-world applications.

References

Bellemare, M. G.; Dabney, W.; and Munos, R. 2017. A distributional perspective on reinforcement learning. *International Conference on Machine Learning*, 449–458.

Chen, W.; Wang, C.; and Wang, Y. 2010. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, 1029–1038. New York, NY, USA: Association for Computing Machinery. ISBN 9781450300551.

Glazebrook, K.; Mitchell, H.; and Ansell, P. 2005. Index policies for the maintenance of a collection of machines by a set of repairmen. *European Journal of Operational Research*, 165(1): 267–284.

Hasselt, H. v. 2010. Double Q-learning. *Advances in Neural Information Processing Systems*, 2613–2621.

Kempe, D.; Kleinberg, J.; and Tardos, E. 2003. Maximizing the Spread of Influence through a Social Network. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-03)*, 137–146. New York, NY, USA: ACM.

Leskovec, J.; Krause, A.; Guestrin, C.; Faloutsos, C.; Vanbriesen, J.; and Glance, N. 2007. Cost-effective outbreak detection in networks. volume 420-429, 420–429.

Liu, K.; and Zhao, Q. 2010. Indexability of Restless Bandit Problems and Optimality of Whittle Index for Dynamic Multichannel Access. *IEEE Transactions on Information Theory*, 56(11): 5547–5567.

Mate, A.; Killian, J. A.; Xu, H.; Perrault, A.; and Tambe, M. 2020. Collapsing Bandits and Their Application to Public Health Interventions. *Neural Information Processing Systems*.

Mate, A.; Perrault, A.; and Tambe, M. 2021. Risk-Aware Interventions in Public Health: Planning with Restless Multi-Armed Bandits. In *AAMAS '21*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533.

Nemhauser, G. L.; Wolsey, L. A.; and Fisher, M. L. 1978. An Analysis of Approximations for Maximizing Submodular Set Functions—I. *Mathematical Programming*, 14(1): 265–294.

Ou, H.-C.; Siebenbrunner, C.; Killian, J.; Brooks, M. B.; Kempe, D.; Vorobeychik, Y.; and Tambe, M. 2022. Networked Restless Multi-Armed Bandits for Mobile Interventions.

Papadimitriou, C. H.; and Tsitsiklis, J. N. 1999. The complexity of optimal queueing network control. *Mathematics of Operations Research*, 24(2): 293–305.

Pastor-Satorras, R.; and Vespignani, A. 2001. Epidemic Spreading in Scale-Free Networks. *Phys. Rev. Lett.*, 86: 3200–3203.

Sutton, R. S. 1999. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *Advances in Neural Information Processing Systems*, 1057–1063.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement Learning: An Introduction*. MIT press.

Whittle, P. 1988. Restless bandits: Activity allocation in a changing world. *Journal of Applied Probability*, 25: 287–298.

World Health Organization. 2019. *World Health Statistics 2019: Monitoring Health for the SDGs, Sustainable Development Goals*. Geneva, Switzerland: World Health Organization.

World Health Organization. 2020. *World Health Organization guidelines on COVID-19: Evidence synthesis and recommendations*. World Health Organization.

Appendix

A Full Proof of Submodularity

We demonstrate that our implementation of Bellman's Equation exhibits submodular properties, which are crucial for the efficiency and effectiveness of greedy algorithms. Submodularity ensures that the marginal gain of adding an element to a set decreases as the set grows, a property leveraged in influence maximization.

Formally, we aim to show that for any $A \subseteq B \subseteq N$ and $v \notin B$:

$$Q(s, A \cup \{v\}) - Q(s, A) \geq Q(s, B \cup \{v\}) - Q(s, B)$$

where $Q(s, A)$ represents the value of taking action set A in state s.

The reward function $R(s, a)$ is inherently submodular. We focus on the expected future value component:

$$\sigma(a) = \sum_{s',u} P_G(s' \mid u) P(u \mid s, a) V(s')$$

To analyze $\sigma(a)$, we model the state transitions using coupled probabilistic simulations. Specifically, for each node $n \in N$, we simulate two coin flips:

- x_n: Outcome under passive action with probability $P(s = 0, a = 0, s' = 1)$.
- y_n: Outcome under active action with probability $P(s = 0, a = 1, s' = 1)$.

We couple these coin flips such that if x_n results in activation ($s' = 1$), then y_n also results in activation. This coupling reflects the assumption that active actions have transition probabilities at least as good as passive actions, ensuring:

$$P(s = 0, a = 1, s' = 1) \geq P(s = 0, a = 0, s' = 1),$$
$$P(s = 1, a = 1, s' = 1) \geq P(s = 1, a = 0, s' = 1).$$

Additionally, for each edge $e \in E$, we simulate a coin flip z_e with bias $p_{v,w}$ to determine if an active node activates its neighbor via a cascade. A heads outcome denotes an active edge, leading to activation, while tails denote no activation.

Through these coupled simulations, we can deterministically determine the set of active nodes after applying an action. Let $\sigma_{XY}(A)$ denote the set of active nodes after the Transition Step, and $\sigma_Z(\sigma_{XY}(A))$ denote the set of active nodes after both Transition and Cascade Steps.

We establish the following properties:

1. $\sigma_{XY}(A) \subseteq \sigma_{XY}(B)$ for $A \subseteq B$. This is because adding more actions (from A to B) cannot decrease the set of active nodes due to the coupling of x_n and y_n.
2. $\sigma_{XY}(A \cup \{v\}) \setminus \sigma_{XY}(A) = \sigma_{XY}(B \cup \{v\}) \setminus \sigma_{XY}(B) = v'$, where v' represents the newly activated nodes resulting from adding action v. This holds because the additional action v affects nodes in the same manner regardless of the existing set A or B, thanks to the coupling ensuring $y_n \geq x_n$.

Using these properties, the submodularity inequality can be rewritten for the independent cascade as:

$$\sigma_Z(\sigma_{XY}(A \cup \{v\})) - \sigma_Z(\sigma_{XY}(A))$$
$$\geq \sigma_Z(\sigma_{XY}(B \cup \{v\})) - \sigma_Z(\sigma_{XY}(B)).$$

This inequality demonstrates that the number of active nodes after an action is submodular with respect to the size of the action set.

Since $V(s)$ depends directly on the total weighted node value and each active node contributes positively, $V(\sigma_{X,Y,Z}(A))$ is also submodular. Consequently, our expected future value can be formulated as:

$$\sigma(A) = \sum_{X,Y,Z} P(XYZ) \cdot V(\sigma_{X,Y,Z}(A))$$

which is a non-negative linear combination of submodular functions, maintaining submodularity. Therefore, $Q(s, a)$, being a sum of submodular functions, is itself submodular.

B Implementation Details for DQN

To handle the computational challenges posed by large graph sizes, we implement our algorithms using efficient data structures and optimization techniques. Specifically, we employ the following strategies during the calculation of Bellman's Equation:

- **Sampling Actions**: Instead of iterating through all possible actions, we sample a subset of actions to estimate the value function, reducing computational overhead. This approach is grounded in methods discussed by Sutton and Barto (Sutton and Barto 2018) and has been effectively utilized in large action space scenarios (Sutton 1999).

- **Monte Carlo Simulations**: To account for stochastic state transitions, we perform multiple simulations and average the results to approximate the expected future value. Monte Carlo methods provide a robust framework for estimating value functions in complex environments (Sutton and Barto 2018; Bellemare, Dabney, and Munos 2017).
- **Reinforcement Learning Framework**: For DQN, we utilize frameworks such as PyTorch to build and train neural networks that approximate the Q-values. This aligns with standard practices in deep reinforcement learning research (Mnih et al. 2015; Hasselt 2010).

These implementation choices enable our algorithms to scale effectively with larger networks while maintaining high performance.

State and Action Encoding The state of the graph is one-hot encoded into a tensor where active nodes are represented by 1 and inactive nodes are represented by 0. Actions are similarly encoded, with 1 being active action and 0 inactive. These tensors are concatenated and fed into the first layer of the neural network.

Network Setup We use a 5-layer fully connected Q-network $Q(s, a; \theta)$, each hidden layer having 128 nodes and ReLU activation function. We initialize an Adam optimizer for the Q-network parameters θ, and a replay buffer that stores $20,000$ transitions and training and testing collectors to gather experience. We use a custom Q policy that chooses k actions each step. For each action in k, with probability ϵ select a random action, otherwise select the action with the highest Q value. Every 50 steps, the policy is updated from mini-batches from the replay buffer using loss calculated and gradient descent. The network is trained for 3 epochs of 100 steps, with a batch size of 64 and γ of 0.9. The terminal state for training is defined by reaching a specific activation percentage.

C Experimental Configurations

For evaluating DQN against Tabular Q-Learning, we train both algorithms on a graph with 15 nodes, 30 edges, and randomly assigned integer values between 1 and 10. Each algorithm selects 2 nodes per timestep.

For evaluating DQN against Hill-Climbing and Whittle Index, we run experiments with two graph sizes: 300 nodes and 300 edges and 2500 nodes and 5000 edges. Nodes are generated with integer values between 1 and 2, and each algorithm selects 10 nodes per timestep.

For all graphs, nodes and edges are randomly generated. Additionally, each node is randomly assigned transition probabilities such that the active transitions are better than passive transitions. Edges between nodes are randomly generated, with a static cascade probability of 0.1 for each edge.

D Supplementary Results

Figure 3 displays the difference in rewards between the algorithms on a large graph with 2500 nodes. Note how the difference in average cumulative reward and reward per timestep are both less than that of the experiment with 300 nodes; this shows actions performed on a larger graph have a smaller impact.

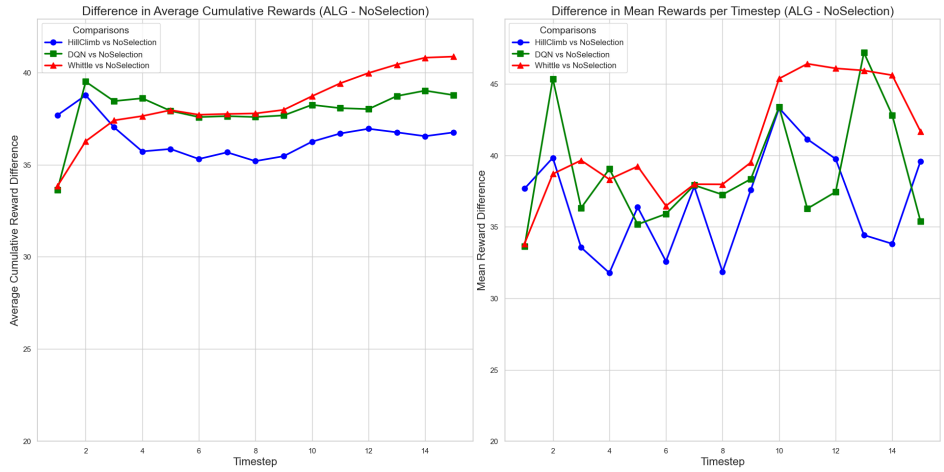

Figure 3: Difference in rewards between DQN, Hill Climbing, and Whittle Index compared to no selection for a large graph. Note how each algorithm is less differentiated compared to No Selection. Max timestep reward is around 2900 and max cumulative reward is around 41500.