國立中興大學資訊科學與工程學系

碩士學位論文

基於 SAC 深度強化式學習於 5G 網路高度動態流量條件下之資源允入控制最佳化

SAC-based Deep Reinforcement Learning Resource Admission Control for Highly Dynamic Traffic Condition in 5G Network

指導教授：陳 煥 博士 Huan Chen

研究生：胡勝勛 Sheng-Hsun Hu

中華民國 一百一十一年七月

摘要

為因應現今大眾對於網路方面的大量需求，擁有著更高頻寬、更低延遲的第五代行動網路 (5th Generation Mobile Networks, 5G) 已逐漸浮出檯面。但是比起前幾個世代，5G 的複雜網路環境需要乘載更大的網路流量，因此想要充分的利用 5G 網路，就必須要有一套方法可以對其進行準確的資源允入控制 (Resource Admission Control, RAC)，透過有效的分配網路資源，來使得網路效能最大化。近幾年來，有越來越多的相關研究指出深度強化式學習 (Deep Reinforcement Learning, DRL) 能夠應用於許多不同類型的複雜環境，本研究便是基於以上基礎，提出一種基於深度強化式學習方法 SAC (Soft Actor-Critic) 的離散化版本來針對擁有不同優先等級與流量的服務來進行允入控制，同時利用該方法的特點，也就是最大熵 (Max Entropy) 的方法來應付不斷變化的網路環境，並且利用優先經驗回放 (Prioritized Experience Replay) 來增加此方法所帶來的效益，對於神經網路的更新方式，本論文添加了一種學習條件，藉此來適度減少更新的次數，能夠有效的減少訓練時間並維持效能。而為了與過去其他在 RAC 問題上使用過的方法做效能上的評估，本篇研究將與傳統的貪婪式方法、固定切片 (Fixed Slice) 方法以及著名的深度 Q 網路 (Deep Q Network, DQN) 來進行比較，並將各種方法應用於各式不同流量的網路環境。最終的實驗結果顯示本篇提出的 Soft Actor-Critic-Resource Admission Control, SAC-RAC 方法在有四種不同等級的網路流量環境中擁有最好的結果，同時比起傳統著名的 DQN 高出約 25%的獎勵收益並降低約 84%的壅塞率。

關鍵字： 第 5 代行動通訊技術、深度強化式學習、SAC、優先經驗回放、資源允入控制

Abstract

In response to today's massive demand for networks, 5th Generation Mobile Networks (5G) with higher bandwidth and lower latency have gradually emerged. However, compared with previous generations, the complex network environment of 5G needs to carry larger network traffic. Therefore, if we want to make full use of the 5G network, we must have a set of methods that can accurately to do resource admission control (RAC), which can maximize network performance by effectively allocating network resources. In recent years, more and more related studies have pointed out that Deep Reinforcement Learning (DRL) can be applied to many different types of complex environments. Based on the above foundation, this paper proposes a deep reinforcement learning method. The discretized version of Soft Actor-Critic (SAC) is used for RAC for services with different priorities and traffic. At the same time, it uses the characteristics of this method which called maximum entropy to cope with the ever-changing network, and use Prioritized Experience Replay to increase the benefits of this method. For the update method of the neural network, this paper adds a learning condition to moderately reduce the number of updates, which can effectively reduce training time and maintain performance. To evaluate the performance of other methods used in the RAC problem in the past, this study will compare the traditional greedy method, the fixed slice method, and the famous Deep Q Network (DQN) to compare and apply various methods to a wide variety of network environments with different traffic flows. The final experimental results show that the Soft Actor-Critic-Resource Admission Control, SAC-RAC method proposed in this paper has the best results in the network traffic environment with four

different levels. At the same time, compared with the traditional famous DQN, the reward income is about 25% higher and the congestion rate is reduced by about 84%.

Keywords: 5th Generation Mobile Networks, **Deep Reinforcement Learning, Soft Actor-Critic, Prioritized Experience Replay, Resource Admission Control**

目錄

表目錄

圖目錄

第一章 緒論

1.1 研究背景

隨著次世代新型態網路已逐漸開始商業化，工業 4.0 的相關技術逐漸進步，物聯網 (Internet of Things, IOT) 也有了更多元的發展，近年來更是漸漸演進到了萬物聯網 (Internet of Everything, IOE) 的時代 [1]，萬物聯網系統將會面臨比起過往還要多上萬甚至上百萬的設備以及流量，用以串連全世界的各種應用。在面對 5G 工業 4.0 等高度複雜網路環境中如此龐大的數據傳輸量，RAC 更是扮演了著實重要的角色 [2]。

5G 為了盡可能的提升網路環境中的服務質量(Quality of Service, QoS)，在系統中定義了三種潛在應用，包括增強型移動寬帶 (enhanced mobile broadband, eMBB)、超可靠低延遲通訊 (ultrareliable and low latency communications, URLLC) 和大規模機器類型通訊 (massive machine type communications, mMTC) [3]，而為了負荷如此龐大的新型態網路，5G 系統採用了網路切片技術、網路功能虛擬化以及軟體定義網路來盡可能地進行網路資源分配，並且滿足良好的 QoS。在維護的過程當中，一旦沒有做好良好的控管，網路環境就會壅塞甚至斷線，如此一來將會使得使用者的整體網路使用體驗下降，反之，若是能夠有效的利用網路資源，則能夠進行大量的流量與請求的傳輸。因此，如何有效的對網路環境進行控管與維護一直以來都是很重要的研究議題。

1.2 研究動機與研究目的

文獻 [4] 的研究提出了一種新的搜尋演算法來滿足不同的頻寬需求，透過組合與整理服務類型來找出合適的切片配置，並且在保證 URLLC 服務的 QoS，同時將 eMBB 的阻塞概率降到最低。這種基於排隊和模型的方法可以獲得理論上的最優接入策略，但需要對網路系統和當前狀況有完善的了解。考慮到 5G 蜂窩系統在實現時的高度動態性和不確定性，極有可能無法獲得太完整的資訊，因此基於模型的 RAC 方案在實務上無法取得令人滿意得結果。

對此，有越來越多的研究指出在 5G 的網路環境當中，無論是使用網路切片來切割環境以分流服務[5][6]或是使用排程來規劃服務順序 [7] 等，強化式學習都能夠有不錯的表現。文獻 [8][9] 中嘗試使用深度 Q 網路 (Deep Q Network, DQN) 為 5G 網絡中的不同優先級的流量請求做資源分配，並且利用神經網絡來估計不同狀態下每個決策的預估長期收益，以便將來代理 (agent) 做出最佳決策。

此外，在穩定控管網路流量的同時，更應該要兼顧到壅塞次數，否則將會形成網路環境是順暢可使用，但卻有多數用戶無法進行連線的現象，這不是一個優良的網路環境會發生的情況。

因此，為了更進一步確認各方法在實際應用時的優劣，需要模擬在全日 24 小時之間的網路流量高峰潮與離峰朝，在面對不確定的網路流量隨機條件時，一個良好的方法必須也要能夠用最快的速度適應與負荷，同時盡可能的降低壅塞次數。如果能夠成功穩定的維持網路服務品質，避免造成壅塞或斷線，才算是一個適用於高度動態流量網路的方法。

本論文的目標就是設計出一個有效的方法，將所有流量請求進行分類，使較為重要的請求 (如醫療保健、國防安全等) 優先序提高，一般民生使用的請求 (如行動通訊、娛樂產業等) 優先序降低，使得在進行大量的流量與請求的傳輸同時保證

重要的流量不會被排除且阻擋在外，且能夠在短時間之內提供穩定且良好的網路服務品質，以及保持較低的壅塞與阻斷概率。

1.3 主要貢獻

本研究將提出透過深度強化學習中進行離散化後的 Soft Actor-Critic Discrete(SACD)，並且應用於高度動態流量網路環境中進行資源允入控制，故本研究的主要貢獻如下幾點:

(一) 本論文基於 SACD 方法，加入優先經驗回放，提出 SAC-RAC 方法並改變學習機制來增進效能。

(二) 本論文提出一個能夠因應不同流量網路的資源允入控制系統，針對流量來進行切片控制進而維持網路效能

(三) 本論文設計一個網路環境，該環境能夠模擬實際應用時所會面對的網路條件，用以評估本論文的方法更能有效的維持網路品質

1.4 論文架構

本論文共有五個篇章，在章節分配上，第一章主要闡明緒論、研究動機、研究目的以及主要貢獻等。第二章將對過去的文獻做一個回顧與探討，包含有關強化式學習的各類方法、5G 網路的架構以及資源允入控制相關的文獻。第三章將詳述本論文所使用主要方法，解釋本論文所提出的深度強化式學習應用於資源允入控制的問題，包含運作流程以及其優劣勢。第四章會進行實驗結果的分析，說明本次研究的實驗環境配置以及模擬方法並且針對本論文所提出的方法與其他對照的方法進行結果分析。最後第五章進行本論文總結並對未來研究可發展性進行探討。

第二章 理論背景

2.1 深度強化式學習

強化式學習 (Reinforcement Learning, RL) ，是一種基於利用代理 (agent) 不斷在環境 (environment) 中不斷嘗試並在環境所回饋的狀態 (state) 中尋求最大預期獎勵 (reward) 的一種機器學習方法，其基礎架構如圖 2-1 所示。這項做法啟發自心理學中的行為主義，即透過在環境給予獎勵或懲罰的機制之下，漸漸的構成對刺激的預期，進而得出獲得最大獎勵的習慣性行為，而在其中引入深度學習中的神經網路，用以面對較為複雜問題，此種做法即稱為深度強化式學習 (Deep Reinforcement Learning, DRL)。

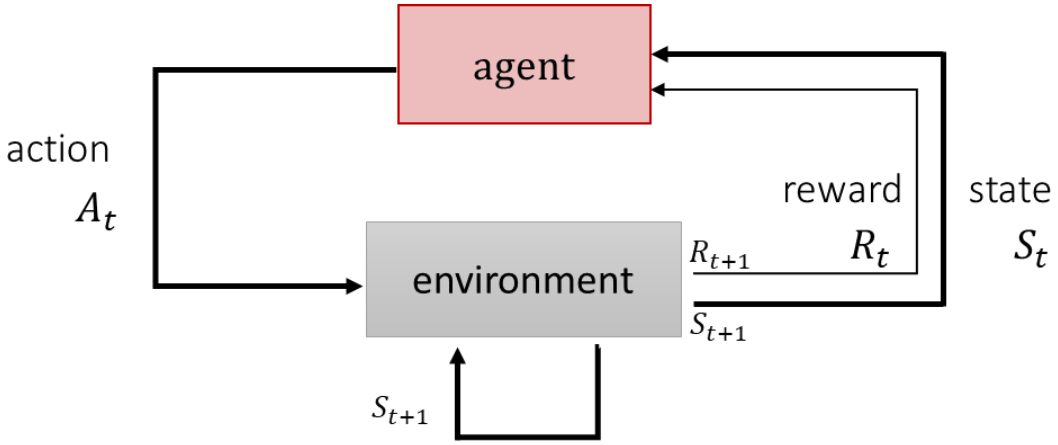


圖 2-1 強化式學習基礎架構圖

2.1.1 馬可夫決策過程

為了使得 agent 能夠順利與環境交換訊息，強化式學習以數學模型-馬可夫決策過程 (Markov Decision Process, MDP) 做為基礎理論 [10]，其中包含了以下兩大特性：

(一) 所有的決策過程皆為隨機性。

(二) 下一個將被轉移的狀態只與當前的狀態相關。

其中，第一個特性即代表無論在任何時刻 t，根據了當下的狀態 S_t 做出了決策 A_t 以後，下一個時間點的狀態 S_{t+1} 無法確定，因為未來是不可預測的。至此我們能稱此環境的轉移機率為馬可夫決策過程的 Dynamic，其以數學式子表示即為 $p(s', r|s, a) = \Pr(S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a)$。

第二個特性稱之為 Markov Property，意旨當前的狀態已經包含了過去所有的狀態，因此在計算時只需要考慮眼前的狀態，因此能夠大幅度減少計算量，同時以較不複雜的方法進行計算。

而在馬可夫決策過程可由五個元素所組成的位元組 (S, A, P, R, γ) 來描述，其中分別代表：

S：環境中所有可能發生的狀態集合。

A：任一狀態下能選擇的有限動作集合。

P：代表任一狀態做完行動後轉移至下一狀態的機率。

R：表示在執行完行動後用以評斷當前狀態下選擇的優劣回饋分數，而最終所追求的整體獎勵總合為 G_t ，由公式 (1) 定義。

γ：用以改變 agent 對獎勵的重視程度，其中 $0 \leq \gamma \leq 1$，當 γ 越趨近於 0 代表 agent 對於過去的獎勵越不在乎，只在乎當前獎勵，反之則將其納入考量。

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

因 G_t 屬於隨機變量，並沒有辦法對任一狀態的價值進行描述，因此將 G 取期望值即可使其轉為確定值並將其定義為價值函數 (Value Function)，並且用以當作在任一狀態下 agent 的行動優劣估計，如公式 (2) 所示。而還有一相似價值函數

的公式 (3)，其稱為動作價值函數 (Action Value Function)，當在任一狀態下執行動作後的期望回報。前者的價值函數用以評估某狀態的價值，後者的動作價值函數則是採取動作後的價值大小，兩者會因環境不同的影響，進而決定要選擇哪一函數來做為尋優的策略。

$$V(s) = E[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | s_t = s] \tag{2}$$

$$q(s,a) = E[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | s_t = s\,, a_t = a] \tag{3}$$

在馬可夫決策過程當中也加入了貝爾曼方程 (Bellman Equation) [10]，使得我們能夠在知道 S_{t+1} 時，就能夠計算出 S_t 的價值，同時也能表示 $V(S)$ 與 $q(s,a)$ 兩者間的遞迴關係，如此一來，便能透過下一個狀態的價值函數與獎勵來得到當前狀態的價值函數更新，定義如公式 (4)。

$$V(s) = E[R_{t+1} + \gamma V(s_{t+1}) | s_t = s] \tag{4}$$

總而言之，強化式學習的最終目的便是尋找一系列行為策略 π，並且能夠將其得到的獎勵最大化，表示對所有狀態 $s \in S$ 中的最大化價值函數，而其中最為優秀的將被定義為 π^* 來表示。而對於所有的π，都必須要滿足 $V^{\pi^*} \geq V^{\pi}$，其中 V^{π^*} 被定義於公式 (5)。因此想要得到最優的策略，只要能夠把當前的所有動作皆嘗試執行一次，並能夠獲得最大的獎勵 V^{π^*}，因此最優秀的 π^* 能夠被定義為公式 (6)。

$$V^{\pi^*}(s) = max_{\pi} V^{\pi}(s) \; = \; \max_{a} q_{\pi^*}(s,a) \tag{5}$$

$$\pi^*(s) = arg\max_{a} q_{\pi^*}(s,a) \tag{6}$$

2.1.2 深度 Q 學習(Deep Q Network)

強化式學習經由 agent 在環境底下，經過一系列的搜索行動後，再透過得到的

最優策略 π^* 引導來獲得最佳的獎勵回報，這種透過獎勵來進行更新的方法我們將其歸類為基於價值的方法。但是在最為傳統的強化 Q 學習 (Q Learning) 方式下會不斷的進行建立表格的動作，透過窮舉法的方式來一一走訪所有獎勵，最後找到最佳解 [11]，因此只適用於離散且較為簡易的環境與問題上。不過，現今許多研究議題的環境維度都偏高，非常不適合採用此種窮舉的方法來實現，因此傳統強化 Q 學習在大多數應用層面是無法使用的。為了解決此問題，Deepmind 在 2015 年提出了一個新的想法，那便是以神經網路來取代建立表格的作法，稱為深度 Q 學習 (Deep Q Network, DQN) [12]，如圖 2-2 所示，將環境 s 作為神經網路的輸入，而輸出是每個動作 a 所帶來的價值 (Q 值)，而神經網路的優化則是透過不斷與環境進行互動與交換訊息來達成。當神經網路透過環境輸入狀態 s 以後並給出動作 a 以後，環境便能相對應給出獎勵 R 以及下個狀態 s'，接著透過損失函數求導來更新策略 π 的模型參數 θ，藉此達到訓練的功用，如公式 (7) 所示。

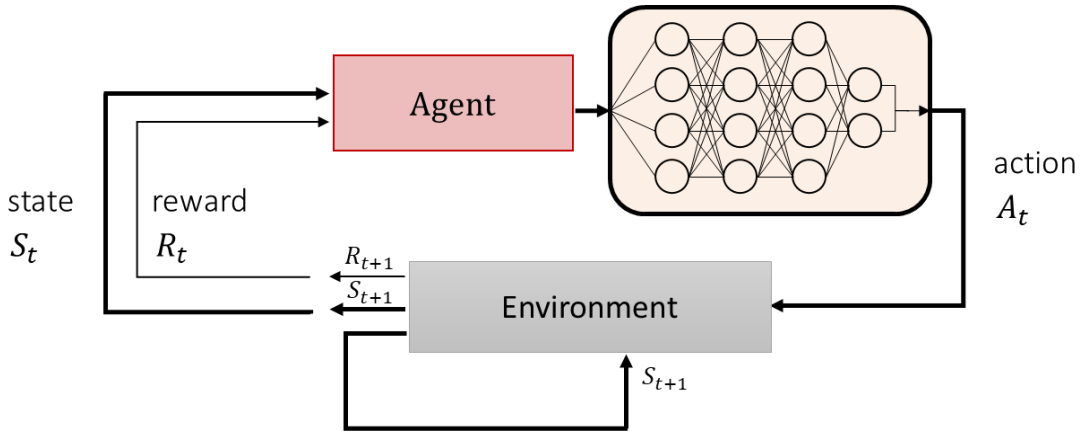


圖 2-2 深度強化學習簡易架構圖

$$L(\theta) = E[\left(TargetQ - Q(s,a)\right)^2] \tag{7}$$

公式 (7) 的損失函數是透過傳統的 Q 學習的更新公式以及目標 Q 值來建立的，兩者被定義在公式 (8) 與公式 (9)，兩者的目的皆是要盡可能地讓當前狀態下

的 Q 值能夠逼近目標 Q 值，進而對整體環境進行獎勵更新，而其最終的標準最優策略 π^*_{std} 都是依據獎勵 R 的最大值來決定，如公式 (10) 所示。

$$Q^*(s,a) = Q(s,a) + \alpha(R + \gamma max_{a'} Q(s',a') - Q(s,a)) \quad (8)$$

$$TargetQ = R + \gamma max_{a'} Q(s',a') \quad (9)$$

$$\pi^*_{std} = arg\, max_\pi \sum_t \mathbb{E}_{(st,at)\sim\rho\pi}[r(s_t,a_t)] \quad (10)$$

2.1.3 策略梯度(Policy Gradients)

上一小節所提到的基於價值的方法，可以透過不斷計算每個動作的價值以後，經過比對再選取反饋最大的做為動作，雖然這樣能夠確保每一次都能夠獲得好的結果，不過當場景轉換成維度較高或是連續空間，就會導致每一個動作的價值變得難以計算，因每一個微小的變化都會導致反饋的獎勵發生變動，因此會導致策略一直不斷的在計算與改變，如此一來便會有可能無法收斂的情形出現。

為了解決此問題，OpenAI 提出了一個新的方法稱之為策略梯度 (Policy Gradients)[33]，該方法將模型的輸出轉換為動作被選擇的機率，接著透過神經網路來找出好的策略梯度來進行更新，直到最後收斂至最佳的狀態。在 Policy Gradients 中每次訓練從初始環境到最後任務結束稱之為一個回合 (episode)，其中 agent 的神經網路藉由參數 θ 來構成策略 π，且在最後會生成訓練軌跡 τ，軌跡概率 $p_\theta(\tau)$ 以及回合最終獎勵 R，其定義於公式 (10)~(12)。

$$\tau = \{s_1, a_1, s_2, a_2, \cdots, s_T, a_T\} \quad (10)$$

$$p_\theta(\tau) = p(s_1) \prod_{t=1}^{T} p_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t) \quad (11)$$

$$R(\tau) = \sum_{t=1}^{T} r_t \quad (12)$$

agent 透過策略π所獲得的期望獎勵如公式 (13) 所示，其中的 R 即是軌跡 τ

在每個回合中最後所得到的獎勵總合，Policy Gradients 的目標便是透過公式 (14) 來改變梯度以及不斷改變參數，用以盡可能的提高期望獎勵。

$$\bar{R}_\theta = \sum_\tau R(\tau)\, p_\theta(\tau) = E_{\tau \sim p_\theta(\tau)}[R(\tau)] \tag{13}$$

$$\nabla \bar{R}_\theta = E_{\tau \sim p_\theta(\tau)}[R(\tau) \nabla log p_\theta(\tau)] \tag{14}$$

理論上只要透過這樣的方式便可以收斂到不錯的結果，但是在實務上是無法採樣到所有的軌跡的，僅能透過公式 (15) 的方式批量採取 N 筆資料，再對其進行計算後做為梯度，再接著使用公式 (16) 來對參數進行更新，其中 η 為學習率，計算完之後再以當下的梯度對下一批資料進行採樣，如此反覆執行。

$$\frac{1}{N}\sum_{n=1}^{N} R(\tau^n) \nabla log p_\theta(\tau^n) = \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n} R(\tau^n) \nabla log p_\theta(a_t^n | s_t^n) \tag{15}$$

$$\theta \leftarrow \theta + \eta \nabla \bar{R}_\theta \tag{16}$$

2.1.4 演員評論家(Actor-Critic)

在前面的小節中分別提到了基於價值的方法以及基於機率的方法，不過前者無法在高維與連續空間中應用，後者也因梯度的原因容易誤入區域最佳解，為了解決前述的兩個問題，而後產生了一個全新的方法為演員評論家 (Actor Critic, AC)[13]，他同時採用了 Q Learning 與 Policy Gradients 各自的優點並捨棄缺點，將兩種結合後透過評價機制來改善高變異數的問題，分別產出了一個由 Policy Gradients 演化而來並且只輸出動作的演員策略模型，另一則是由 Q Learning 衍生出只輸出 Q 值的評論家策略模型，且同時將回合更新的制度修改為單步更新，藉此改善學習模型容易陷入局部最佳解的問題，整體架構如圖 2-3 所示。從架構圖中可得知，AC 產生了兩組不同的類神經網路，且環境所反饋的狀態與獎勵將會分別影響到兩個網路，接著透過時間差分 (Temporal Difference, TD) 的公式 (17) 來計

算誤差並更新評論家的模型。公式中 γ 做為折扣係數，用於評斷演員所執行的動作優劣，不過 $V(s_{t+1})$ 是一個估計值，因此 TD 會產生一些偏差估計，稱之為 TD Error，雖然每一步的計算會有偏差估計，但與前面以回合更新的方法比較，TD Error 的變異數會較小，而演員的模型則會透過目標函數來進行更新，進而朝向更有可能獲取最大價值的方向訓練。

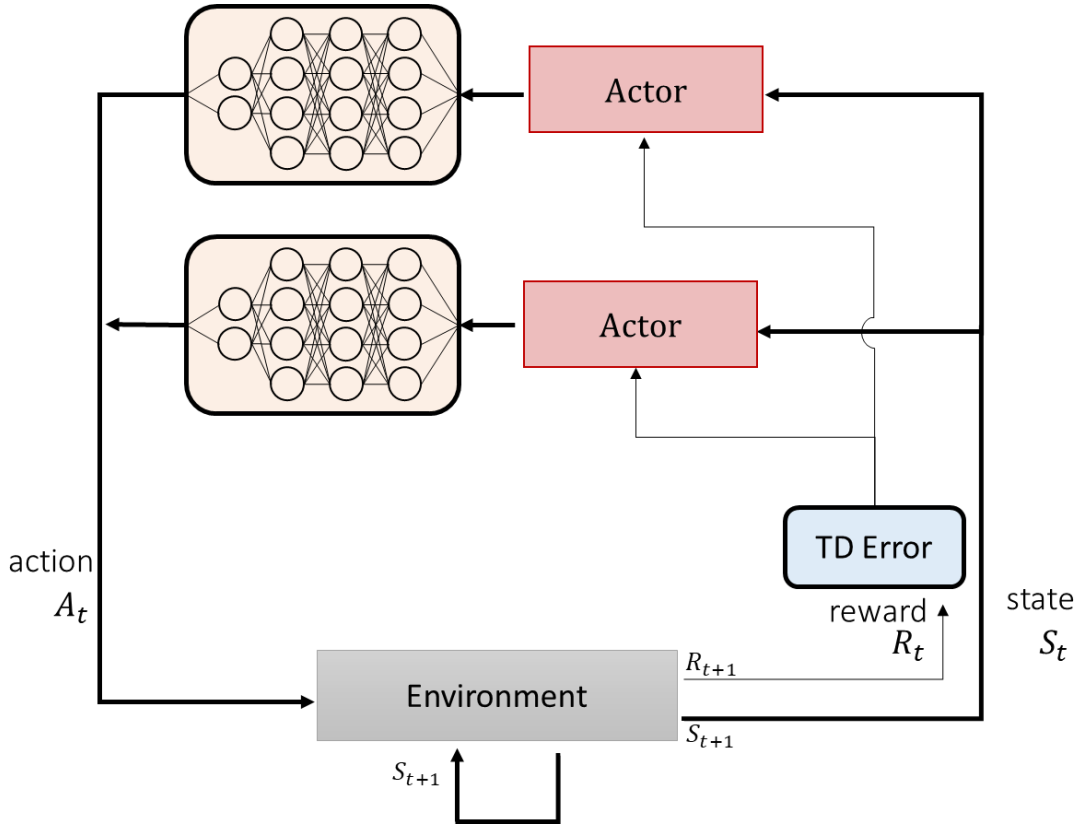


圖 2-3 演員評論家簡易架構圖

$$\delta = R + \gamma V(s_{t+1}) - V(s_t) \tag{17}$$

2.1.5 雙延遲深度確定性策略梯度(Twin Delayed Deep Deterministic Policy Gradients)

在連續空間的議題上，學者們為了避免現行的方法過分估計評論家模型的 Q

值導致成效不彰，在 2018 年便提出了新的方法來解決高度估計問題，稱之為雙延遲深度確定性策略梯度 (Twin Delayed Deep Deterministic Policy Gradients; TD3)[14]，此方法針對 AC 修改了三個部分，首先第一部分是受到雙 Q 學習 (Double Q Learning)[15] 所啟發，將評論家的模型增加為兩個，並且在後續運作時選擇倆倆模型中的最小值來盡可能減少偏差，如此一來演員模型會去選擇較低的 Q 值，避免高估的值被累計，同時增加被低估值被選擇到的機率。第二部分是降低了更新演員模型的頻率，原因是三個模型容易會對彼此造成影響，如果採用到了較為差的策略 (被高估的策略) 會導致結果不準確，因此降低更新頻率以後能夠保持模型穩定學習並且減少偏差。第三部分是為了減少過擬合所造成的高變異數目標值所進行的噪聲正歸化，由於相似的動作會有相似的值出現，因此透過平滑一系列的目標策略來對目標策略增加一些少量的隨機噪音變量來穩定其學習的過程，不過需要對添加的噪聲範圍進行一定程度的限制，以避免噪聲量過大導致策略會偏離原始動作。

2.1.6 近似端策略優化(Proximal Policy Optimization)

由於 Policy Gradients 想要學習到好的策略需要找到一個合適的學習率，但是要尋找到合適的參數配置難度是非常困難的，且如果在訓練的過程中每一次的策略變化太大就有可能使得模型表現不好，因此 OpenAI 在 2017 年推出了近似端策略優化 (Proximal Policy Optimization, PPO)[16]，如公式 (18)，主要核心概念在目標函式 $J(\theta)$ 中直接新增了名為 KL 散度 (Kullback-Leibler divergence, KLD) 的新約束項目，藉此來有效限制每次迭代中可以變更的策略範疇，用以衡量個策略間的差異性，使其盡可能的相似。PPO 同時也讓新的目標函式能夠實現在多個訓練步驟當中去使用小的 batch_size 來進行更新，目的是想解決學習率難以定義的狀況。公式 (19) 中的 $D_{KL}(\theta,\theta')$ 所代表的是指模型所輸出的兩個動作機率分布之間的

距離，而並非是參數的距離，因為每個參數之間任何一點微小的變化都於當下的動作都有著不一定的影響，因此可以透過 β 來控制 D_{KL} 的權重，盡可能減少估計時所產生的變異數。

$$J_{ppo}^{\theta'}(\theta) = J^{\theta'}(\theta) - \beta D_{KL}(\theta||\theta') \tag{18}$$

$$D_{KL}(\theta||\theta') = E_{x\sim P}(log\frac{\theta(x)}{\theta'(x)}) \tag{19}$$

2.1.7 Soft Actor-Critic

SAC 是由 Haarnoja T 於 2018 年所提出的演算法[28]，其特點在於嘗試將熵 (Entropy) 正則化，並且透過尋找最大熵 (Max Entropy) 的方式來使得演算法能夠在更多的解空間當中進行探索，以利在新的環境中學習新的任務。其中最大熵的特性如[29]所示，最優的策略不僅僅只考慮最終總獎勵值，而是在每一個狀態的當下所產生的動作的熵要盡量的提高，這個舉動會使得整體決策的隨機性增加，而 SAC 的最優決策公式如 (20) 所示。

$$\pi_{MaxEnt}^{*} = arg\,max_{\pi}\sum_{t}\mathbb{E}_{(st,at)\sim\rho\pi}[r(s_t,a_t) + \alpha H\big(\pi(\cdot\,|S_t)\big)] \tag{20}$$

上述公式中的 $\alpha H\big(\pi(\cdot\,|S_t)\big)$ 即是熵的計算方式，由公式 (21) 得出，意旨決策 π 在狀態 S_t 下的熵值，而 α 稱之為溫度參數，會在學期過程當中自動進行調整，熵的用途是來衡量當下隨機變量的隨機性，當熵的值越大，代表策略 π 就越有機會選擇不一樣的動作，使得策略更加有多樣性。

$$H\big(\pi(\cdot\,|s_t)\big) = E_{a\sim\pi(\cdot|S_t)}[-log\pi(a|s)] \tag{21}$$

而為了盡可能最大化目標，作者使用了在最大熵的框架內交替進行策略評估以及策略更新的方法 soft policy iteration。對此，作者將 soft V-function 定義為公式 (22) 。

$$V(S_t) \coloneqq E_{a\sim\pi}[Q(s_t,a_t) - \alpha\log\left(\pi(a_t|s_t)\right)] \tag{22}$$

接著從隨機初始化的函數 $Q\colon S\times A\rightarrow R$ 開始取得 soft Q function，並且重複

利用以下修改過後的貝爾曼方程式 (23) ，其中 $P:S\times A\to S$ 給定當前的狀態與動作，得出下一狀態的分布。

$$T^{\pi}Q(s_t,a_t)\coloneqq r(s_t,a_t)+\gamma E_{s_{t+1}\sim p(s_t,a_t)}[V(S_{t+1})] \tag{23}$$

在連續空間中，作者先使用了帶有參數θ的神經網路來對 soft Q function 中的 $Q_\theta(s_t,a_t)$ 進行參數化 (24)，進而訓練 soft Q function 來使其盡可能最小化 Soft Bellman residual，其中 D 為過去訓練經驗的回放緩存區， $V_{\bar{\theta}}(s_{t+1})$ 則是使用 Q 的目標網路以及從回放緩存區所採樣後經由公式 (22) 進行蒙特卡羅方法來估計出來的。

$$J_Q(\theta)=E_{(s_t,a_t)\sim D}[\frac{1}{2}(Q_\theta((s_t,a_t)-(r(s_t,a_t)+\gamma E_{s_{t+1}\sim p(s_t,a_t)}[V_{\bar{\theta}}(s_{t+1})]))^2] \tag{24}$$

接著，為了使策略更新後能夠朝著獎勵最大化的方向發展，作者將更新策略中的 soft Q function 改為指數形式，同時也根據 KL 散度將該指數形式投影成可接受的策略空間，使策略被限制在參數化後的分布中，使其更容易被處理，因此整體策略改進步驟如公式 (25) 所示，其中劃分函數 $Z^{\pi_{old}}(S_t)$ 是不太好處理的，但是他並不會影響到新策略的梯度，故可以忽略。

$$\pi_{new}=\underset{\pi\in\Pi}{argmin}\,D_{KL}(\pi(\cdot\,|S_t)\|\frac{\exp(\frac{1}{\alpha}Q^{\pi_{old}}(s_t,\cdot))}{Z^{\pi_{old}}(s_t)}) \tag{25}$$

在連續狀態中，使用具有平均值與共異變數的參數 ϕ 來對策略 $\pi_\phi(a_t|s_t)$ 進行參數化，將其定義成在連續空間中較常使用的高斯策略。接著將公式 (25) 乘上溫度參數 α 並忽略劃分函數，進而學習策略參數 (26)。

$$J_\pi(\phi)=E_{s_t\sim D}[E_{a_t\sim\pi\phi}[\alpha\log(\pi_\phi((a_t|s_t))-Q_\theta(s_t,a_t)]] \tag{26}$$

作者也提到，這樣的做法代表需要對策略的輸出分布求期望值，這樣會導致誤差沒有辦法進行正常的反向傳播，為了解決這個問題，作者採用了重參數化的方法 (reparameterization trick) 將輸出與噪聲向量做結合，而非直接採用原本輸出的隨機

分布，為了達成此目的，作者對動作輸出部分重新做了定義 (27)，而新的策略目標則修改為公式 (28)，其中 $\varepsilon_t \sim N(0, I)$。

$$a_t = f_\phi(\in_t; s_t) \tag{27}$$

$$J_\pi(\phi) = E_{s_t \sim D, \in_t; s_t}[\alpha log\ (\pi_\phi(f_\phi(\in_t; s_t)|s_t)) - Q_\theta(s_t, f_\phi(\in_t; s_t))] \tag{28}$$

此外，作者於隔年基於以上架構，額外又提供了一種學習溫度參數的方法，該方法可以提供溫度目標值的長期效益推導，使得我們並不需要將其設置為超參數，如公式 (29) 所示，其中 $\bar{H}$ 是一個與目標熵值超參數等價的常數向量，因公式包含了期望值，故無法對其求最小值，因此只能從回放經驗池中採樣後，再對其進行最小化。

$$J(\alpha) = E_{at \sim \pi t}[-\alpha(log\pi_t(a_t|s_t) + \bar{H})] \tag{29}$$

基於以上公式即可得到具有隨機策略的 SAC 深度強化式學習方法，其演算法架構如演算法 1 所示。此外由圖 2-4 可以更加瞭解隨機策略的重要性，該圖以一個簡易的迷宮遊戲為範例，假設星星為終點，而圓圈為 Agent 的初始位置。在圖 2-4 左側為傳統之強化式學習方法，在訓練過程中有可能會有兩種策略，其一為往下移動來接近終點，因其路線較近，因此為下方圖表的高峰；而另一策略則為向右進行移動，因路線較遠，因此為下方圖表之低峰。而 Q 函數如下方斜線分布，在傳統的方法下都是在眾多策略中尋找最大 Q 值的分布中心，在學習的過程當中會盡可能的在最高峰附近進行探索，進而學習應該向下走能夠獲得最高的獎勵，而直接放棄了低峰的次要路線。因此若是已經學習好的向下路線突然的被阻擋，使得原本學習的路線消失，如此一來便會導致原本學習到的策略失去作用，而為了重新抵達終點，則比須從頭開始學習新的路線，將探索範圍轉移至新的高峰附近。而要解決這個問題，最直接的方式便是確保在學習的過程當中，agent 能夠盡可能的探索所有有可能的狀態，在尋找高峰的同時也要考慮到其他次要且有前途的策略。在圖 2-4

右側，根據指數的 Q 值定義策略，利用 Energy-based Policy 的定義 [29]，即可以使得策略直接符合 Q 的分布，使其具有波茲曼分布的特性，讓每一個動作都被賦予一個特定且符合 Q 值的分布，讓所有動作被選取的機率皆不為 0，大大擴展了搜索的範圍，且滿足隨機策略的需求，讓 Agent 能夠在學習向下走的路線同時，也能夠學習到向右走的路徑，如此一來，即使環境不斷的變化，Agent 也能夠迅速的尋找到下一個替代策略，而非從頭開始學習。這也同時意味著這樣的作法能夠擁有非常強大的探索能力，在複雜的環境中盡可能的探索解空間，讓學習完成的 Agent 能夠提升本身的泛化性，更能夠應付各種複雜的環境條件，也能在面對各種干擾時能夠迅速的適應環境。此種特性在面對 5G 如此高度動態流量的網路環境時可說是非常有利，該策略能夠在不斷變動的網路環境底下快速的找到合適的策略，進而維持整體網路完靜的 QoS。而文獻 [30] 將 SAC 應用於各種不同的環境中進行測試，其結果如圖 2-5 所示，在與其他傳統深度強化式學習相比，SAC 能夠擁有更高的性能，因此本研究將採用 SAC 的演算法來解決 5G 高度動態流量網路環境的複雜問題。

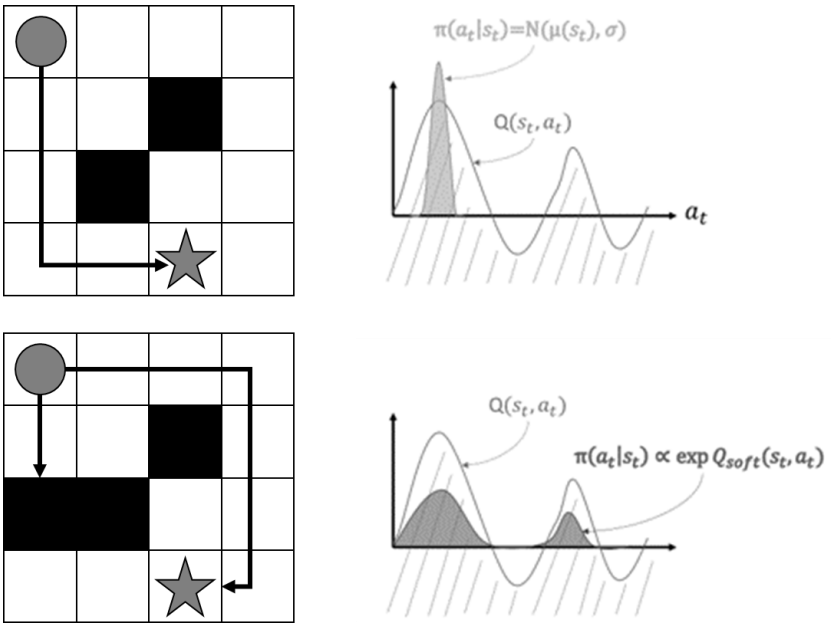


圖 2-4 隨機策略的重要性

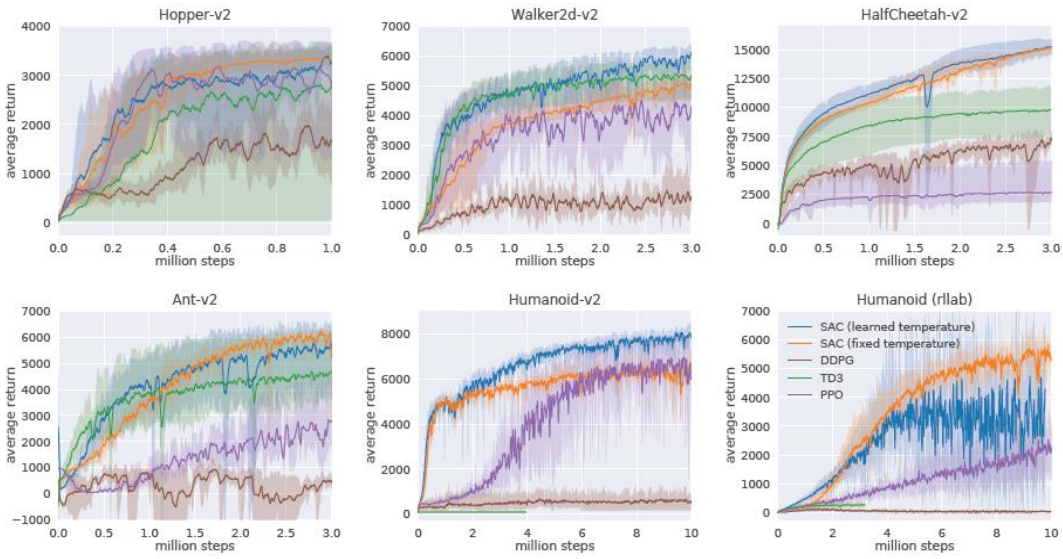


圖 2-5 SAC 與其他深度強化式學習之比較[30]

演算法 1 Soft Actor-Critic 演員評論家		
1.	**Input :** θ_1, θ_2, ϕ	➢ Initial parameters
2.	$\bar{\theta}_1 \leftarrow \theta_1, \bar{\theta}_2 \leftarrow \theta_2$	➢ Initialize target network weights
3.	$D \leftarrow \emptyset$	➢ Initialize an empty replay pool
4.	**for** each iteration **do**	
5.	**for** each environment step **do**	
6.	$a_t \sim \pi_\phi(a_t \mid s_t)$	➢ Sample action from the policy
7	$s_{t+1} \sim \mathrm{p}(s_{t+1} \mid s_t, a_t)$	➢ Sample transition from the environment
8.	$D \leftarrow D \cup \{(s_t, a_t, r(s_t, a_t), s_{t+1})\}$	➢ Store the transition in the replay pool
9.	**end for**	
10.	**for** each gradient step **do**	
11.	$\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in \{1,2\}$	➢ Update the Q function parameters
12.	$\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$	➢ Update policy weights
	$\alpha \leftarrow \alpha - \lambda \hat{\nabla}_\alpha J(\alpha)$	➢ Adjust temperature
13.	$\bar{\theta}_i \leftarrow \tau\theta_i + (1-\tau)\ \bar{\theta}_i$ for $i \in \{1,2\}$	➢ Update target network weights
14.		
15.	**end for**	
16.	**end for**	
17.	**Output :** θ_1, θ_2, ϕ	➢ Optimized parameters

2.2 5G 網路架構

5G 網路架構由一個名為第三代合作夥伴計畫 (3rd Generation Partnership Project, 3GPP) 的機構所發布 [34]，該機構目前由日本、北美洲、中國、歐洲、印度與韓國所組成，自 2G 網路架構開始，關於部屬和管理標準制定等等都由該組織一手包辦，時至今日 5G 網路定義也已經成形，主要有以下幾項技術來獲得比 4G 更廣的頻寬、更高的速度以及更低的延遲與應用：

(一) 軟體定義網路 (SDN)：以分層式架構來分層管理與分配各式網路功能。

(二) 網路功能虛擬化 (NFV)：將傳統硬體設備以軟體形式呈現，如此便能大幅增加配置的彈性、部屬的效率以及減低成本開銷。

(三) 多進多出 (MIMO)：隨著硬體技術進步，現今已慢慢出現毫米及尺寸的天線，如此一來便能在設備中放入更多的天線，使傳輸速度更上層樓。

(四) 波束成型 (Beamforming)：透過天線陣列 [17] 來調整訊號傳輸方向，使得原本浪費的訊號得以集中往需要的位置傳輸。

(五) 網路切片 (Network Slice)：為 5G 網路與本篇論文中最大的重點，將在以下做更詳細的說明。

2.2.1 網路切片

5G 網路中最核心且重要的技術就是網路切片，是做為網路資源管理與分配的主要推動者 [18]，其示意如圖 2-6 所示，就如同字面意思，是將從各個設備、基地台以及核心網路 (Core Network, CN) 中的網路切割成多個固定或是動態分配的小切片，這些切片是相互獨立的虛擬端到端網路，就像是將電腦的硬碟切割成不同的 C、D 碟一樣。如此一來在面對各式各樣的多使用者情境時，能夠有效的提升核心網路和無線存取網路 (Radio Access Network) 的靈活性，能夠更容易的掌控時間延

遲 (Latency)、頻寬 (Bandwidth)、安全性 (Security) 等各個面向，隨時分配並調整有限的網路資源來盡可能提升 QoS [19]-[20]。

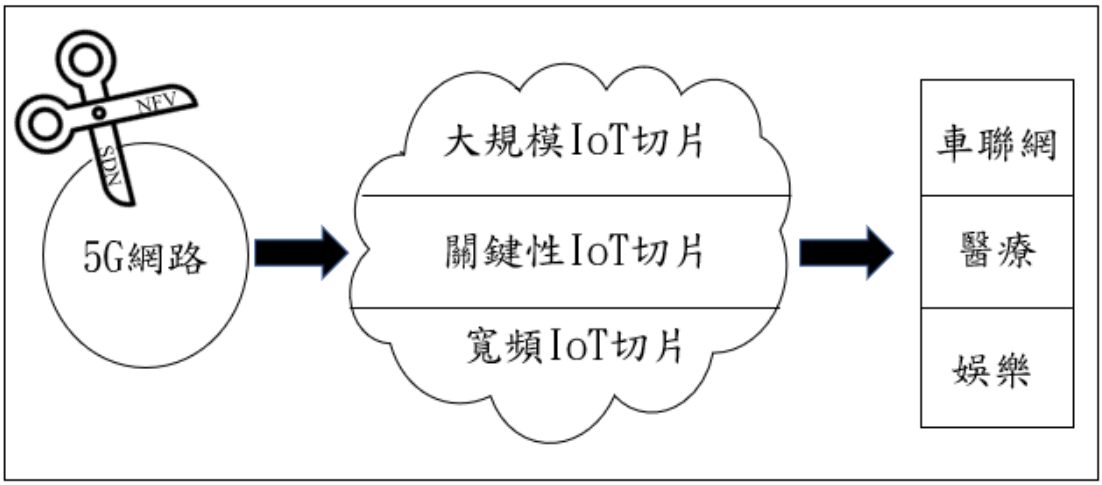


圖 2-6 網路切片示意圖

2.3 資源允入控制

在一般常見的區域網路中，單位時間內所能承受的數據請求量會依照範圍內的基地台數量而有所變動，為了能夠在不改變基地台數量的前提下維持良好的 QoS，就必須要對區域網路進行資源允入控制 (Resource Admission Control, RAC)。

RAC 就像是守門員一樣，根據目前環境的情況，可以對即將來的服務做出不同決策，如：接受、使其等待或是放棄。其中有些模型是可以接受服務等待的，有些則是會選擇直接放棄。對此在文獻 [21]-[23] 中已經整理出了諸多不同類型的 RAC 方法，無論是接受服務等待，或是無法接受則直接放棄，都圍繞著同一個核心概念，也就是給予不同的優先級，當對即將到來的服務進行分級，便可以有效的對其進行管理與控制，能夠有效的保證重要的服務能夠穩定的進行連接。

然而 RAC 是一個非常複雜的問題，因此使用神經網路來計算是一個非常有前

瞻性的方法，但要時刻穩定控制區域網路內的服務數量，同時又必須對即將抵達的服務進行允入控制，這樣離散且高維度的複雜問題，對於傳統的統計式機器學習來說負荷量實在過大，因此並不適用。為了解決此問題，越來越多研究將 RAC 建模為半馬可夫決策模型 (Semi-Markov Decision Process, SMDP)[24]-[27]，不過傳統的 SMDP 是使用動態規劃，在面對如此龐大的網路環境是不可行的，因此學者們便開始將此問題轉移至 RL 來解決 [25]-[27]。

2.4 排隊理論

為了能夠準確地模擬真實網路使用情形，就必須要先能夠合理的模擬網路使用請求的實際情況。1909 年，一位在丹麥電信公司的工程師厄朗發明了排隊理論[35]來建構通訊系統，目前此架構也已經各式的工業工程、商場甚至醫院[36]-[37]等地方廣泛使用，而後於 1953 年由後續學者提出了所謂的 A/B/C 表示法[38]，該表示法中包含了 A/B/C/X/Y 等 5 個元素，其中：

A：代表輸入規則，其常用的分配標準為。

M：輸入間距獨立且遵循卜瓦松分布或指數分布。

D：輸入間距獨立且為常數。

Ek：輸入間距獨立且同一分配的愛爾朗分配參數為 k。

GI：輸入間距符合一般獨立分配 (General Independent)。

B：代表服務時間應服從何種規律，通常情況與 A 相同。

C：代表模型中有多少平行隊列。

X：代表隊列最大容量限制。

Y：代表服務規則。

一般將系統輸入的項目稱為顧客，而在輸入規則與服務時間中顧客可以依照不同的機率分布來抵達與執行，一般可分為指數分布、卜瓦松分布、幾何分布、定

長分布、愛爾朗分布等；服務規則有先到先服務 (First Come First Service, FCFS)、後到先服務 (Last Come First Service, LCFS)、優先權服務 (Priority, PR)、隨機選擇服務 (Random Selection for Service, RSS) 等。

因此，若假設有一等候模型為 M/M/2/∞/FCFS，即代表輸入規則 A 遵照指數分配，服務時間 B 遵照指數分配、平行隊列數為 2、隊列最大容量無限制，且採用 FCFS 的規則。此外，當 X 為∞且 Y 為 FCFS 時，在表示上會將此兩項省略，因此該等候模型可簡稱為 M/M/2。

2.4.1 卜瓦松分布(Poisson Distribution)

卜瓦松分布是在排隊理論當中最常被用來使用的一種分布，其原因將於下方解釋。

在二項分配中，假設實驗為 n 次每次成功機率為 p，失敗機率為 q，且設 X 為成功的次數，則二項分布之機率分配函數、期望值、變異數如公式 (30)~(32) 所示。

$$P(X = x) = C_x^n p^x q^{n-x} \tag{30}$$

$$E(x) = np \tag{31}$$

$$Var(x) = npq \tag{32}$$

而卜瓦松分布為二項分配的一種極端現象，可看做 $n \to \infty$，$p \to 0$ 之二項分配，其期望值與變異數皆為 λ。

假設在單位時間 T 當中發生事件的次數為 λ，卜瓦松分配具有以下性質：

1. 一段時間中發生事件的次數與另一段時間發生的次數獨立。
2. 一段時間中發生事件的平均次數與時間長短成比例。
3. 在極短的時間內發生事件的機率趨近於 0。

因卜瓦松分布具有以上特性，能夠描述在單位時間內隨機事件發生的次數的

機率分布，用來當作顧客的輸入依據是較為公正且符合實際情況的，對此幾乎所有有關 RAC 的相關研究皆是以排隊理論為基礎來往下進行研究[40]-[41]。而圖 2-7 為當 λ 從 1 調整至 10 的分布情形，在模擬網路傳輸的環境時，可以用於模擬各式不同的流量輸入。

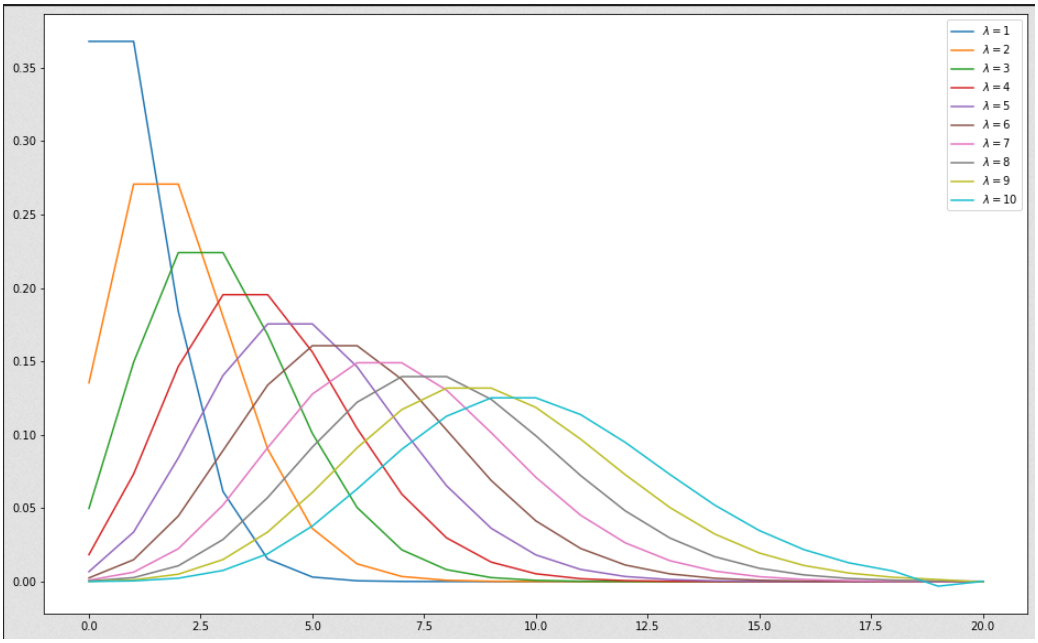


圖 2-7 卜瓦松分布理論曲線圖

本論文中所使用的網路模擬環境，將依照 2.4 節中所提到的排隊理論建立一組為 M/M/C 的網路服務架構，其中 C 將視為可提供服務的網路切片數量，並於不同的實驗環境當中進行調整，以盡可能最大化網路效能。

2.5 小結

在 5G 高度動態流量網路環境的資源允入控制問題上，除了基礎的貪婪做法與固定切片方法以外，使用強化式學習已經是很熱門的一種解決方法，深度 Q 學習能夠有效地透過神經網路的大量計算來理解環境，並開始做出相對應的對策，只是相對的需要花上更多的時間來進行理解與計算。而在下一章節中將提到本論文的

主要使用方法 SACD，係屬於深度強化式學習的一種，由 SAC 變種而來，其主要亮點在於能夠用最短的時間來找到多種不同的目標策略，進而應付如此高度動態的流量網路。

第三章 研究方法

3.1 SACD 的資源管理控制

本論文提出一個基於 SACD 的資源管理控制方法 SAC-RAC，應用於透過排隊理論所建立的 5G 網路作為環境，並將可使用的切片數量與即將來臨的網路服務請求作為輸入，進而完成對整體網路環境的管理。具體的架構如圖 3-1 與演算法 2 所示，而有關 SACD 的詳細內容，將在接下來的小節中進行敘述。

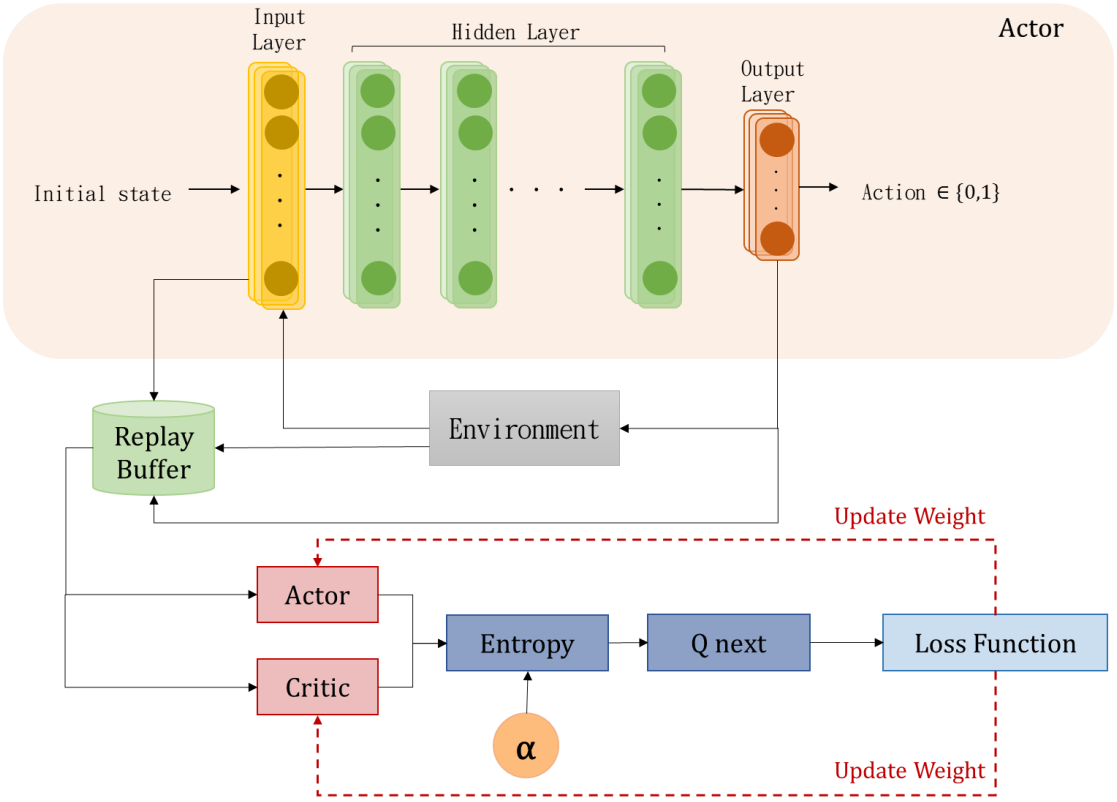


圖 3-1 SAC-RAC 簡易架構圖

演算法 2 ： SAC-RAC	
1. Initialize network $\theta_{e1}, \theta_{e2}, \phi$	➢ Initialize the parameter of agent
2. set $\theta_{t1} \leftarrow \theta_{e1}, \theta_{t2} \leftarrow \theta_{e2}$	➢ Set parameter of target networks
3. set $\alpha \leftarrow 1, D \leftarrow \emptyset$, DLength $\leftarrow 0$	➢ Set entropy weight and replay buffer
4. set exponents $\rho \leftarrow 0.4, \beta \leftarrow 0.6$	➢ Set prioritized replay exponents
5. get slice and QoS flow condition	➢ Get network information from 5G network
6. **for** each iteration **do**:	➢ Training SAC-RAC
7. s ← state of 5G network	➢ Get state of 5G Network
8. a ← $\pi^{\phi}(s)$	➢ Get action from actor
9. r ← r(s,a)	➢ Calculate reward as reward function
10. s′ ← state of 5G network when next service coming	➢ Get next state of 5G network
11. $\delta \leftarrow$ max TD-error in buffer	➢ Make max td error as default
12. $D \leftarrow D \cup \{(s, a, r, s', \delta)\}$	➢ Store (s, a, r, s', δ) into replay buffer
13. $DLength \leftarrow DLength + 1$	➢ Record the length of replay buffer
14. s = s′	➢ Transition to the next state
15. **if** $DLength > predefined\ batch\ size$ and *reward* <	
16. sample transitions $(S, A, R, S', \delta)_i \sim P(j) = \delta_i^{\rho} / \sum_i \delta_i^{\rho}$	➢ Sample transitions according to TD-errors
17. $w_j = \left(N * P(j)\right)^{-\beta} / max_i w_i$	➢ Compute importance sampling for critic loss
18. input transitions to networks to get $Q_{e1}, Q_{e2}, Q_t, \pi^{\phi}$	
19. $\delta_j \leftarrow \lvert min(Q_{e1}, Q_{e2}) - Q_t \rvert$	➢ Update td errors to change priorities Equation
20. compute losses of two critic networks as $\frac{1}{N}\sum_j w_j \left(Q_{ei} - Q_t\right)_j^2$, i = 1,2	
21. compute loss of actor network as Eq. (3)	
22. using stochastic gradient descent update $\theta_{e1}, \theta_{e2}, \phi$	➢ Update parameters with calculated losses
23. e**nd for**	
24. w**hile** 5G network working:	
25. admission control as π^{ϕ}	➢ Run as policy π^{ϕ}
26. Δreward ← |expected reward – reward|	➢ Identify whether traffic conditions have changed
27. i**f** Δreward > expected reward * threshold:	➢ Determine if traffic have changed
28. **end while**	

3.2 Soft Actor-Critic Discrete

在第二章節中所提到的 SAC 可以在許多複雜的環境底下獲得不錯的結果，顯然的非常適合用來解決 5G 網路環境的 RAC 問題，但是其中卻隱含了一個非常致命性的問題，即是 SAC 只適用於連續問題上，並不能解決離散問題。而本文所提到的 5G 網路環境 RAC 問題正屬於離散問題，也就代表著 SAC 方法並不能在此發

揮作用。

而 Christodoulou P 在 2019 年提出了 SAC 的離散化版本 Soft Actor-Critic Discrete(SACD)[31] ，其目的便是希望能夠將 SAC 應用於更多更廣的複雜問題上。

為了將 SAC 進行離散化，分別需要將 SAC 中五項重要的公式進行修改：

- Q network 結構：

  將 Q function 由 $Q:S\times A\rightarrow\mathbb{R}$ 更改為 $Q:S\rightarrow\mathbb{R}^{|A|}$ ，將 soft Q function 的輸出改為每個有可能被選擇的 action 的 Q value，比起原本的做法還要來的更有效，且這種作法在原本的連續空間上是不可能做到的，因為在連續空間中要列出的動作數量可以說是無限多的。

- Policy 結構：

  Policy 策略將不再需要輸出動作分布的均值與共異變數，取而代之的是直接輸出動作分布，將從 $\pi:S\rightarrow\mathbb{R}^{2|A|}$ 修改為 $\pi:S\rightarrow[0,1]^{|A|}$ ，並且在最後使用 softmax 來確保能夠得到一個有效的機率分布。

- Value loss function：

  在原本的 SAC 算法當中，在連續問題的環境底下為了將 soft Q function 中的 $J_Q(\theta)$ 最小化時，因為無法對動作分布直接求期望，因此必須從經驗回放池中進行採樣再做蒙特卡羅估計；但是現在環境題轉移到了離散問題，也就代表了能夠對其直接做計算，這也代表了能將 soft Q value 的公式 (25) 修改為公式 (33) 。

$$V(S_t)\coloneqq\pi_t(s_t)^T(\mathrm{Q}(s_t)-\alpha\log(\pi(s_t))] \tag{33}$$

- Temperature loss function：

如同第 3 點，也可以針對溫度計算的部分做相同的更改，進而減少變異數的估計，如公式 (34) 所示。

$$J(\alpha) = \pi_t(s_t)^T[-\alpha(\log(\pi_t(s_t)) + \bar{H})] \quad (34)$$

- Policy loss function：

在先前的 SAC 做法中，為了使 $J_\pi(\phi)$ 最小化而使用重參數的技巧，但在離散化後便可以直接對其計算期望，因此策略的新目標函式將由公式 (31) 改為公式 (32)。

$$J_\pi(\phi) = E_{s_t \sim D[\pi_t(s_t)^T[\alpha \log(\pi_\phi(s_t)) - Q_\theta(s_t)]]} \quad (35)$$

綜合上述改動，SACD 的整體架構如演算法 3 所示。

演算法 3 Soft Actor-Critic Discrete 離散化演員評論家

Initialize
$Q_{\theta_1}: S \to \mathbb{R}^{|A|}, Q_{\theta_2}: S \to \mathbb{R}^{|A|}, \pi_\phi: S \to [0,1]^{|A|}$ ➢ Initialize local networks
$Q_{\theta_1}: S \to \mathbb{R}^{|A|}, Q_{\theta_2}: S \to \mathbb{R}^{|A|}$ ➢ Initialize target networks

$\bar{\theta}_1 \leftarrow \theta_1, \bar{\theta}_2 \leftarrow \theta_2$ ➢ Equalize target and local network weights
$D \leftarrow \emptyset$ ➢ Initialize an empty replay buffer

for each iteration **do**
  for each environment step **do**
    $a_t \sim \pi_\phi(a_t|s_t)$ ➢ Sample action from the policy
    $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$ ➢ Sample transition from the environment
    $D \leftarrow D \cup \{(s_t, a_t, r(s_t, a_t), s_{t+1})\}$ ➢ Store the transition in the replay pool
  end for
  for each gradient step **do**
    $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in \{1,2\}$ ➢ Update the Q function parameters
    $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$ ➢ Update policy weights
    $\alpha \leftarrow \alpha - \lambda \hat{\nabla}_\alpha J(\alpha)$ ➢ Adjust temperature
    $\bar{Q}_i \leftarrow \tau Q_i + (1-\tau)\ Q_i$ for $i \in \{1,2\}$ ➢ Update target network weights
  end for
end for
Output : θ_1, θ_2, ϕ ➢ Optimized parameters

3.3 優先經驗回放(Prioritized Experience Replay)

在一般深度強化式學習的訓練過程當中，都會產生數個由 (s_t, a_t, r_t, s_{t+1}) 所組成的位元組，稱之為 Transition，主要是用於紀錄訓練過程的經歷，而添加經驗回放的機制便是將 n 條 Transition 收錄在 replay buffer 中，其中 n 為 replay buffer 可容納的總數，若是超過則會將最舊的經驗刪除。使用此作法的目的是要能夠有效利用訓練過程中所發生過經驗，避免浪費，同時使得整體訓練效果增加，其簡易的示意圖如圖 3-2 所示，基於以上本文將該機制添加進 SACD 的方法中。

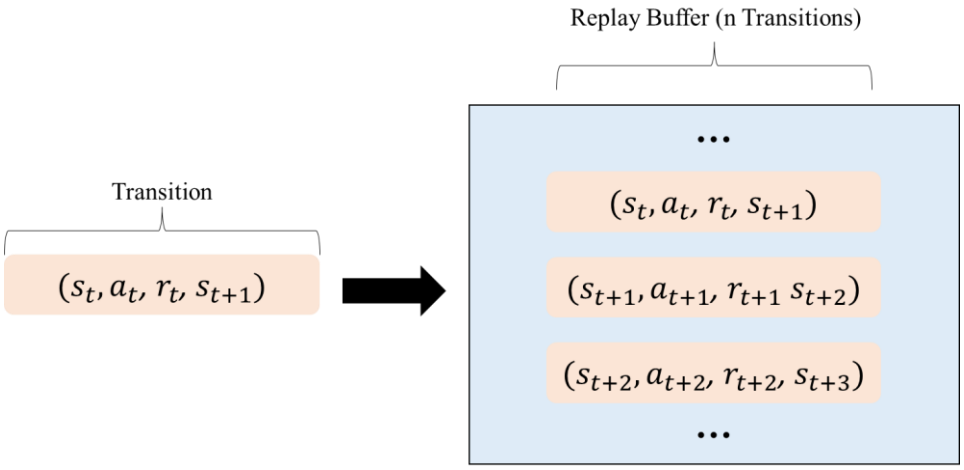


圖 3-2 經驗回放機制簡易示意圖

但是當面臨的環境逐漸複雜時，這便代表著被儲存的經驗將會逐漸增多，而在相似情況下甚至近乎相同的經驗也會不斷增加，若是一直對這樣的 replay buffer 進行均勻抽樣時，便會導致相似的經驗反覆被提取出來，使得較少出現的經驗非常難以被抽取出來做訓練，導致 agent 常常會只能應付時常重複出現的 state，當出現了少樣本數的特例時，反而容易表現不佳。

對此 Schaul T 等人在 2016 年提出了優先經驗回放的機制 [32]，為的就是要讓這些較少出現的特例擁有較高被選取的優先序，使得他們更容易被採樣出來進行訓練，更因為調整了 replay buffer 中的排序，打斷了每條 Transition 的相關性，

能夠有效的史訓練成果變好。優先經驗回放的具體作法是將 replay buffer 中的每一條 transitions 分別添加了一個 TD error，記為 δ_t，當該條經驗的 δ_t 偏小，則被認為已多次訓練，因此誤差較小；反之則是因缺乏訓練而誤差偏大，因此每條經驗被抽樣的機率可以用兩種方式來取代，其一是與 $|\delta_t|$ 成正比，如公式 (36) 所示，而 ϵ 為一個極小的數，用以避免機率被設為 0；其二是針對 $|\delta_t|$ 做排序，使得較大的值往前排放，將能夠更加優先被取出，即與 $|\delta_t|$ 成反比，如公式 (37) 所示。

$$p_t \propto |\delta_t| + \epsilon \tag{36}$$

$$p_t \propto \frac{1}{rank(t)} \tag{37}$$

然而，因為採樣方式更改成了非均勻抽樣，會使得每一個 transition 被抽到的機率都不相同，如果這時所有的學習率 α 還是維持全部統一調整的話，就會造成誤差越來越大，因此作者也提到了學習率的更新方式應該改為公式 (38)，其中的 β 作者也建議在一開始應該設的盡量小一點，而後隨著訓練時間逐漸增大直到 1 為止。

$$\alpha \cdot (np_t)^{-\beta}, \beta \in (0,1) \tag{34}$$

圖 3-3 為優先經驗回放機制的整體示意圖，包含了新的 Transitions、抽取的方式以及學習率調整方式。

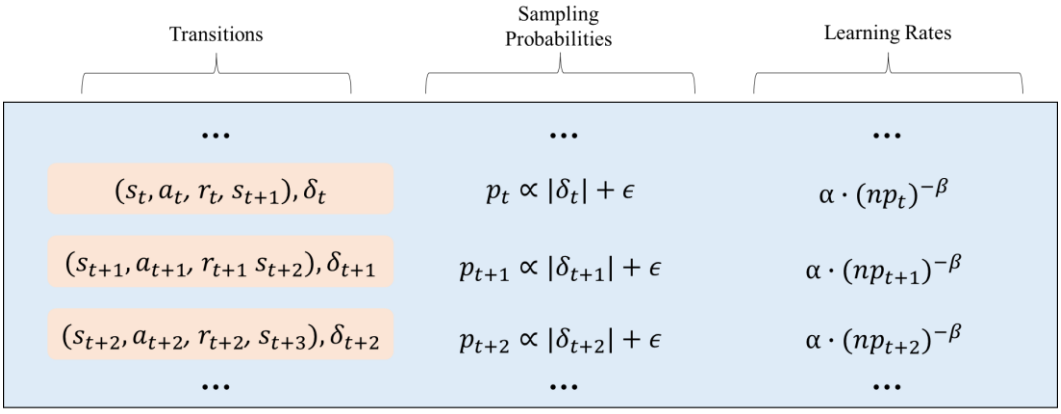


圖 3-3 優先經驗回放簡易示意圖

3.4 有條件的學習方法(Conditional Learning)

如前述所言，RAC 的環境是一直不斷在變化的，每一次的變動都會導致訓練過程中 agent 的獎勵收斂會非常不穩定。雖然適量的收斂浮動能夠使得 agent 適應更多樣的環境，但若是一直無法穩定浮動的區間，則會導致最終訓練成效非常差勁。對此，本論文在 agent 的訓練過程中添加了一種有條件的學習方法，即是透過超參數來限制 agent 對經驗池進行抽樣並學習的次數。在訓練過程中，只要經驗池中的存放量足夠且當前回合獎勵收益低於超參數所設定的閥值，agent 便會定期的進行抽樣且學習，反之，若是當前回合獎勵收益高於所設定之閥值，便不會對神經網路進行更新，而是繼續前往下一個回合進行訓練。透過此種做法，不僅能夠有效的穩定收斂的浮動區間，同時也因為降低了對神經網路更新的次數，因此能夠減少訓練所花費的時間。

3.5 系統架構描述

本論文將流程分為三個部分，首先是先調整好要模擬的網路環境，接著調整參數進行訓練，最後針對訓練好的 agent 來測試效果，整體架構圖如圖 3-4 所示，以下為補充說明：

1. 環境可針對不同流量大小來進行模擬，可供的流量種類為 1~4 種不同優先序的服務，並使用 lambda 參數來調整數量多寡，若為 0 則代表不製造此服務。
2. 開始訓練前 SAC-RAC、DQN、Greedy 可選擇要提供多少切片來給 agent 進行分配，Fixed Slice 則需手動調整，將切片盡可能的平均分給不同的服務，在本次實驗中分別給予 30、40、50 以及 60 的單位的大小，並且可以決定獎勵的閥值來控制 agent 訓練的時間。

3. 測試時分別對不同的方法進行 50 次的實驗，並將結果取平均，再對各方法的結果進行比較。

本論文的目標是期望能找到一個深度強化式學習的方法，能夠在最短的時間內收斂，並且能夠盡可能的拿到好的成果，同時也要能夠應付不同流量環境底下的輸入，藉此維護網路環境品質。

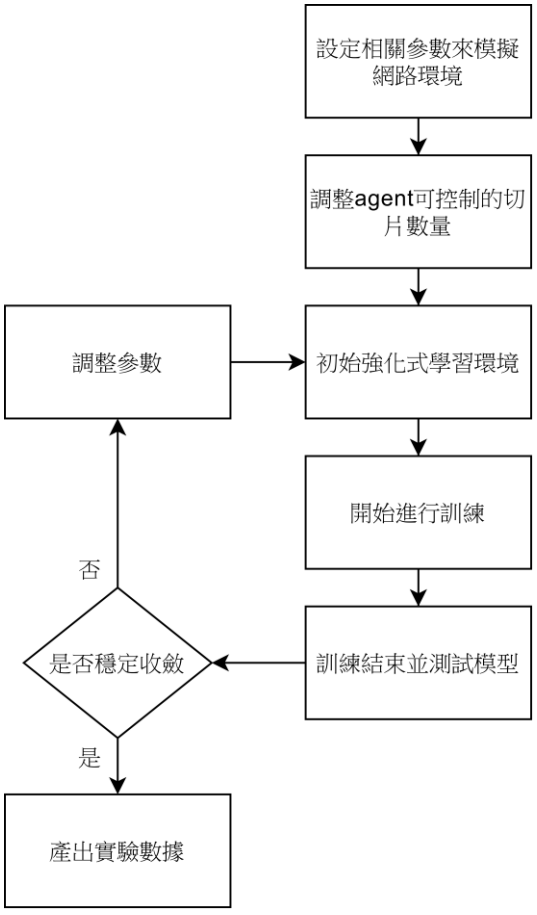


圖 3-4 研究方法之流程圖

3.6 深度強化式學習環境描述

3.6.1 狀態空間(state space)

本論文的狀態空間將當前的網路環境狀態當作輸入，由六個位元組成的一個數組，其中分別記錄了當前占用中的切片數量、等級 1~4 的服務總數以及下一個

即將到來的服務等級。

3.6.2 動作空間(action space)

本論文中 agent 的動作僅有兩種選擇，根據當前環境的資訊來決定要接受或拒絕抵達的服務，此外在實驗環境中並沒有佇列等待的選項，因此當 agent 選擇拒絕以後，該服務即會被阻擋並進行丟棄。

3.6.3 獎勵函數(reward function)

當 agent 完成一個動作後即可透過獎勵函數來獲得回饋，藉此評斷此次動作的好壞，進而將策略學習好。在本論文中，獎勵會依照服務的等級來給予相對應的獎勵，服務分別有 1~4 個等級，而獎勵分別為 (1,2,4,8) ，而獎勵函數如公式 (34) 所示，其中 R 代表獎勵， a_t 表示當下的動作， $level_t$ 為當下所來的服務等級。

$$R = a_t \cdot 2^{level_t - 1} \ , a_t \in \{0,1\} \tag{34}$$

不過，本論文的最終目的為穩定維持網路環境，若是 agent 的行動導致網路環境超出負荷，即接受的服務量大於切片數量，則會給予嚴厲的負懲罰 R= -100，來促使 agent 能夠避免此情況。

第四章 模擬環境與實驗結果

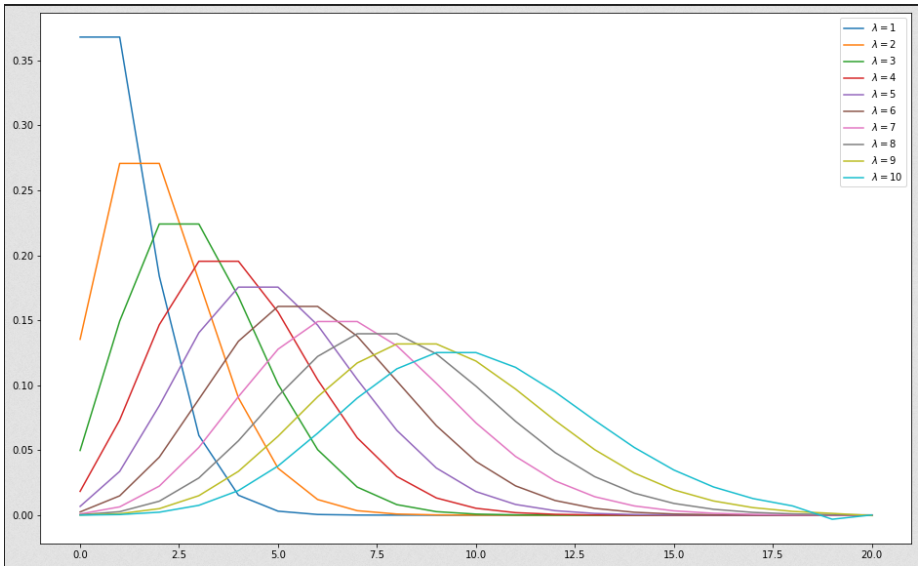
本論文的所有實驗、測試與結果分析皆在研究室中的其中一台 server 中運行，該台電腦的作業系統為 Ubuntu 18.04，配有 12 核心的 Intel i9-9900K 且為 3.6GHz 時脈的 CPU，記憶體規格為 40GB，顯示卡則為 NVIDIA GeForce RTX 2080 Ti。而本次實驗中所使用的 SACD 方法因使用於資源允入控制上，將其稱為 Soft Actor-Critic Resource Admission Control (SAC-RAC) ，除了提出的此方法以外，同時也會和添加了優先經驗回放的 DQN、傳統貪婪式做法以及經典固定切片方法來做比較。

本論文所有實驗上方法皆使用 python 3.6 來進行實作與測試。神經網路參數僅有 agent 數量共同為 3 個，並且訓練回合數都為 1000 個回合，其餘細微參數將依照方法的不同來分開對其進行優化。

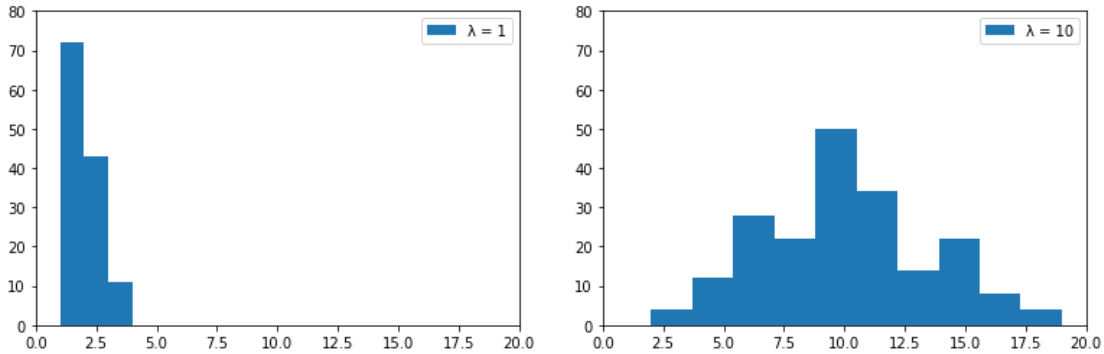
為了能夠證明本論文提出方法的有效性，在實驗上將模擬多種不同流量的網路環境，並且以最終所獲得的獎勵來做為最主要的指標並進行比較。在實驗上，所有的方法都將重複執行 50 次，並將過程的獎勵做平均，以盡可能的降低因環境隨機性所帶來的誤差。當實驗完成後，除了最終獎勵以外，同時針對每組實驗記錄的網路壅塞率來進行比較，在兩組深度強化式學習方法中，也會將訓練過程所耗費的時間以及收斂程度來做比較。詳細實驗結果將於接下來作呈現。

4.1 實驗環境

呈如前各章節所述，在實驗開始前需要先以公平的方式先行產生合理的流量服務，因此本論文使用了 Simpy 3.0.11 的套件來產生符合卜瓦松分布的流量。為了證明該套件所產生的分布確實有符合卜瓦松分布，圖 4-1 分別展示了以 Simpy 所產生的流量分布圖以及數學定理上卜瓦松分布的分布圖，由圖 4-1 可得知，Simpy 所產生的流量確實有符合該分布。



(a)卜瓦松分配之數學理論分布圖



(b)Simpy 產生之流量分布圖

圖 4-1 流量分布對比

為了確定本論文所提出的方法是否能夠在各種不同的流量環境中維持一定的QoS，因此在實驗中在訓練與測試時皆採用了不同的服務流量與切片數量來進行訓練，所使用的數量如表 1 所示。本實驗將服務分為四個等級，例如 [10,0,0,10] 則代表將服務 1 與 4 的 λ 設為 10，服務 2 與 3 為 0，在測試環境中流量數將從 0.5 開始輸入，以每回合增加 0.5 的規律直到增加至 10。而針對 SAC-RAC 與 DQN 的神經網路中較為重要的相關參數條列於表 2 表 1 中。

表 1 測試與訓練流量數

訓練流量數	測試流量數	切片數量	環境代號
[10,0,0,10]	[10,0,0,10]	30	環境 1-1
		40	環境 1-2
		50	環境 1-3
	[10,10,10,10]	30	環境 2-1
		40	環境 2-2
		50	環境 2-3
[15,0,0,15]	[10,0,0,10]	30	環境 3-1
		40	環境 3-2
		50	環境 3-3
[10,10,10,10]	[10,10,10,10]	30	環境 4-1
		40	環境 4-2
		50	環境 4-3
		60	環境 4-4

表 2 實驗參數表

參數名稱	SAC-RAC	DQN
runs per agent	3	
discount rate	0.9	
batch size	1024	1500
buffer size	10000	
episode	1000	
learning rate	actor : 0.001 critic : 0.01	0.001
linear hidden units	actor : [50,100,50,10] critic : [50,100,50,10]	[100,200,100,50]

4.2 實驗結果

4.2.1 不同流量的測試環境

首先針對較為基本的環境開始進行測試，以 30 個切片、服務流量為 [10,0,0,10] 做訓練，以 [10,0,0,10] 做測試。圖 4-2 為測試結果，其詳細數據於附錄表 3，縱軸代表該方法所獲得的獎勵以及網路環境壅塞率，前者需越高越好，後者則相反；橫軸表示當前環境中所輸入的流量數。從結果中可觀察到，當環境中的流量數逐漸增多的情況下，SAC-RAC 的方法能夠更有效的穩定並維持良好的 QoS，同時在壅塞率的方面在比起其他的方法也能同時維持較低的數值。

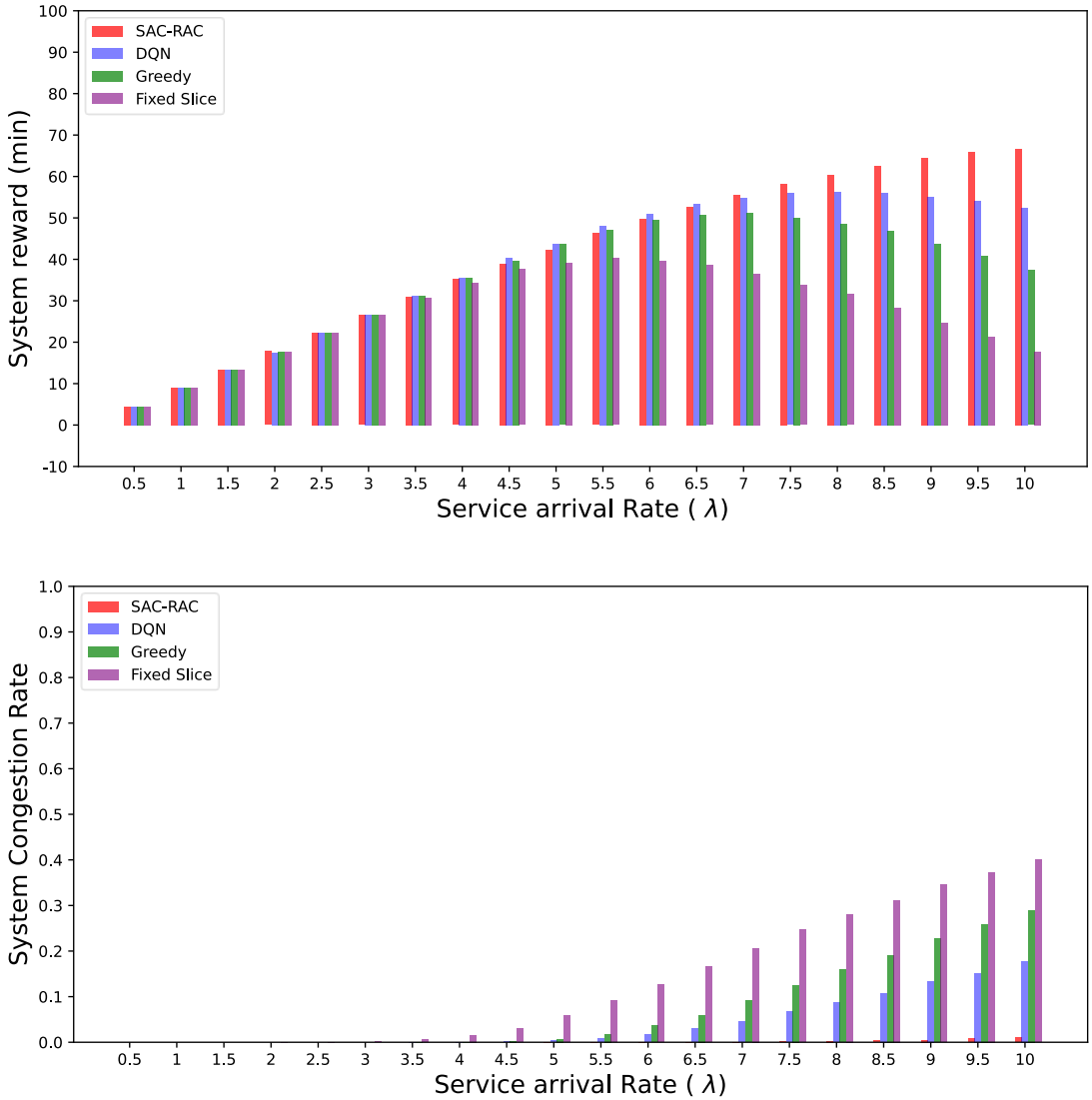


圖 4-2 環境 1-1 實驗結果

圖 4-3 為 SAC-RAC 與 DQN 在訓練時每個 Epoch 所獲得的獎勵，可以發現 SAC-RAC 充分的發揮了優勢，在非常短的時間內就已經完成收斂，並且持續穩定；相比 DQN 則需較長的時間才能夠收斂至一個穩定的區間，在這之前則需要較長的收斂時間以及面對動態環境時所會遭受到的不穩定收益。

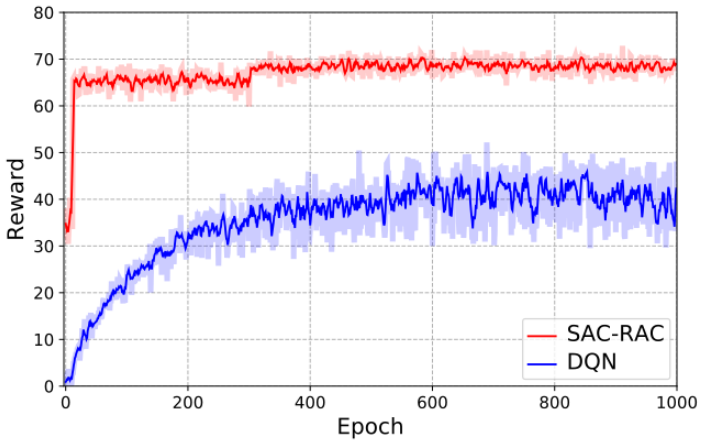


圖 4-3 環境 1-1 Epoch 收斂曲線圖

在相同的訓練架構下，本實驗也模擬了一個不同的流量環境，該環境的流量更相近於日常生活中，在 24 個單位時長內流量將不再固定產生，而是會有相對應的動態流量高峰潮與離峰潮，盡可能的模擬出日常生活中會出現的網路流量，例如在半夜時使用者較少使得流量較少，反之在白天時期使用者增多而造成流量的增加，圖 4-4 為實驗結果，由上而下分別為獲得的獎勵、壅塞率以及當下的流量數，橫軸則為模擬經過的時間單位。本實驗針對收益效果較好的兩種深度強化式學習方法來做測試，結果表明，SAC-RAC 方法越在流量高峰潮時越能夠表現出自身優勢，穩定的有更高的成果收益，且也能夠維持較低的壅塞率。

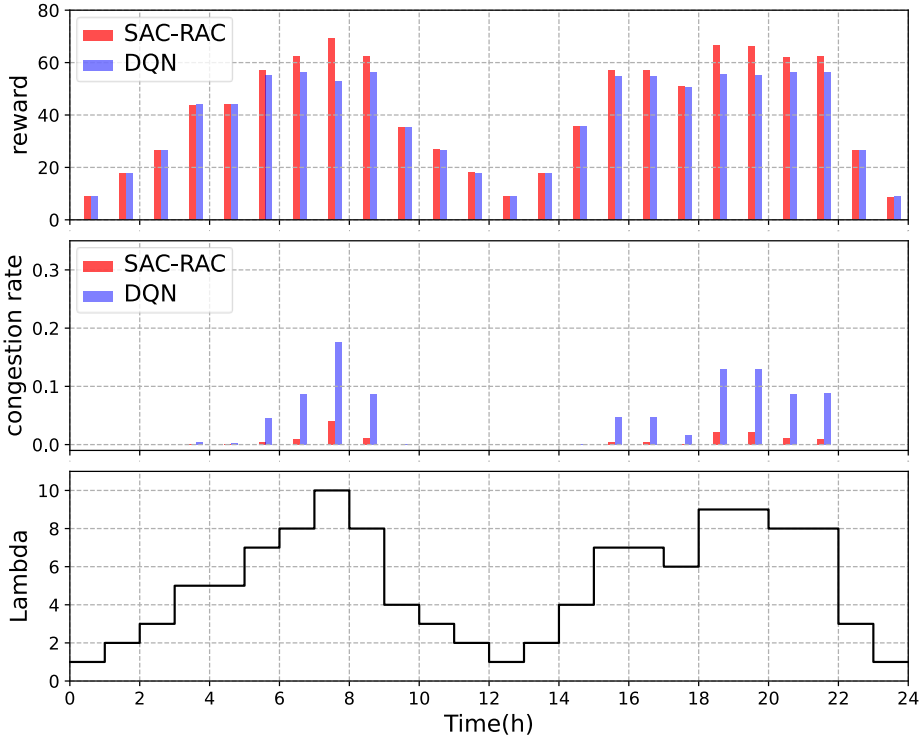


圖 4-4 環境 1-1 動態流量網路測試結果

接著使用下一組環境 2-1 的參數來進行實驗，也就是將測試環境的流量放大一倍，修改為 $[10, 10, 10, 10]$ ，這意味著即將來臨的服務數量也將放大一倍，目的是想了解若是使用較少的流量數進行訓練，若是遭遇流量較大的環境時是否能夠應付。依照常理判斷，這樣的實驗結果應該不會太過理想，因為實際測試時已經大幅超出訓練時的狀況。而其實驗結果於圖 4-5 中顯示，其詳細數據於附錄表 4，四種方法比起前一個實驗所獲得的獎勵皆有所下降，甚至出現負數，與前述推論的結果相符，這樣的訓練方式無法負荷如此龐大的流量請求。儘管如此，SAC-RAC 還是能夠獲得相對來說較高的收益，以及較低的壅塞率。

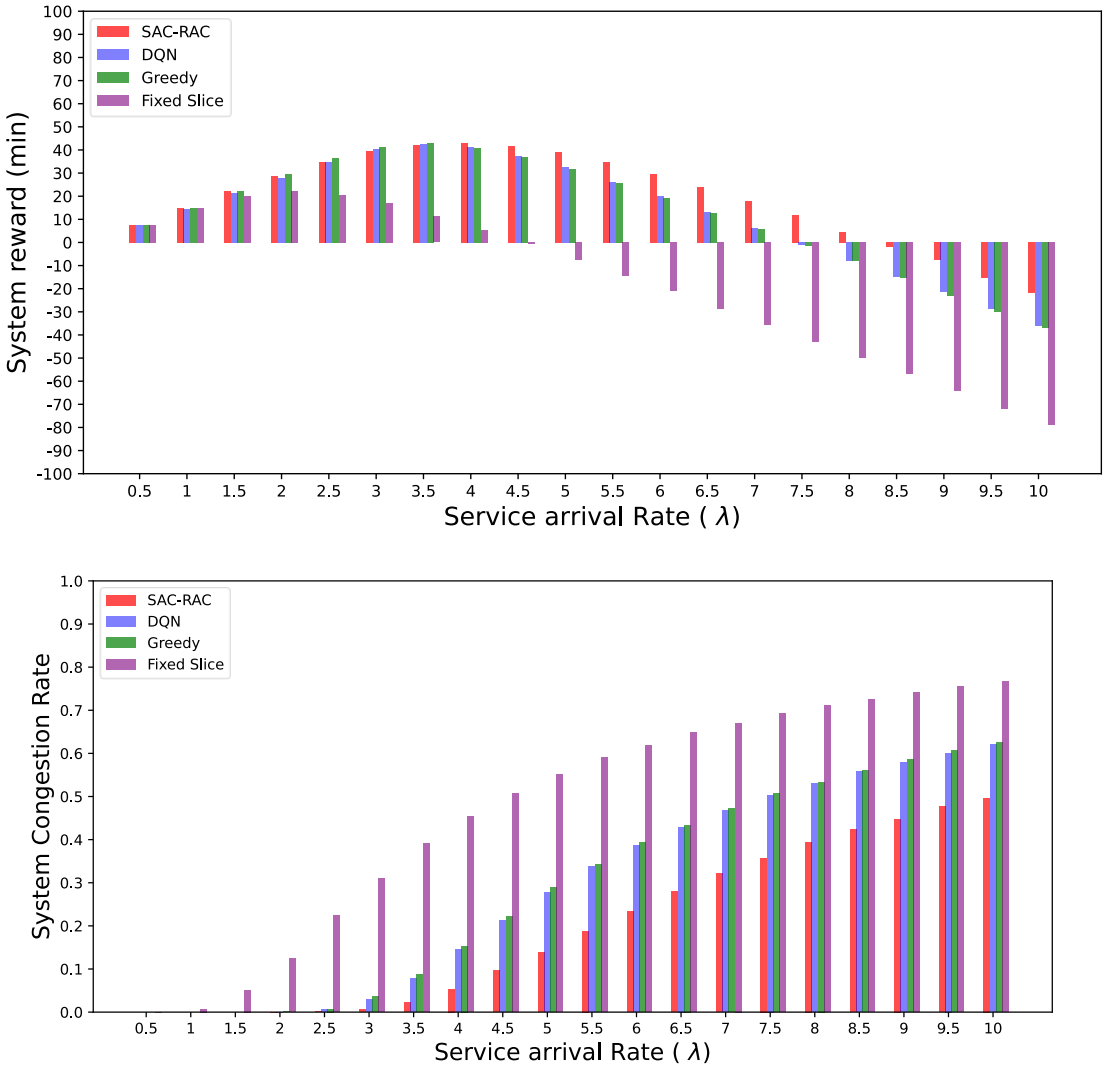


圖 4-5 環境 2-1 實驗結果

經由上一個實驗結果得知，若 agent 面臨比訓練時還要大的流量，容易表現不佳，因此環境 3-1 嘗試將訓練時的流量總數提高，使其比測試時的流量更多，目的是為了瞭解這樣的以大流量的訓練方式是否能夠正常應付小流量環境，進而避免環境 2-1 所遭遇的困難。圖 4-6 為實驗結果，其詳細數據於附錄表 5，可以發現其結果與環境 1-1 非常相似，也就代表著在條件允許下使用較大的流量來進行訓練是可行的方法之一，不過相對應的代價便是比起環境 1-1 來說，需要更多的訓練時間。

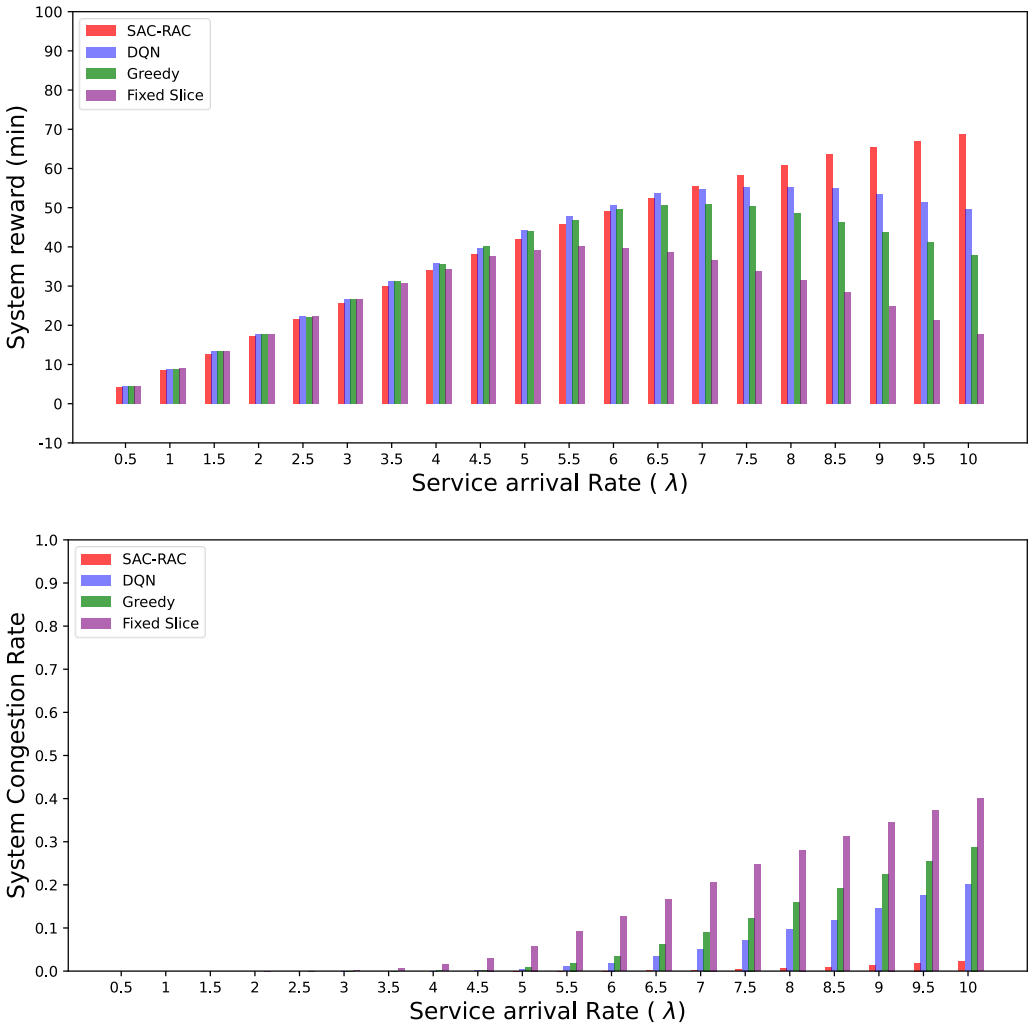


圖 4-6 環境 3-1 實驗結果

4.2.2 不同切片的測試環境

經由前述實驗結果發現，無論在任何訓練與測試條件下，在實驗前期所有方法的結果皆不相上下，從壅塞率的結果來判斷，推測是因為當前的流量數較小，且切片容量足夠，因此就算接受所有的服務也不會造成網路環境崩潰，導致結果非常相似。為了證實切片數量會影響實驗結果，因此在相同的訓練環境底下，分別對上述測試環境採用 40 與 50 個切片，即環境 1-2、1-3、2-1、2-2、3-1、3-2 來進行實驗。首先環境 1-2、2-2 與 3-2 的實驗結果顯示如下圖 4-7 到圖 4-9，其詳細數據分別對應附錄表 6 至表 8。從實驗結果發現，當切片數量增加至 40 時，所有方法的收益

都顯著的向上增長，無論在獎勵或是壅塞率的表現上都比 30 個切片還要來的更加優秀。

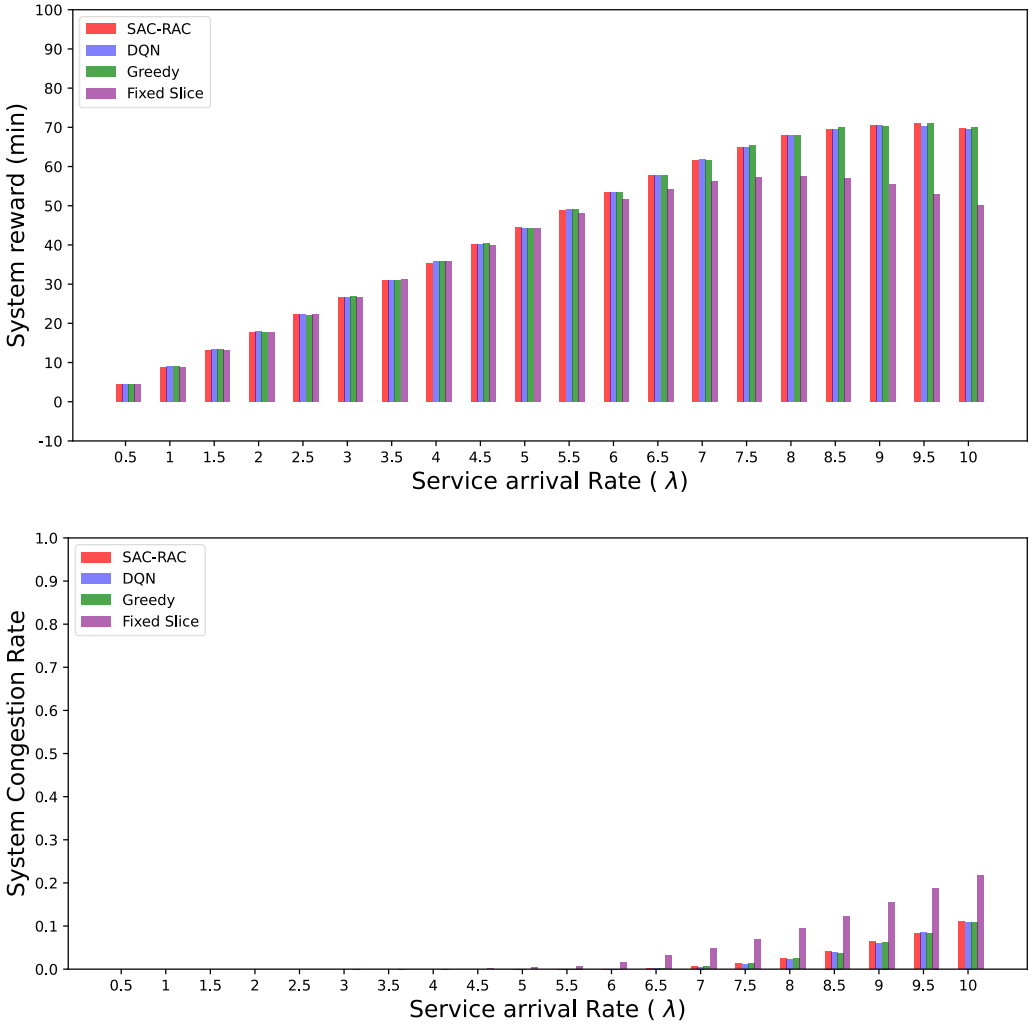
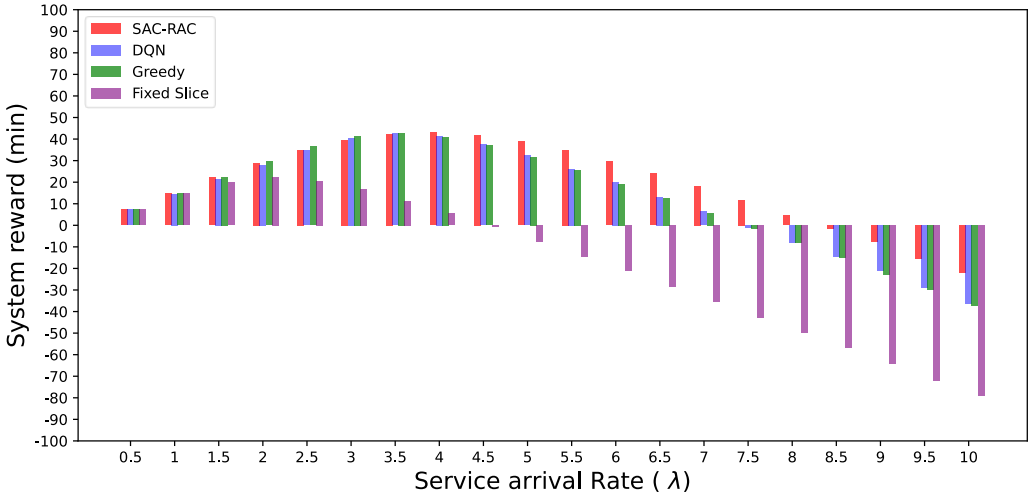


圖 4-7 環境 1-2 實驗結果



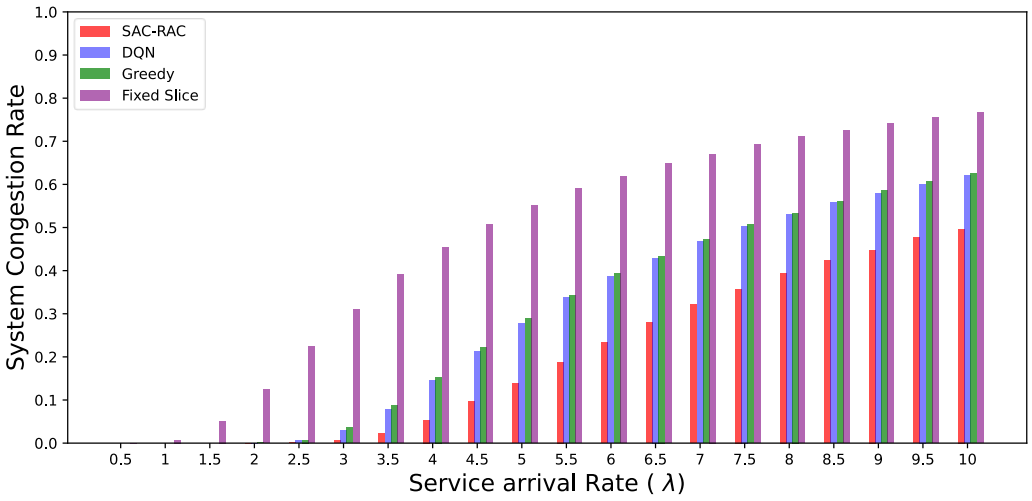


圖 4-8 環境 2-2 實驗結果

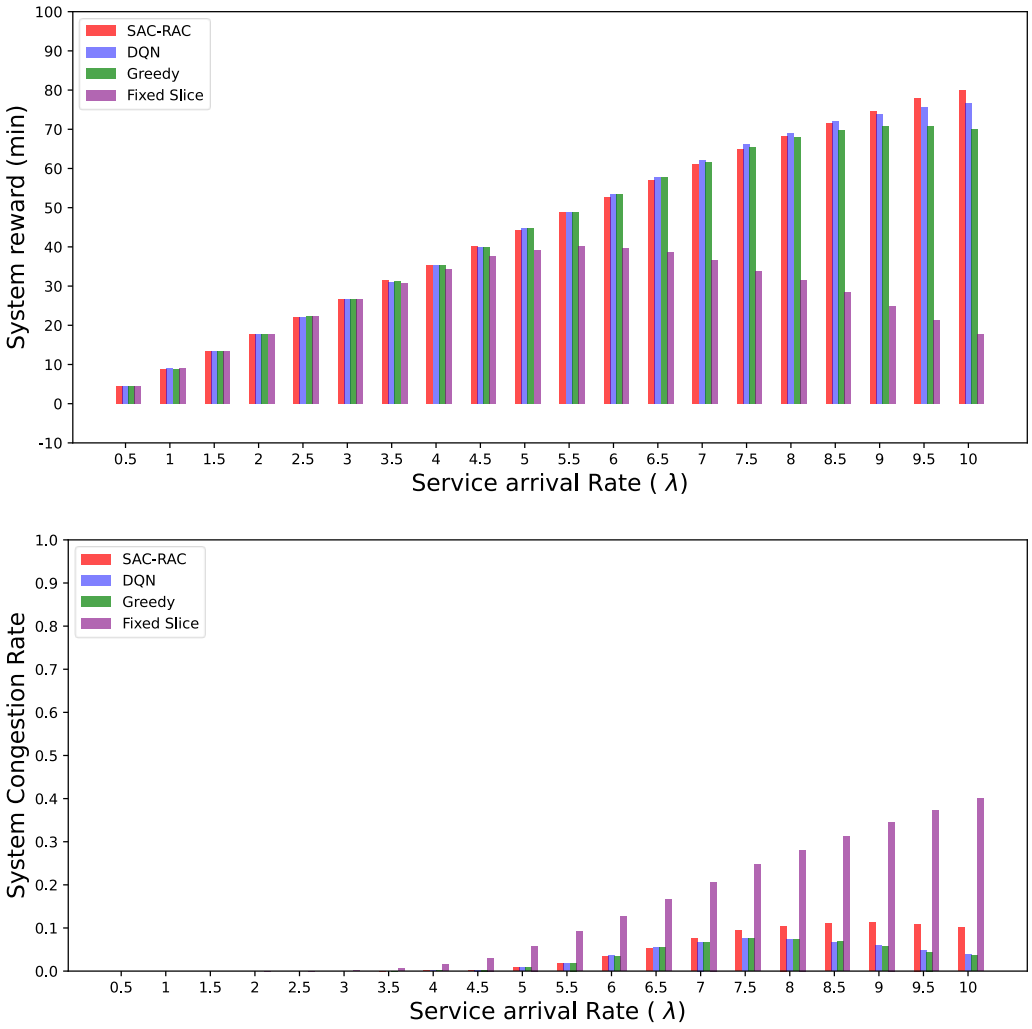


圖 4-9 環境 3-2 實驗結果

經由上述實驗結果發現，當切片數量增加至 40 以後所有方法的結果都有顯著提升，不過整體效益增加了，但是當服務流量逐漸增加時依舊會有會有下降的趨勢，為更進一步確認是否因切片所影響，故將切片數量再度往上增加至 50 以證實其之間的相關性，若是增加後能夠再更進一步提升表現，則代表與切片數量有著一定程度上的關聯。環境 1-3、2-3 與 3-3 為相同訓練與測試架構下，將切片數量更改為 50 的環境，其測試結果如下圖 4-10 到圖 4-12 所示，其詳細數據分別對應附錄表 9 至表 11。從實驗結果可以得知，效果比起 40 個切片再更顯著上升，由此可以證明無論在何種訓練以及測試架構，切片數量都有一定可完全負荷的區間，當抵達的流

量數目超越了此區間時，才是開始考驗各個方法在調配資源上的差異。

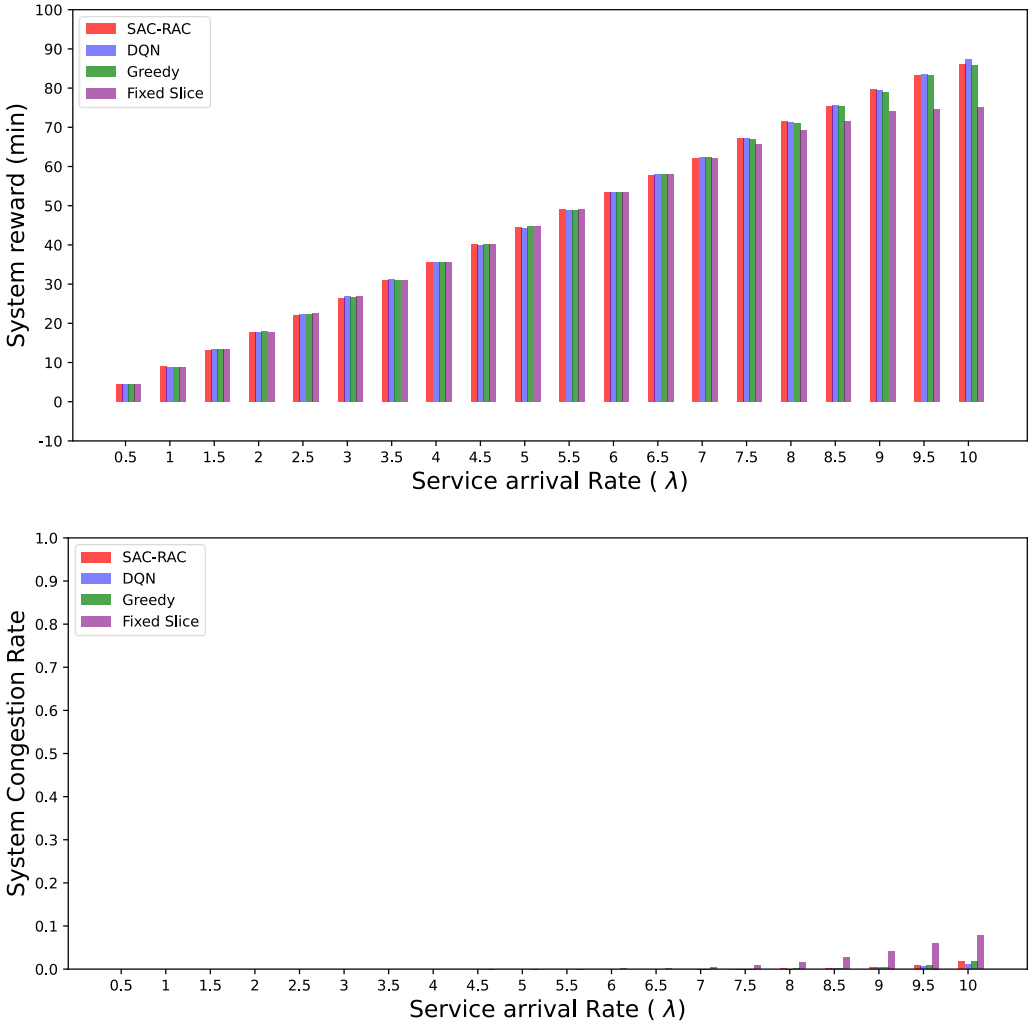


圖 4-10 環境 1-3 實驗結果

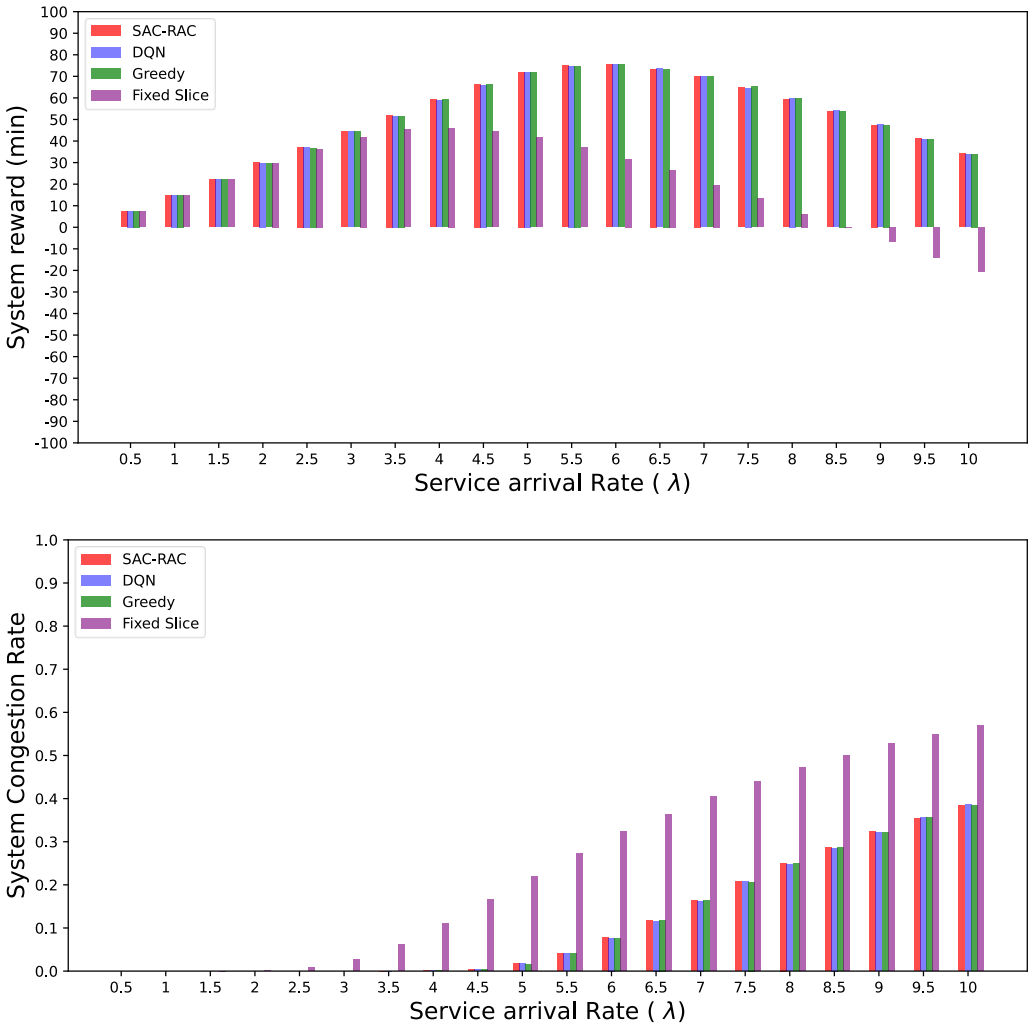


圖 4-11 環境 2-3 實驗結果

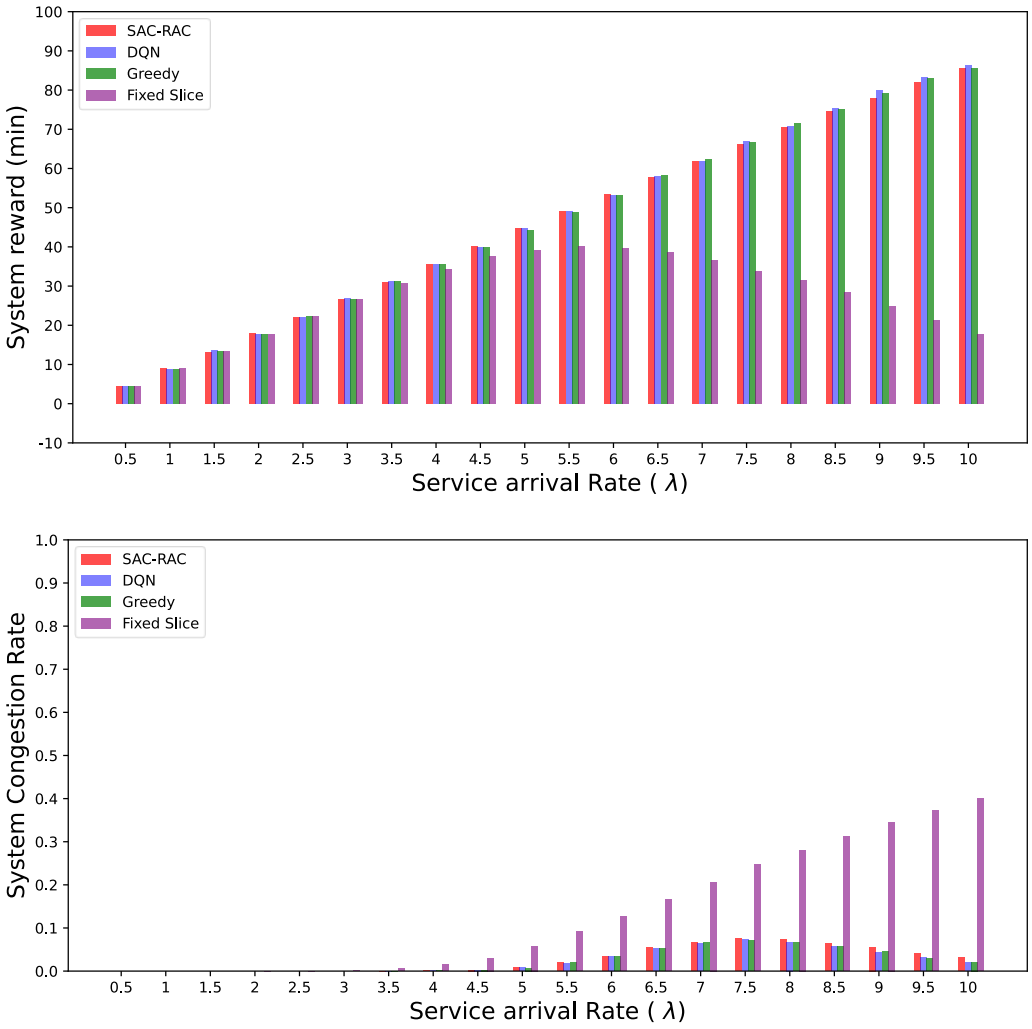


圖 4-12 環境 3-3 實驗結果

經由環境 2-1 至 2-3 的實驗結果發現，想使用較少的服務進行訓練並且應用於較大的環境中雖然可行，但是並不能保證一定能夠獲取較好的結果。從環境 3-1 至 3-3 也發現於訓練時提前放大訓練數量也不會使效能提升。為了確認原因除了在於切片數量以外，若是以與環境 2-1 至 2-3 相同的服務流量進行訓練，是否會有不同的結果產生，因此接下來將對環境的訓練架構進行調整，並觀測其造成的影響。接下來所使用的環境 4-1 至 4-4 皆是以 [10,10,10,10] 的大流量數進行訓練，且以 [10,10,10,10] 來做測試，其中使用不同的切片數量。而以目前為止的實驗表明此類以較大的流量進行訓練是可以負荷較小的流量測試的，因此接下來並不會對

[10,0,0,10] 的環境進行測試。

4.2.3 不同流量的訓練與測試環境

圖 4-13 為環境 4-1 之實驗結果，其詳細數據於附錄表 12，SAC-RAC 在前期雖然收益略顯劣勢，但在後期流量數逐漸增長時便慢慢地展現出自身優勢。而此環境的測試架構與環境 2-1 相同，但在不同的訓練架構下，以大流量進行訓練的環境 4-1 中 SAC-RAC 不僅取得了比環境 2-1 更好的結果，更是所有方法中後期唯一將獎勵收益維持於正數的方法，因此推測訓練環境也會間接地影響到收斂結果，且證明 SAC-RAC 是個能夠有效進行管理的方法。

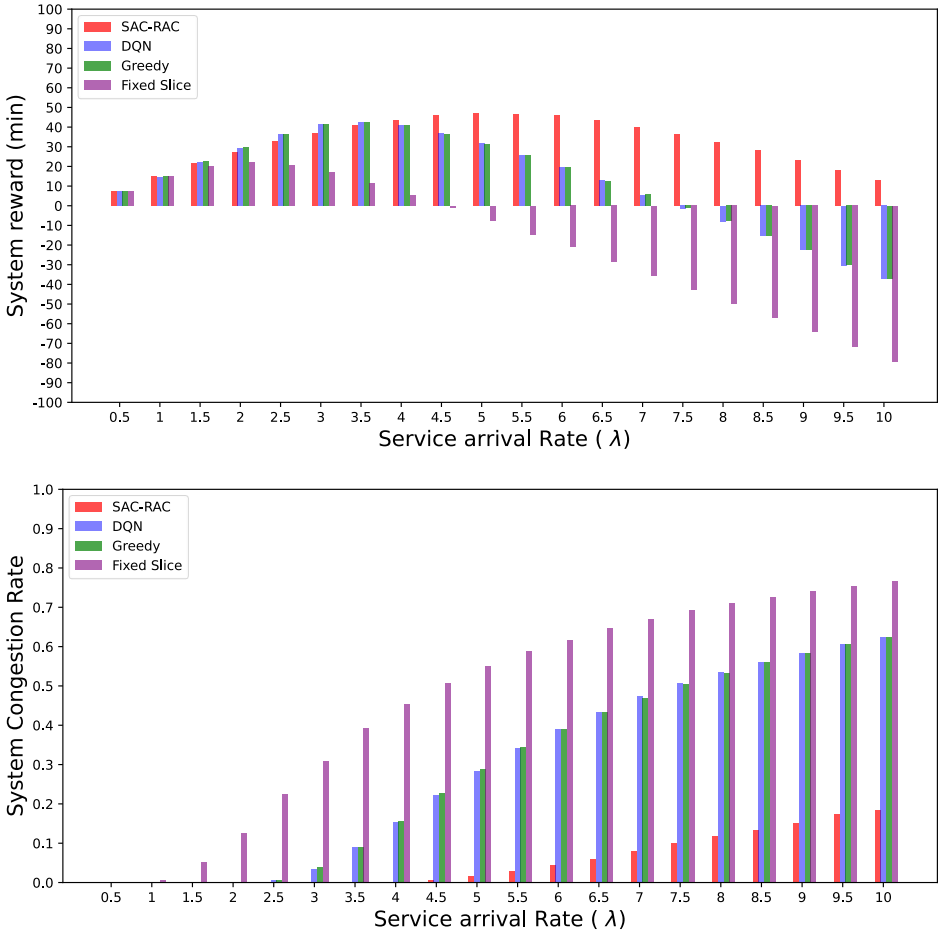


圖 4-13 環境 4-1 實驗結果

經由前幾小節的實驗發現，切片數量會顯著的影響收斂結果，因此從環境 4-2

開始將逐漸增加其中的切片數量，嘗試是否能夠增加其收斂效益。以下圖 4-14 為環境 4-2 之實驗結果，其詳細數據於附錄表 13，增加 10 個切片後，在 40 個切片的環境下 SAC-RAC 開始慢慢穩定收斂，不僅收益已經回歸正數，在前期也與其他方法持平，而在壅塞率部分也依舊穩定的保持較低的水平。

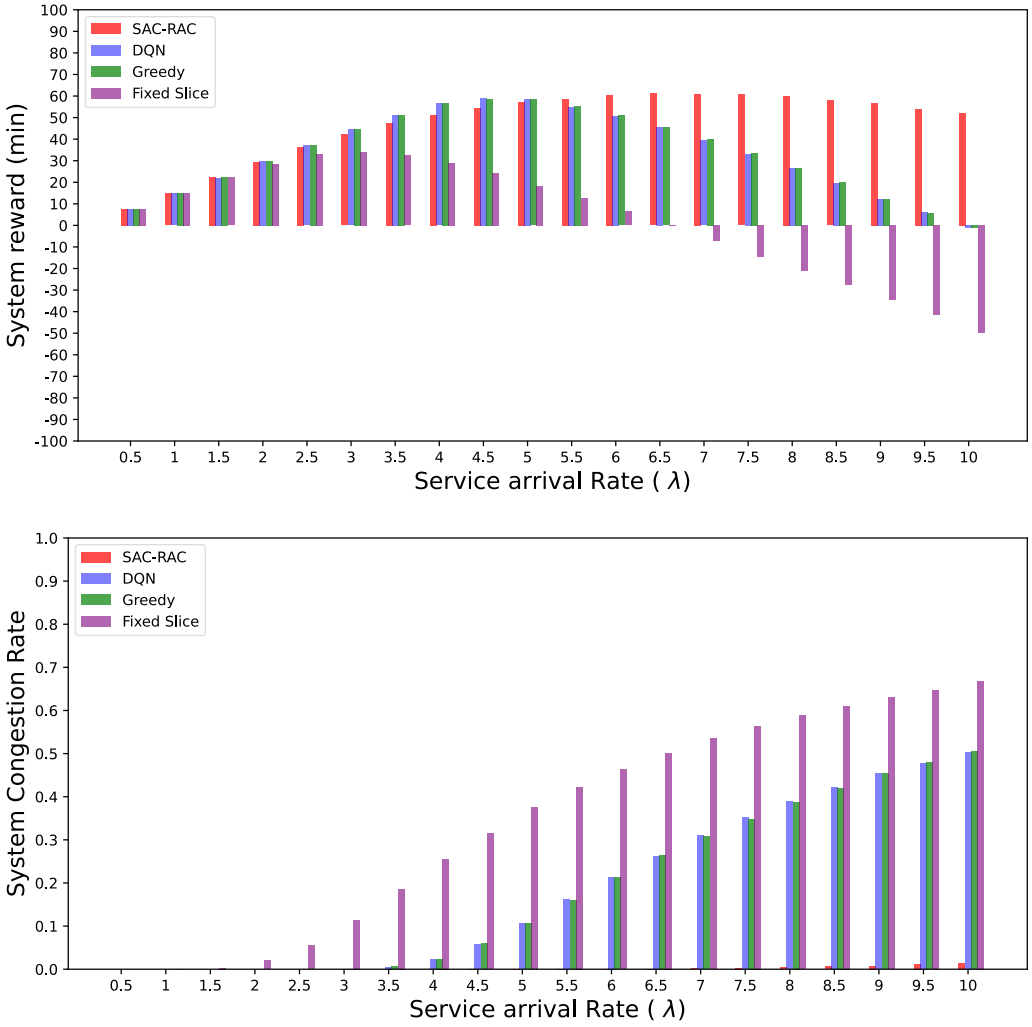


圖 4-14 環境 4-2 實驗結果

附錄

表 14 與圖 4-15 為環境 4-3 之實驗結果，與環境 2-3 相同，將切片數量增加至 50 個，所有的方法獎勵收益便都開始增加，且 SAC-RAC 更優於在環境 2-3 的結

果，並且保持穩定的領先，由此實驗結果可得知，與前述實驗結果相符，確定無論是訓練架構或是切片數量，確實會顯著影響最終方法的收益結果。

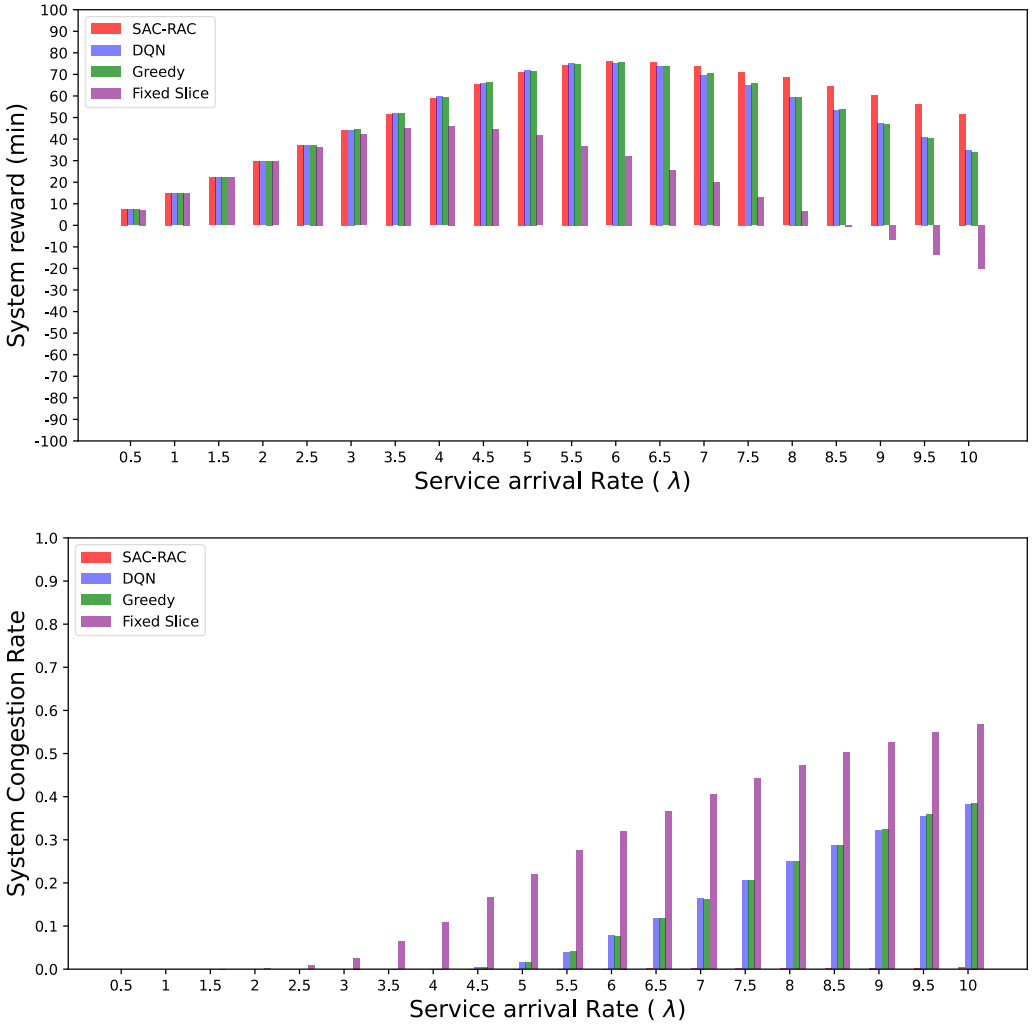


圖 4-15 環境 4-3 實驗結果

最後，將切片數量提升至 60，也就是最初環境的 2 倍，實驗結果如附錄表 15 與圖 4-16 所示。結果顯示，所有的方法皆收斂至正數，SAC-RAC 依然處於優先位置，不過可以發現在大多數實驗中貪婪式做法與 DQN 的收斂結果近乎一致，這代表了因為切片數量足夠負荷，因此即使不做任何策略性的分配也可以獲得不錯的效益，因此可以得知切片數量依舊是影響最大的主因之一，不過因為不可能無上限的提供切片數量，因此如何透過有效的方法來穩定收益才是最重要的。

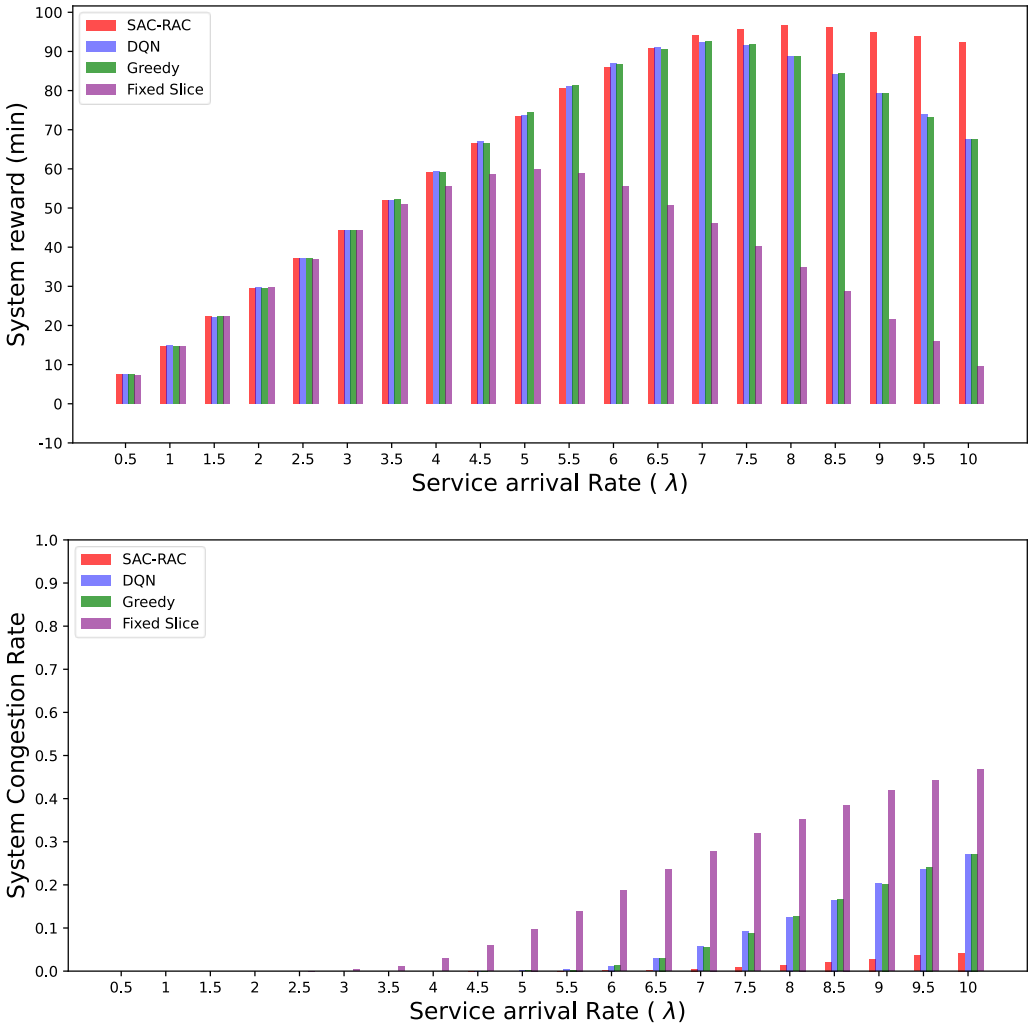


圖 4-16 環境 4-4 實驗結果

第五章 結論與未來展望

5.1 結論

本論文提出一個基於深度強化式學習的架構且針對 5G 動態流量網路環境的問題來做解決。在研究問題上必須要盡可能的在高度動態流量的網路環境中維持住高水平的收益，並且還得要維持高質量的 QoS，也就是避免網路癱瘓與壅塞。在主要方法上，本篇論文提出了一種基於 SACD 的變種方法，使其能在此離散的環境底下作用。此外，本論文更針對該方法進行架構上的改良，新增了優先回放機制並且調整該深度強化式學習方法所要進行抽樣學習的時機，以利盡可能的提升收益與訓練速度。最後在實驗上本論文分別產生了三大種不同訓練架構且各自擁有不同的測試架構與切片數量的網路環境。本論文所提出的方法在各種不同的流量環境中均有良好的獎勵收益以及較低的壅塞水平，可明顯的比其他方法帶來更加高效且穩定的網路環境。

5.2 未來展望

考量未來 5G 網路將會有更多種且更複雜的網路流量與請求輸入，如何讓深度強化式學習的方法能夠更迅速且精準的對其進行處理，是一個遲早需要面對的問題，不管是對神經網路架構進行調整，或是讓輸入訊號進行 One-hot Encoding 來增強神經網路對參數的敏感度，都是可行且有效的方法。

此外在考量要做出拒絕特定流量時更需要注意關於網路阻斷率的問題，否則將會大大的降低整體網路環境的 QoS，使得網路環境不堪使用。

參考文獻

[1] M. R. Palattella, M. Dohler, L. A. Grieco, R. Rizzo, J. Torsner, T. Engel and L. Ladid, ‘‘Internet of Things in the 5G era: Enablers, architecture, and business models,’’ *IEEE Journal on Selected Areas in Communications.*, vol. 34, no. 1, pp. 510–527, Mar. 2016

[2] M. O. Ojijo and O. E. Falowo, ‘‘A Survey on Slice Admission Control Strategies and Optimization Schemes in 5G Network,’’ *IEEE Access*, vol. 8, pp. 14977-14990, 2020

[3] ITU-R, “M.2083 IMT Vision – “Framework and overall objectives of the future development of IMT for 2020 and beyond,” July 2015. [Online]. Available: http://www.itu.int/rec/R-REC-M/en.

[4] V. N. Ha, T, T, Nguyen, L. B. LE, and J. F. Frigon, “Admission Control and Network Slicing for Multi-Numerology 5G Wireless Networks,” *IEEE Networking Letters*, vol.2, pp. 5-9, 2020

[5] M. Z. Chowdhury, M. T. Hossan and Y. M. Jang, “Applying Model-Free Reinforcement Learning Algorithm in Network Slicing for 5G,” *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*, pp. 1-4, 2019

[6] Y. Abiko, D. Mochizuki, T. Saito, D. Ikeda, T. Mizuno and H. Mineno, “Proposal of Allocating Radio Resources to Multiple Slices in 5G using Deep Reinforcement Learning,” *2019 IEEE 8th Global Conference on Consumer Electronics (GCCE)*, pp. 1-2, 2019

[7] D. Zavyalova and V. Drozdova, “5G Scheduling using Reinforcement Learning,” *2020 International Multi-Conference on Industrial Engineering and Modern Technologies (FarEastCon)*, pp. 1-5, 2020

[8] Z. Xiong, Y. Zhang, D. Niyato, R. Deng, P. Wang and L. C. Wang, “Deep Reinforcement Learning for Mobile 5G and Beyond: Fundamentals, Applications, and Challenges,” *IEEE Vehicular Technology Magazine*, vol.14, pp. 44-52, 2019

[9] B. Guo, X. Zhang, Y. Wang and H. Yang, “Deep-Q-Network-Based Multimedia Multi-Service QoS Optimization for Mobile Edge Computing Systems,” *IEEE*

Access, vol.7, pp. 160961-160972, 2019

[10] R. S. Sutton and A. G. Barto, “Reinforcement Learning: An Introduction,” Cambridge, MA, USA:MIT Press, vol. 1, 2017.

[11] C. J. C. H. Watkins and P. Dayan, “Q-learning,” *Mach Learn*, vol. 8, pp. 279–292, 1992

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, pp. 529-533, 2015

[13] V. Londa and J. Tsitsiklis, “Actor-critic algorithms,” *Advances in Neural Information Processing Systems*, vol. 13, pp. 1008–1014, 1999.

[14] S. Fujimoto, H. V. Hoof and D. Meger, “Addressing function approximation error in actor-critic methods,” in *arXiv* [Online]. Available: https://arxiv.org/abs/1802.09477, 2018.

[15] H. V. Hasselt, “Double q-learning,” *Neural Information Processing Systems*, pp. 2613–2621, 2010.

[16] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” in *arXiv* [Online]. Available: https://arxiv.org/abs/1707.06347, 2017.

[17] T. Varum, A. Ramos and J. N. Matos, “Planar microstrip series-fed array for 5G applications with beamforming capabilities,” *2018 IEEE MTT-S International Microwave Workshop Series on 5G Hardware and System Technologies (IMWS-5G)*, pp. 1-3, 2018

[18] G. Wang, G. Feng, S. Qin, R. Wen and S. Sun, “Optimizing network slice dimensioning via resource pricing”, *IEEE Access*, vol. 7, pp. 30331-30343, 2019.

[19] X. Li, J. Rao, H. Zhang and A. Callard, “Network slicing with elastic SFC”, *IEEE 86th Veh. Technol (VTC-Fall)*, pp. 1-5, 2017.

[20] Y. L. Lee, J. Loo, T. C. Chuah and L.-C. Wang, “Dynamic network slicing for

multitenant heterogeneous cloud radio access networks", *IEEE Transactions on Wireless Communications*, vol. 17, pp. 2146-2161, 2018.

[21] H. Beigy and M.R. Meybody, "User-based Call Admission Control Policies for Cellular Mobile Systems: A Survey," *J CSI on Computer Science and Engineering*, vol.10, no.2(b), pp. 45-58, 2003.

[22] M. Ghaderi and R. Boutaba, "Call Admission Control in Mobile Cellular Networks: A Comprehensive Survey," *Wireless Communication and Mobile Computing*, vol.6, no.1, pp.69-93, 2006

[23] J. Hou and Y. Fang, "Mobility-based Call Admission Control Schemes for Wireless Mobile Networks," *Wireless Communication and Mobile Computing*, vol.1, pp.269-282, 2002.

[24] D. Mitra, M. I. Reinman and J. Wang, "Robust dynamic admission control for unified cell and call QoS in statistial multiplexers," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 5, pp. 692-707, 1998.

[25] H. Tong and T. X. Brown, "Adaptive call admission control under quality of service constraints: a reinforcement learning solution", *IEEE International Journal of Network Management*, vol. 18, 2002.

[26] S. Mignanti, A. D. Giorgio and V. Suraci, "A model based RL admission control algorithm for next generation networks," *The Second Intenational Conference on Next Generation Mobile Application Services and Technologies*, pp. 303-308, 2008.

[27] N. Lilith and K. Doganacay, "Reinforcement learning-Based dynamic guard channel scheme with maximum packing for cellular telecommunications systems," *3rd International Conference on Wireless Communication networking and Mobile Computing*, 2007.

[28] T. Haarnoja, A. Zhou, P. Abbeel and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” in *arXiv* [Online]. Available: https://arxiv.org/abs/1801.01290

[29] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, “Maximum entropy inverse reinforcement learning,” *In AAAI Conference on Artificial Intelligence*, pp. 1433–1438, 2008.

[30] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel and S. Levine, “Soft actor-critic algorithms and applications,” in *arXiv* [Online]. Available: https://arxiv.org/abs/1812.05905, 2018b.

[31] P. Christodoulou, “Soft Actor-Critic for Discrete Action Settings,” in *arXiv* [Online]. Available: https://arxiv.org/abs/1910.07207

[32] T. Schaul, J. Quan, I. Antonoglou, D. Silver, “Prioritized Experience Replay” in *arXiv* [Online]. Available: https://arxiv.org/abs/1511.05952

[33] R. Houthooft, R. Y. Chen, P. Isola, B. C. Stadie, F. Wolski, J. Ho, P. Abbeel, “Evolved Policy Gradients,” in *arXiv* [Online]. Available: https://arxiv.org/abs/1802.04821

[34] The 3GPP website. [Online]. Available: https://www.3gpp.org//

[35] A. k. Erlang (1878 - 1929). (n.d.). Plus Maths. Retrieved June 29, 2022, from https://plus.maths.org/content/os/issue2/erlang/index

[36] L. W. Dowdy, V. A. F. Almeida, D. A. Menasce. "Performance by Design: Computer Capacity Planning by Example". Archived from the original on 2016-05-06. Retrieved 2009-07-08.

[37] S. Kira. "Hershey Medical Center to open redesigned emergency room". The Patriot-News. Archived from the original on June 29, 2016. Retrieved March 12, 2009.

[38] D. G. Kendall, Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded Markov chain, Ann. Math. Stat. 1953

[39] S. Yoon, M. E. Lewis, “Optimal Pricing and Admission Control in a Queueing System with Periodically Varying Parameters.” Queueing Systems 47, 177–199 (2004).

[40] A. Karamouzian, E. Teimoury, and M. Modarres. “A model for admission control of returned products in a remanufacturing facility using queuing theory”. *Int J Adv Manuf Technol* **54,** 403–412 (2011).

[41] H. Klessig, A. Fehske and G. Fettweis, "Admission control in interference-coupled wireless data networks: A queuing theory-based network model," 2014 12th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt), 2014, pp. 151-158

附錄

表 3 環境 1-1 實驗結果數據

	Reward				Congestion			
λ	SAC RAC	DQN	Greedy	Fixed Slice	SAC RAC	DQN	Greedy	Fixed Slice
0.5	4.4	4.4	4.4	4.4	0%	0%	0%	0%
1	9.0	9.1	9.0	9.1	0%	0%	0%	0%
1.5	13.3	13.3	13.3	13.3	0%	0%	0%	0%
2	17.8	17.6	17.7	17.8	0%	0%	0%	0%
2.5	22.3	22.3	22.2	22.4	0%	0%	0%	0%
3	26.5	26.6	26.6	26.7	0%	0%	0%	0%
3.5	31.0	31.2	31.3	30.7	0%	0%	0%	1%
4	35.2	35.6	35.6	34.4	0%	0%	0%	2%
4.5	39.0	40.3	39.7	37.6	0%	0%	0%	3%
5	42.3	43.8	43.6	39.2	0%	0%	1%	6%
5.5	46.3	48.0	47.2	40.3	0%	1%	2%	9%
6	49.7	51.0	49.6	39.7	0%	2%	4%	13%
6.5	52.6	53.4	50.8	38.6	0%	3%	6%	17%
7	55.5	54.9	51.2	36.5	0%	5%	9%	21%
7.5	58.1	55.9	50.0	33.7	0%	7%	13%	25%
8	60.3	56.2	48.6	31.6	0%	9%	16%	28%
8.5	62.7	56.1	46.8	28.4	0%	11%	19%	31%

9	64.6	55.1	43.8	24.8	0%	13%	23%	35%
9.5	65.9	54.2	40.8	21.3	1%	15%	26%	37%
10	66.6	52.5	37.4	17.6	1%	18%	29%	40%

表 4 環境 2-1 實驗結果數據

	Reward				**Congestion**			
λ	**SAC RAC**	**DQN**	**Greedy**	**Fixed Slice**	**SAC RAC**	**DQN**	**Greedy**	**Fixed Slice**
0.5	7.4	7.4	7.4	7.4	0%	0%	0%	0%
1	14.8	14.6	14.8	14.8	0%	0%	0%	1%
1.5	22.1	21.4	22.4	19.9	0%	0%	0%	5%
2	28.8	27.9	29.5	22.3	0%	0%	0%	13%
2.5	34.9	34.7	36.5	20.5	0%	1%	1%	22%
3	39.6	40.4	41.4	16.9	1%	3%	4%	31%
3.5	42.3	42.5	42.8	11.2	2%	8%	9%	39%
4	43.0	41.3	41.0	5.5	5%	14%	15%	45%
4.5	41.9	37.5	37.1	-0.8	10%	21%	22%	51%
5	38.9	32.4	31.6	-7.5	14%	28%	29%	55%
5.5	34.9	26.0	25.6	-14.5	19%	34%	34%	59%
6	29.6	20.0	19.1	-21.0	23%	39%	39%	62%
6.5	24.1	13.1	12.7	-28.6	28%	43%	43%	65%
7	18.1	6.4	5.8	-35.5	32%	47%	47%	67%
7.5	11.9	-1.2	-1.5	-43.0	36%	50%	51%	69%

8	4.6	-8.0	-8.1	-49.9	39%	53%	53%	71%
8.5	-1.8	-14.7	-15.2	-56.9	42%	56%	56%	73%
9	-7.7	-21.3	-23.2	-64.3	45%	58%	59%	74%
9.5	-15.4	-28.8	-30.1	-71.9	48%	60%	61%	76%
10	-21.9	-36.2	-37.2	-79.1	50%	62%	63%	77%

表 5 環境 3-1 實驗結果數據

	Reward				**Congestion**			
λ	**SAC RAC**	**DQN**	**Greedy**	**Fixed Slice**	**SAC RAC**	**DQN**	**Greedy**	**Fixed Slice**
0.5	4.3	4.5	4.4	4.4	0%	0%	0%	0%
1	8.6	8.8	8.9	9.1	0%	0%	0%	0%
1.5	12.5	13.4	13.4	13.3	0%	0%	0%	0%
2	17.3	17.7	17.8	17.8	0%	0%	0%	0%
2.5	21.7	22.4	22.1	22.4	0%	0%	0%	0%
3	25.7	26.6	26.6	26.7	0%	0%	0%	0%
3.5	30.1	31.4	31.2	30.7	0%	0%	0%	1%
4	34.1	35.7	35.7	34.4	0%	0%	0%	2%
4.5	38.2	39.7	40.1	37.6	0%	0%	0%	3%
5	42.1	44.2	44.1	39.2	0%	0%	1%	6%
5.5	45.8	47.8	46.9	40.3	0%	1%	2%	9%
6	49.2	50.8	49.6	39.7	0%	2%	3%	13%
6.5	52.5	53.7	50.7	38.6	0%	3%	6%	17%

7	55.4	54.6	51.0	36.5	0%	5%	9%	21%
7.5	58.4	55.2	50.4	33.7	0%	7%	12%	25%
8	60.9	55.3	48.6	31.6	1%	10%	16%	28%
8.5	63.7	55.0	46.4	28.4	1%	12%	19%	31%
9	65.5	53.5	43.8	24.8	1%	15%	22%	35%
9.5	67.0	51.4	41.3	21.3	2%	18%	26%	37%
10	68.7	49.6	37.8	17.6	2%	20%	29%	40%

表 6 環境 1-2 實驗結果數據

	Reward				**Congestion**			
λ	**SAC RAC**	**DQN**	**Greedy**	**Fixed Slice**	**SAC RAC**	**DQN**	**Greedy**	**Fixed Slice**
0.5	4.4	4.4	4.4	4.4	0%	0%	0%	0%
1	8.8	9.0	9.0	8.9	0%	0%	0%	0%
1.5	13.2	13.4	13.3	13.2	0%	0%	0%	0%
2	17.7	17.9	17.8	17.8	0%	0%	0%	0%
2.5	22.2	22.3	22.2	22.3	0%	0%	0%	0%
3	26.7	26.6	26.9	26.7	0%	0%	0%	0%
3.5	31.1	30.9	31.1	31.1	0%	0%	0%	0%
4	35.3	35.9	35.9	35.7	0%	0%	0%	0%
4.5	40.2	40.1	40.3	39.9	0%	0%	0%	0%
5	44.6	44.4	44.3	44.2	0%	0%	0%	0%
5.5	48.8	49.0	49.1	48.1	0%	0%	0%	1%

6	53.4	53.6	53.4	51.8	0%	0%	0%	2%
6.5	57.7	57.7	57.9	54.2	0%	0%	0%	3%
7	61.7	62.0	61.6	56.4	1%	0%	1%	5%
7.5	65.0	64.9	65.4	57.4	1%	1%	1%	7%
8	67.9	68.0	67.9	57.5	2%	2%	2%	9%
8.5	69.5	69.6	70.0	57.1	4%	4%	4%	12%
9	70.5	70.5	70.2	55.5	6%	6%	6%	15%
9.5	71.0	70.2	70.9	53.0	8%	9%	8%	19%
10	69.8	69.6	69.9	50.2	11%	11%	11%	22%

表 7 環境 2-2 實驗結果數據

	Reward				**Congestion**			
λ	**SAC RAC**	**DQN**	**Greedy**	**Fixed Slice**	**SAC RAC**	**DQN**	**Greedy**	**Fixed Slice**
0.5	7.4	7.4	7.4	7.3	0%	0%	0%	0%
1	14.9	14.9	14.8	14.8	0%	0%	0%	0%
1.5	22.2	22.3	22.4	22.2	0%	0%	0%	0%
2	29.4	29.6	29.7	28.8	0%	0%	0%	2%
2.5	36.5	37.3	37.0	33.1	0%	0%	0%	6%
3	42.9	44.3	44.4	34.2	0%	0%	0%	11%
3.5	48.6	51.4	51.4	32.4	0%	1%	1%	19%
4	52.9	56.3	56.5	29.1	1%	2%	2%	26%
4.5	55.5	58.6	58.9	24.3	2%	6%	6%	32%

5	56.4	58.5	58.4	19.3	4%	11%	11%	37%
5.5	55.3	55.1	55.5	13.4	7%	16%	16%	42%
6	53.4	50.8	50.6	7.2	10%	21%	22%	46%
6.5	50.1	45.9	45.4	-0.1	14%	26%	26%	50%
7	45.1	39.6	39.3	-6.9	18%	31%	31%	53%
7.5	41.2	33.7	33.5	-13.2	20%	35%	35%	56%
8	35.3	26.6	27.0	-21.1	25%	39%	39%	59%
8.5	30.4	20.3	20.0	-27.5	27%	42%	42%	61%
9	24.0	13.0	12.4	-34.8	30%	45%	45%	63%
9.5	17.0	5.3	6.6	-41.1	33%	48%	48%	65%
10	10.8	-1.5	-1.6	-49.8	36%	50%	51%	67%

表 8 環境 3-2 實驗結果數據

	Reward				**Congestion**			
λ	**SAC RAC**	**DQN**	**Greedy**	**Fixed Slice**	**SAC RAC**	**DQN**	**Greedy**	**Fixed Slice**
0.5	4.4	4.5	4.4	4.4	0%	0%	0%	0%
1	8.8	9.0	8.8	9.1	0%	0%	0%	0%
1.5	13.3	13.5	13.3	13.3	0%	0%	0%	0%
2	17.8	17.9	17.7	17.8	0%	0%	0%	0%
2.5	21.9	22.2	22.3	22.4	0%	0%	0%	0%
3	26.6	26.6	26.7	26.7	0%	0%	0%	0%
3.5	31.4	31.0	31.1	30.7	0%	0%	0%	1%

4	35.4	35.3	35.4	34.4	0%	0%	0%	2%
4.5	40.1	40.0	39.9	37.6	0%	0%	0%	3%
5	44.3	44.7	44.7	39.2	1%	1%	1%	6%
5.5	48.9	48.7	48.8	40.3	2%	2%	2%	9%
6	52.8	53.4	53.4	39.7	3%	4%	3%	13%
6.5	56.9	57.7	57.8	38.6	5%	5%	5%	17%
7	61.1	62.2	61.7	36.5	8%	7%	7%	21%
7.5	64.9	66.1	65.4	33.7	9%	8%	8%	25%
8	68.3	69.0	67.9	31.6	10%	7%	7%	28%
8.5	71.6	72.0	69.7	28.4	11%	7%	7%	31%
9	74.5	74.0	70.9	24.8	11%	6%	6%	35%
9.5	77.9	75.6	70.7	21.3	11%	5%	4%	37%
10	80.0	76.6	70.0	17.6	10%	4%	4%	40%

表 9 環境 1-3 實驗結果數據

	Reward				**Congestion**			
λ	**SAC RAC**	**DQN**	**Greedy**	**Fixed Slice**	**SAC RAC**	**DQN**	**Greedy**	**Fixed Slice**
0.5	4.4	4.4	4.5	4.4	0%	0%	0%	0%
1	8.9	8.8	8.8	8.9	0%	0%	0%	0%
1.5	13.2	13.3	13.3	13.3	0%	0%	0%	0%
2	17.7	17.7	17.9	17.8	0%	0%	0%	0%
2.5	22.2	22.2	22.3	22.5	0%	0%	0%	0%

3	26.5	26.9	26.6	26.8	0%	0%	0%	0%
3.5	31.1	31.2	31.1	31.1	0%	0%	0%	0%
4	35.7	35.5	35.7	35.6	0%	0%	0%	0%
4.5	40.3	39.9	40.1	40.2	0%	0%	0%	0%
5	44.6	44.1	44.7	44.7	0%	0%	0%	0%
5.5	49.0	48.8	48.9	49.0	0%	0%	0%	0%
6	53.3	53.4	53.5	53.5	0%	0%	0%	0%
6.5	57.9	58.0	58.1	58.1	0%	0%	0%	0%
7	62.0	62.4	62.4	62.1	0%	0%	0%	0%
7.5	67.1	67.1	66.9	65.7	0%	0%	0%	1%
8	71.4	71.2	71.2	69.2	0%	0%	0%	2%
8.5	75.3	75.6	75.4	71.6	0%	0%	0%	3%
9	79.6	79.4	78.9	74.0	0%	0%	0%	4%
9.5	83.3	83.6	83.2	74.7	1%	1%	1%	6%
10	86.1	87.3	85.8	75.2	2%	1%	2%	8%

表 10 環境 2-3 實驗結果數據

	Reward				**Congestion**			
λ	**SAC RAC**	**DQN**	**Greedy**	**Fixed Slice**	**SAC RAC**	**DQN**	**Greedy**	**Fixed Slice**
0.5	7.4	7.5	7.4	7.4	0%	0%	0%	0%
1	14.8	14.9	14.9	14.7	0%	0%	0%	0%
1.5	22.2	22.3	22.2	22.1	0%	0%	0%	0%
2	30.0	29.9	29.6	29.7	0%	0%	0%	0%

2.5	37.2	37.3	36.8	36.4	0%	0%	0%	1%
3	44.4	44.4	44.4	42.0	0%	0%	0%	3%
3.5	52.0	51.7	51.7	45.4	0%	0%	0%	6%
4	59.4	58.8	59.2	46.1	0%	0%	0%	11%
4.5	66.4	66.1	66.2	44.5	0%	0%	0%	17%
5	72.1	72.0	71.8	41.6	2%	2%	1%	22%
5.5	75.1	74.8	74.9	37.0	4%	4%	4%	27%
6	75.6	75.5	75.5	31.7	8%	8%	8%	32%
6.5	73.5	73.6	73.5	26.6	12%	12%	12%	36%
7	70.0	69.9	69.9	19.5	16%	16%	16%	41%
7.5	64.9	64.7	65.2	13.3	21%	21%	21%	44%
8	59.2	60.0	59.7	6.3	25%	25%	25%	47%
8.5	53.7	54.3	53.9	-0.1	29%	29%	29%	50%
9	47.5	47.8	47.3	-7.0	32%	32%	32%	53%
9.5	41.1	40.8	40.7	-14.0	35%	36%	36%	55%
10	34.1	33.8	33.9	-20.8	38%	39%	38%	57%

表 11 環境 3-3 實驗結果數據

	Reward				**Congestion**			
λ	**SAC RAC**	**DQN**	**Greedy**	**Fixed Slice**	**SAC RAC**	**DQN**	**Greedy**	**Fixed Slice**
0.5	4.5	4.4	4.4	4.4	0%	0%	0%	0%
1	9.0	8.9	8.9	9.1	0%	0%	0%	0%

1.5	13.2	13.5	13.3	13.3	0%	0%	0%	0%
2	17.9	17.8	17.8	17.8	0%	0%	0%	0%
2.5	22.1	22.1	22.4	22.4	0%	0%	0%	0%
3	26.8	26.8	26.8	26.7	0%	0%	0%	0%
3.5	30.9	31.3	31.2	30.7	0%	0%	0%	1%
4	35.6	35.6	35.6	34.4	0%	0%	0%	2%
4.5	40.1	39.9	40.0	37.6	0%	0%	0%	3%
5	44.7	44.8	44.4	39.2	1%	1%	1%	6%
5.5	49.1	49.0	48.9	40.3	2%	2%	2%	9%
6	53.4	53.1	53.2	39.7	3%	3%	3%	13%
6.5	57.9	58.1	58.2	38.6	5%	5%	5%	17%
7	61.9	61.7	62.3	36.5	7%	6%	7%	21%
7.5	66.3	66.9	66.8	33.7	8%	7%	7%	25%
8	70.6	70.9	71.5	31.6	7%	7%	7%	28%
8.5	74.6	75.3	75.2	28.4	6%	6%	6%	31%
9	78.0	80.0	79.1	24.8	6%	4%	5%	35%
9.5	81.9	83.2	82.9	21.3	4%	3%	3%	37%
10	85.6	86.5	85.7	17.6	3%	2%	2%	40%

表 12 環境 4-1 實驗結果數據

	Reward				Congestion			
λ	SAC RAC	DQN	Greedy	Fixed Slice	SAC RAC	DQN	Greedy	Fixed Slice
0.5	7.5	7.5	7.3	7.4	0%	0%	0%	0%
1	14.9	14.7	14.9	14.8	0%	0%	0%	1%
1.5	21.5	22.3	22.5	19.9	0%	0%	0%	5%
2	27.2	29.4	29.6	22.3	0%	0%	0%	13%
2.5	32.6	36.5	36.3	20.5	0%	1%	1%	22%
3	36.9	41.3	41.3	16.9	0%	3%	4%	31%
3.5	40.9	42.7	42.6	11.2	0%	9%	9%	39%
4	43.7	41.0	40.9	5.5	0%	15%	16%	45%
4.5	45.9	36.9	36.4	-0.8	1%	22%	23%	51%
5	47.0	31.9	31.5	-7.5	2%	28%	29%	55%
5.5	46.4	25.6	25.5	-14.5	3%	34%	34%	59%
6	45.8	19.6	19.6	-21.0	4%	39%	39%	62%
6.5	43.3	12.8	12.5	-28.6	6%	43%	43%	65%
7	39.8	5.5	5.9	-35.5	8%	47%	47%	67%
7.5	36.3	-1.8	-1.2	-43.0	10%	51%	51%	69%
8	32.5	-8.1	-7.8	-49.9	12%	53%	53%	71%
8.5	28.2	-15.5	-15.3	-56.9	13%	56%	56%	73%
9	23.2	-22.5	-22.6	-64.3	15%	58%	58%	74%
9.5	18.0	-30.3	-30.3	-71.9	17%	61%	61%	76%

10	12.8	-37.2	-37.0	-79.1	19%	62%	62%	77%

表 13 環境 4-2 實驗結果數據

	Reward				**Congestion**			
λ	**SAC RAC**	**DQN**	**Greedy**	**Fixed Slice**	**SAC RAC**	**DQN**	**Greedy**	**Fixed Slice**
0.5	7.5	7.5	7.4	7.5	0%	0%	0%	0%
1	14.8	14.8	14.9	14.9	0%	0%	0%	0%
1.5	22.3	22.0	22.3	22.2	0%	0%	0%	0%
2	29.5	29.9	29.8	28.5	0%	0%	0%	2%
2.5	36.3	37.0	37.1	33.0	0%	0%	0%	5%
3	42.2	44.5	44.8	34.1	0%	0%	0%	11%
3.5	47.3	51.3	51.1	32.5	0%	1%	1%	19%
4	51.2	56.5	56.7	28.9	0%	2%	2%	26%
4.5	54.4	59.1	58.6	24.4	0%	6%	6%	32%
5	57.3	58.5	58.4	18.4	0%	11%	11%	38%
5.5	58.8	54.9	55.4	12.6	0%	16%	16%	42%
6	60.3	50.8	51.0	6.4	0%	21%	21%	46%
6.5	61.2	45.7	45.4	0.0	0%	26%	26%	50%
7	61.1	39.4	40.0	-7.4	0%	31%	31%	53%
7.5	60.7	32.9	33.7	-14.5	0%	35%	35%	56%
8	60.0	26.6	26.8	-21.2	0%	39%	39%	59%
8.5	57.9	19.8	20.2	-27.4	1%	42%	42%	61%

9	56.6	12.2	12.4	-34.6	1%	45%	45%	63%
9.5	54.1	6.3	5.8	-41.7	1%	48%	48%	65%
10	52.2	-1.2	-1.3	-49.8	1%	50%	50%	67%

表 14 環境 4-3 實驗結果數據

	Reward				**Congestion**			
λ	**SAC RAC**	**DQN**	**Greedy**	**Fixed Slice**	**SAC RAC**	**DQN**	**Greedy**	**Fixed Slice**
0.5	7.4	7.4	7.4	7.3	0%	0%	0%	0%
1	14.9	14.8	14.8	14.9	0%	0%	0%	0%
1.5	22.2	22.2	22.2	22.1	0%	0%	0%	0%
2	29.5	29.6	29.7	29.7	0%	0%	0%	0%
2.5	37.1	37.0	37.1	36.3	0%	0%	0%	1%
3	44.3	44.2	44.4	42.1	0%	0%	0%	3%
3.5	51.7	51.8	52.0	45.2	0%	0%	0%	6%
4	58.8	59.7	59.2	46.2	0%	0%	0%	11%
4.5	65.5	66.0	66.3	44.7	0%	1%	0%	17%
5	71.1	72.0	71.7	41.7	0%	2%	0%	22%
5.5	74.3	75.1	74.7	36.9	0%	4%	0%	27%
6	76.0	75.3	75.7	32.1	0%	8%	0%	32%
6.5	75.6	73.8	73.7	25.8	0%	12%	0%	37%
7	73.7	69.7	70.4	19.9	0%	16%	1%	40%
7.5	71.2	65.2	65.7	13.0	0%	21%	1%	44%

8	68.5	59.4	59.5	6.5	0%	25%	2%	47%
8.5	64.6	53.4	53.7	-0.5	0%	29%	4%	50%
9	60.3	47.4	46.8	-6.8	0%	32%	6%	53%
9.5	56.0	40.8	40.2	-13.9	0%	36%	8%	55%
10	51.5	34.7	34.0	-20.0	0%	38%	11%	57%

表 15 環境 4-4 實驗結果數據

	Reward				**Congestion**			
λ	**SAC RAC**	**DQN**	**Greedy**	**Fixed Slice**	**SAC RAC**	**DQN**	**Greedy**	**Fixed Slice**
0.5	7.5	7.4	7.4	7.4	0%	0%	0%	0%
1	14.6	14.8	14.7	14.8	0%	0%	0%	0%
1.5	22.3	22.1	22.3	22.4	0%	0%	0%	0%
2	29.6	29.8	29.5	29.7	0%	0%	0%	0%
2.5	37.1	37.2	37.1	36.9	0%	0%	0%	0%
3	44.3	44.3	44.4	44.3	0%	0%	0%	0%
3.5	51.9	52.1	52.2	50.9	0%	0%	0%	1%
4	59.0	59.4	59.1	55.5	0%	0%	0%	3%
4.5	66.5	67.2	66.6	58.7	0%	0%	0%	6%
5	73.4	73.8	74.6	59.8	0%	0%	0%	10%
5.5	80.5	81.0	81.4	58.9	0%	0%	0%	14%
6	86.1	86.9	86.8	55.6	0%	1%	1%	19%
6.5	90.9	91.0	90.6	50.7	0%	3%	3%	24%

7	94.2	92.3	92.6	46.0	0%	6%	5%	28%
7.5	95.7	91.5	91.8	40.2	1%	9%	9%	32%
8	96.8	88.8	88.7	34.9	1%	13%	13%	35%
8.5	96.3	84.3	84.5	28.9	2%	16%	17%	39%
9	94.9	79.4	79.2	21.7	3%	20%	20%	42%
9.5	93.9	73.9	73.3	15.9	4%	24%	24%	44%
10	92.4	67.6	67.6	9.7	4%	27%	27%	47%