# 3.1 問題定義

在現代自動化倉儲的機器人移動履行系統（RMFS）中，隨著訂單量的增長與機器人部署密度的提高，交通壅塞（traffic congestion）已成為制約整體運作效率的核心瓶頸。大量的機器人在有限的通道內執行任務，尤其在路網的交叉點（intersection），極易因路權協調不當而產生衝突與等待，從而引發一系列連鎖負面效應，包括：訂單履行時間延遲、機器人無效等待造成的能源浪費、以及系統整體吞吐量的下降。

因此，本研究的核心問題在於：**如何設計一套有效且能自我適應的交通控制策略，以動態調節路口的通行權，從而最小化機器人的等待時間與整體能耗，最終提升整個倉儲系統的運作效率？**

為回答此問題，必須先建構一個能夠準確模擬上述挑戰的實驗平台。本章節所闡述的實驗環境與系統架構，便是為此一核心問題所量身打造的高擬真度測試平台（high-fidelity testbed），其目的在於為後續章節中不同控制演算法的實現、訓練與比較，提供一個穩定、可控且可量化的基礎。

# 3.1.1 本章節結構

本節旨在全面性地介紹支持本研究的實驗平台。內容將涵蓋四個核心部分：首先，詳細闡述所建構的倉儲模擬環境，包括其物理佈局與核心實體（3.2.1節）；其次，說明為實現演算法評估而設計的交通控制系統架構（3.2.2節）；再者，條列實驗所使用的具體硬體與軟體配置（3.2.3節）；最後，定義用以衡量各控制策略成效的量化評估指標（3.2.4節）。透過本節的闡述，讀者將能完整理解實驗進行的基礎環境與評估標準。

# 3.2 實驗環境與系統架構

為有效驗證並比較本研究所提出的神經演化強化學習（NERL）交通控制策略與其他基線方法，必須建構一個能夠準確反映真實世界倉儲運作挑戰的高擬真度模擬平台。此平台不僅是演算法的測試場域，更是確保所有比較皆在公平、可控且可量化基礎上進行的先決條件。

本節將全面闡述此一實驗平台的構成。內容依序分為四個核心部分： - **3.2.1 倉儲模擬環境設計**：詳細介紹模擬世界的物理佈局、路網結構，以及包含機器人、貨架、工作站等在內的核心實體元件。 - **3.2.2 交通控制系統架構**：深入解析為實現演算法整合與評估所設計的軟體架構，說明系統如何驅動決策、執行控制與觸發學習。 - **3.2.3 實驗硬體與軟體配置**：條列進行所有模擬與訓練實驗所使用的具體硬體規格與軟體函式庫。 - **3.2.4 效能評估指標定義**：明確定義用以衡量不同控制策略成效的關鍵量化指標（KPIs），作為後續章節實驗結果分析的依據。

透過本節的闡述，讀者將能完整理解本研究進行實驗的基礎環境與評估標準。

# 3.2.1 倉儲模擬環境設計

為對交通控制策略進行有效評估，本研究首先建構了一個高擬真度的倉儲模擬環境。此環境不僅定義了物理空間的佈局，還包含了多種動態實體及其互動規則，共同構成了一個複雜的機器人移動履行系統（RMFS）。本節將詳細闡述其設計。

### 1. 物理環境與佈局

模擬倉儲建立在一個二維的離散化網格之上，每個網格單元均有其特定功能。整體佈局採用了功能分區的設計，以確保運作流程的有序性。

* **中央儲存區**: 位於倉庫中心，由大量**儲位 (Pod Location)** 密集排列而成。此區域的通道被設計為嚴格的**單向通道 (One-way Aisles)**，水平與垂直通道的流動方向交錯排列，此設計從物理層面極大地簡化了交通管理的複雜性，旨在減少機器人對向行駛時的潛在衝突。
* **工作站區**: 分佈在倉庫的兩側。一側為**揀貨站 (Picking Station)**，是訂單履行的出口；另一側為**補貨站 (Replenishment Station)**，是貨物進入系統的入口。
* **充電站 (Charging Station)**: 散佈在儲存區內，由部分儲位轉化而來，供機器人自主充電。

**【圖表建議：圖 3.2.1 - 倉儲佈局示意圖】** 為直觀展示佈局，建議此處插入一張示意圖，用不同顏色標示出儲存區、揀貨區、補貨區與充電站，並用箭頭清晰標示出單向通道的流動方向。

### 2. 核心實體與生命週期

系統的動態行為由多種核心實體之間的互動所驅動。

* **機器人 (Robot)**: 作為系統中最核心的活動單元，機器人具備一套複雜的狀態機來管理其工作流程，包括 閒置 (idle)、前往貨架 (taking\_pod)、運送貨架 (delivering\_pod)、在站處理 (station\_processing) 及 返回貨架 (returning\_pod) 等狀態。本研究為機器人建立了精細的物理與能量模型，其能耗計算不僅考慮了負載，還涵蓋了啟動成本與再生制動（剎車能量回收），為能源效率評估提供了堅實基礎。此外，機器人具備基於優先級的自主避障邏輯，能夠在一定程度上自主解決局部衝突。

**【圖表建議：圖 3.2.2 - 機器人狀態轉換圖】** 為清晰展示機器人的工作流程，建議此處插入一張 UML 狀態機圖，描繪其核心狀態以及觸發狀態轉換的事件（如「分配新任務」、「到達工作站」等）。

* **貨架 (Pod)**: 是儲存貨物 (SKU) 的移動載體。每個貨架可存放多種 SKU，並記錄了每種 SKU 的當前數量與補貨閾值。當存貨量低於閾值時，系統將自動觸發對應的補貨任務。
* **工作站 (Station)**: 是人機協作的節點。當機器人將貨架運送至工作站後，系統會模擬工人的揀貨或補貨延遲。為應對高流量，工作站還設計了動態路徑調整機制，當站內機器人過多時會啟用備用長路徑以緩解擁塞。

### 3. 訂單與任務流程

模擬的驅動力來源於訂單。一個**訂單 (Order)** 代表一份客戶需求，包含多種需要揀選的 SKU。系統會將訂單分解為一個或多個**任務 (Job)**。一個任務的核心是「將指定的貨架運送到指定的工作站」，這是可直接分配給機器的最小工作單元。整個流程如下： 1. 系統接收訂單。 2. 訂單所需 SKU 被定位到特定的貨架上。 3. 系統生成一個或多個任務，並將其放入任務佇列。 4. 閒置的機器人從佇列中領取任務，開始其取貨、送貨、返程的工作生命週期。 5. 當一個訂單所需的所有 SKU 都被成功送達工作站後，該訂單被標記為完成。

# 3.2.2 交通控制系統架構

為實現不同交通控制演算法的彈性整合與公平比較，本研究設計了一套基於**策略模式 (Strategy Pattern)** 與**工廠模式 (Factory Pattern)** 的模組化軟體架構。此架構的核心在於將「決策演算法」與「系統執行框架」進行分離，確保無論是簡單的規則式邏輯還是複雜的深度強化學習模型，都能在相同的基礎上運作與評估。系統主要由以下三個組件構成：

### 1. TrafficController (交通控制器抽象基類)

此抽象類（Abstract Base Class, ABC）定義了所有交通控制策略必須遵循的統一介面。其最核心的方法為 get\_direction(intersection, tick, warehouse)，該方法接收路口當前的詳細狀態（局部資訊）以及整個倉儲系統的狀態（全域資訊），並回傳該路口的通行決策（例如 "Horizontal" 或 "Vertical"）。透過強制所有控制器實作此介面，系統確保了對不同演算法呼叫方式的一致性。此外，基類中也整合了標準化的統計數據收集功能，用以記錄各類效能指標。

### 2. TrafficControllerFactory (控制器工廠)

此類別採用工廠設計模式，負責根據外部設定（如實驗組態檔中指定的控制器類型）動態創建對應的 TrafficController 子類別實例。當模擬核心需要一個控制器時，僅需提供一個如 "dqn"、"nerl" 或 "time\_based" 的字串識別碼，工廠即可回傳一個對應的、已初始化的控制器物件。此設計將控制器的「創建邏輯」與「使用邏輯」完全解耦，大幅提升了實驗流程的靈活性與可擴展性，使得切換不同的控制策略無需修改任何核心模擬程式碼。

### 3. IntersectionManager (路口管理器)

路口管理器是整個交通控制系統的中央協調者與執行引擎，其運作流程構成了一個完整的閉環控制系統： 1. **持有控制器實例**：在模擬初始化階段，IntersectionManager 會透過控制器工廠獲取當前實驗所需的控制器實例。 2. **驅動決策迴圈**：在模擬的每個時間單位 (tick)，管理器會遍歷倉儲中的所有路口。 3. **獲取決策**：對於每一個路口，它會呼叫 TrafficController 實例的 get\_direction() 方法來獲取該路口的通行指令。 4. **執行決策**：根據控制器回傳的指令，IntersectionManager 會更新路口的內部狀態，例如改變允許通行的方向。 5. **觸發模型訓練**：特別地，對於強化學習類型的控制器（DQN/NERL），在決策與執行步驟完成後，IntersectionManager 還會接著呼叫其 train() 方法，將剛剛發生的狀態轉換（State-Action-Reward-NextState）提供給模型，使其能夠從經驗中學習和優化。

**【圖表建議：圖 3.2.1 - 交通控制系統運作序列圖】**

為使讀者能更直觀地理解此運作流程，強烈建議在此處插入一張 **UML 序列圖 (Sequence Diagram)**。該圖應清晰地展示從模擬器主迴圈 (Simulation Loop) 觸發，到 IntersectionManager 遍歷路口，再到 TrafficController 進行決策 (get\_direction)，最後由 IntersectionManager 更新路口狀態 (updateAllowedDirection) 並觸發學習 (train) 的完整訊息傳遞順序。

# 3.2.3 實驗硬體與軟體配置

為確保本研究的可重複性與結果的有效性，所有模擬、訓練與評估實驗均在明確定義的硬體平台與標準化的軟體環境下進行。本節將詳細條列相關配置。

### 硬體配置

本研究的計算任務主要分為兩部分：在雲端高效能平台上進行模型訓練，以及在本機端進行開發、調試與結果分析。

#### 訓練環境 (Runpod Secure Cloud)

所有計算密集型的模型訓練任務均於 Runpod 雲端平台上執行，以利用其強大的計算資源，加速學習過程。 - **CPU**: 42 vCPU - **GPU**: 1 x NVIDIA GeForce RTX 4090 - **記憶體 (RAM)**: 83 GB

#### 開發與分析環境 (使用者筆記型電腦)

程式開發、初步測試、參數調校及最終的數據分析與視覺化則在本地端電腦完成。 - **CPU**: AMD Ryzen 9 6900HX - **GPU**: NVIDIA GeForce RTX 3080 Ti - **記憶體 (RAM)**: 16.0 GB

### 軟體配置

軟體環境的選擇旨在兼顧開發效率、計算效能與社群支援的廣泛性。 - **作業系統**: - **訓練環境**: Ubuntu 22.04 (於 runpod/pytorch:2.1.0-py3.10-cuda11.8.0-devel Docker 容器中) - **開發環境**: Microsoft Windows 11 - **程式語言**: Python 3.10 - **核心計算函式庫**: - PyTorch (版本 2.1.0): 作為核心深度學習框架，用於建構與訓練 DQN 及 NERL 神經網路模型。 - numpy (版本 1.24.4): 為所有數值計算提供基礎，廣泛應用於狀態表示、獎勵計算等。 - pandas (版本 2.0.3): 主要用於實驗數據的處理、儲存與分析。 - **模擬與分析工具**: - networkx (版本 3.2.1): 用於建構與分析倉儲的路網圖結構。 - scikit-learn (版本 1.5.2): 在本研究中主要用於數據正規化等預處理步驟。 - **視覺化函式庫**: - matplotlib (版本 3.7.2): 用於生成靜態的二維圖表，如折線圖、柱狀圖等。 - seaborn (版本 0.13.2): 基於 matplotlib，提供更美觀、更統計學導向的視覺化圖表，如熱力圖。 - **系統工具**: - psutil (版本 5.9.8): 用於監控系統資源使用情況。

**【表格建議：表 3.2.2 - 軟體函式庫與用途】**

為使軟體配置更加一目了然，建議在此處插入一個表格。該表格應包含三欄：「函式庫」、「版本號」與「在研究中的主要用途」，將上述所有函式庫整理進去。

# 3.2.4 效能評估指標定義

為客觀、量化地評估不同交通控制策略的優劣，本研究建立了一套綜合性關鍵績效指標（KPIs）。為清楚定義，我們首先約定以下數學符號： - $ R $: 倉儲中所有機器人的集合。 - $ O\_{completed} $: 在模擬期間內所有已完成訂單的集合。 - $ P $: 所有機器人通過路口的事件（passing event）的集合。 - $ T\_{sim} $: 總模擬時長（ticks）。

### 1. 效率指標 (Efficiency Metrics)

**總能量消耗 (Total Energy Consumption)** 此指標衡量系統整體的能源使用效率，是本研究的核心優化目標之一。其計算方式為模擬期間所有機器人活動所消耗的能量總和。

其中 $ E\_r $ 代表單一機器人 $ r $ 在整個模擬過程中的總能耗。

**平均機器人利用率 (Average Robot Utilization)** 此指標反映了機器人族群的整體繁忙程度。其定義為所有機器人處於非閒置狀態的時間佔總模擬時間的平均百分比。

其中 $ |R| $ 是機器人總數，$ t\_{active}(r) $ 是機器人 $ r $ 的總活動時間。

### 2. 流量指標 (Throughput Metrics)

**訂單完成總數 (Completed Orders Count)** 此指標直接衡量系統在固定時間內的總產出，反映了整體的運作效率。

**平均訂單處理時間 (Average Order Processing Time)** 此指標衡量系統處理單一訂單的響應速度，其定義為所有已完成訂單從開始處理到結束的平均耗時。

其中 $ t\_{complete}(o) $ 和 $ t\_{start}(o) $ 分別是訂單 $ o $ 的完成時間與開始時間。

**平均交叉口等待時間 (Average Intersection Waiting Time)** 此指標直接反映交通控制策略的協調效率。它計算的是每一次機器人通過路口事件中，等待時間的平均值。

其中 $ |P| $ 是機器人通過路口的總次數，$ t\_{wait}(p) $ 是單次通過事件 $ p $ 的等待時間。

### 3. 穩定性指標 (Stability Metrics)

**總停止-啟動次數 (Total Stop-and-Go)** 此指標反映了交通流的平順程度。頻繁的啟停不僅消耗額外能量，也代表交通流不穩定。

其中 $ N\_{s-g}(r) $ 是機器人 $ r $ 在路口因等待而啟停的總次數。

# 3.3 基線控制器設計

為客觀且嚴謹地評估本研究所提出的深度強化學習交通控制方法（詳見第 3.4 節）的效能，必須將其與一組具有代表性且易於理解的基線控制器（Baseline Controllers）進行比較。基線控制器提供了一個效能參考點，使我們能夠量化複雜演算法所帶來的實際改進程度。一個理想的基線應當反映業界現行的或直觀的解決方案。

本研究選用兩種不同但具代表性的邏輯來設計基線控制器：一種是完全不考慮實時交通狀況的固定時制控制器，另一種是根據路口即時需求進行反應的動態控制器。這兩者分別代表了靜態與動態控制策略的基礎形式，能夠全面地衡量強化學習模型在不同交通情境下的適應性與優越性。

本章節將依序詳細闡述以下兩種基線控制器的內部設計原理、決策邏輯與關鍵參數：

1. **時間基礎控制器 (Time-Based Controller)**：一種基於固定時間週期切換路權的靜態控制器。
2. **佇列基礎控制器 (Queue-Based Controller)**：一種根據路口等待隊列的長度與任務優先級進行決策的動態反應式控制器。

# 3.3.1 時間基礎控制器

時間基礎控制器（TimeBasedController）是最基礎的靜態交通控制策略。其核心思想源於傳統的都市交通號誌系統，完全不考慮路口的實時交通流量或任何動態變化，僅依賴一個預先設定的、固定的時間週期來循環切換水平與垂直方向的通行權。這種方法的優點在於其極致的簡單性與可預測性，但缺點也同樣明顯——無法適應交通需求的波動，容易在交通繁忙時造成不必要的等待，或在交通稀少時浪費通行時間。

### 設計原理與決策邏輯

此控制器的運作完全由三個參數決定：水平方向綠燈時間（(T\_{H\_green})）、垂直方向綠燈時間（(T\_{V\_green})），以及由兩者加總構成的完整訊號週期長度（(T\_{cycle})）。

[ T\_{cycle} = T\_{H\_green} + T\_{V\_green} ]

在模擬運行的任何一個時間刻（tick），控制器會透過取餘運算來判斷當前處於訊號週期中的哪個時間點（(t\_{pos})），其計算方式如下：

[ t\_{pos} = T\_{cycle} ]

根據 (t\_{pos}) 的值，控制器作出通行方向的決策。如果 (t\_{pos}) 小於水平方向的綠燈時間 (T\_{H\_green})，則給予水平方向通行權；反之，則給予垂直方向通行權。決策規則可表示為：

[ =

]

在我們的倉儲環境中，由於貨架（Pod）主要沿著水平方向排列，機器人在水平方向的移動頻率與數量遠高於垂直方向。為了配合此一特性，在參數設定上，我們給予水平方向更長的綠燈時間（例如，(T\_{H\_green} = 70) ticks），而垂直方向的綠燈時間則相對較短（例如，(T\_{V\_green} = 30) ticks），以期在不考慮即時狀態下，達到一個初步的流量平衡。

**【圖表建議：圖 3.3.1 - 時間基礎控制器訊號週期示意圖】**

建議在此處插入一張時間軸圖，清晰地展示 (T\_{cycle}) 的構成，並標示出 (T\_{H\_green}) 和 (T\_{V\_green}) 的區間，以及在不同區間內對應的通行方向決策（Horizontal/Vertical）。

# 3.3.2 佇列基礎控制器

佇列基礎控制器（QueueBasedController）是一種動態反應式控制策略，其設計目的是為了解決時間基礎控制器無法感知即時交通需求的根本性缺陷。此控制器會持續監測路口兩個方向的等待隊列，並結合機器人當前所執行任務的緊急程度，動態地計算出通行權的歸屬。相較於靜態的時間基礎控制器，佇列基礎控制器能夠更靈活地適應交通流量的變化，將通行資源優先分配給需求更迫切的方向。

### 設計原理與決策邏輯

此控制器的核心在於將兩個關鍵因素量化並結合，以進行決策：**機器人數量**與**任務優先級**。

#### 1. 任務優先級權重系統

在倉儲作業中，並非所有機器人任務都具有相同的重要性。例如，正在將貨架送往揀貨站的機器人若被延誤，會直接影響訂單的履行效率；而一台空車前往貨架區的機器人，其任務的緊急程度則相對較低。為了體現此差異，我們為不同的機器人狀態（robot.current\_state）定義了一套優先級權重（(W\_{priority})）：

| 機器人狀態 (current\_state) | 任務描述 | 優先級權重 ((W\_{priority})) |
| --- | --- | --- |
| delivering\_pod | 運送貨架至揀貨站 | 3.0 |
| returning\_pod | 將貨架送回儲存區 | 2.0 |
| taking\_pod | 前往儲存區領取貨架 | 1.0 |
| idle | 閒置或待命中 | 0.5 |
| station\_processing | 於揀貨站內處理中 | 0.0 |

#### 2. 方向優先級計算

對於路口的每一個方向（水平 H 或垂直 V），控制器會計算一個加權的優先級總和（(P\_{H}) 或 (P\_{V})）。此數值等於該方向上所有等待機器人（(R\_{dir})）各自的任務優先級權重之和。計算公式如下：

[ P\_{dir} = *{i R*{dir}} W\_{priority}(i) ]

其中，(W\_{priority}(i)) 代表機器人 (i) 當前狀態所對應的優先級權重。

此外，考慮到倉儲佈局導致的水平方向先天交通流量較大的特性，我們引入了一個偏好因子（(*{bias})，bias\_factor），對水平方向的優先級總和進行加權，以給予其額外的競爭優勢。因此，最終用於比較的水平方向優先級 (P’*{H}) 為：

[ P’*{H} = P*{H} \_{bias} ]

#### 3. 決策流程

控制器的決策流程如下： 1. **最小綠燈時間檢查**: 為避免因交通狀況快速變化而導致路口訊號頻繁切換（這會造成機器人反覆加減速，浪費能源），控制器會先檢查自上次方向切換以來，是否已達到一個最小綠燈時間（(T\_{min\_green})）。若否，則維持當前方向不變。 2. **優先級比較**: 若已滿足最小綠燈時間，控制器會計算加權後的水平優先級 (P’*{H}) 與垂直優先級 (P*{V})。 3. **特殊情況處理**: - 如果任一方向沒有等待的機器人，則立即將通行權給予有機器人的另一方向。 - 如果兩方向皆無機器人，則維持當前狀態。 4. **最終決策**: 在一般情況下，控制器會比較 (P’*{H}) 和 (P*{V})，將通行權分配給優先級總和較高的方向。

[ =

]

透過此一機制，佇列基礎控制器能夠在兼顧穩定性（最小綠燈時間）與佈局特性（偏好因子）的同時，對即時的交通需求做出合理且有效率的反應。

# 3.3.3 基線控制器參數設定

為了確保實驗的有效性和可重複性，本研究對兩種基線控制器所使用的參數進行了明確的定義和標準化。這些參數是在初步實驗中根據經驗選擇的，旨在讓控制器在通用場景下表現出合理且穩定的性能。

### 1. 時間基礎控制器 (TimeBasedController)

此控制器的邏輯完全由固定的時間週期驅動，其參數設定如下：

| 參數名稱 | 預設值 | 單位 | 說明 |
| --- | --- | --- | --- |
| horizontal\_green\_time | 70 | ticks | 水平方向的綠燈持續時間。由於倉儲佈局，水平幹道承擔了更重的東西向交通流，因此給予更長的通行時間。 |
| vertical\_green\_time | 30 | ticks | 垂直方向的綠燈持續時間。 |
| **週期總長度** | **100** | **ticks** | **一個完整的信號週期 (70 + 30)。** |

### 2. 佇列基礎控制器 (QueueBasedController)

此控制器根據實時的交通狀況進行決策，其參數涉及到了決策的靈敏度和對不同任務的偏好。

| 參數名稱 | 預設值 | 單位/類型 | 說明 |
| --- | --- | --- | --- |
| min\_green\_time | 1 | ticks | 最小綠燈時間。設定一個非常小的值，是為了防止信號在兩個有衝突的請求之間過於頻繁地「振盪」，但仍然保持了控制器對交通變化的快速反應能力。 |
| bias\_factor | 1.5 | float (乘數) | 水平方向偏好因子。此乘數會被應用於水平方向的加權佇列計算結果上，以補償水平幹道天然的較高交通流量，避免垂直方向因少量高優先級機器人而過於頻繁地搶佔通行權。 |
| priority\_weights | {"delivering\_pod": 3.0, "returning\_pod": 2.0, "taking\_pod": 1.0, "idle": 0.5} | Dictionary | 任務優先級權重。此字典定義了處於不同任務狀態的機器人的重要性。例如，一個正在運送貨物以完成訂單 (delivering\_pod) 的機器人，其權重是閒置 (idle) 機器人的 6 倍。 |

# 3.4 DRL 問題構成要素

在設計一個深度強化學習智能體之前，必須嚴格定義其與環境互動的邊界和目標。這一步驟稱為強化學習問題的公式化 (Formulation)，主要包含三個核心要素的定義：狀態空間 (State Space)、動作空間 (Action Space) 和獎勵函數 (Reward Function)。

* **狀態空間** (S) 定義了智能體能從環境中「看到」的所有資訊。
* **動作空間** (A) 定義了智能體能「做出」的所有決策。
* **獎勵函數** (R) 定義了智能體的「目標」，它將特定的狀態或（狀態-動作）對映為一個純量訊號，指導智能體的學習方向。

本章節將詳細闡述這三個構成要素在本研究中的具體設計。

# 3.4.1 狀態空間 (State Space)

狀態空間的設計是強化學習成功的關鍵，它必須為智能體提供足夠的、有意義的資訊，以做出有效決策，同時又要避免維度災難。在本研究的 RMFS 交通控制問題中，狀態被定義為一個能全面反映交叉路口局部交通狀況、並兼顧下游路口潛在影響的特徵向量。

每個交叉路口控制器的狀態 s 是一個 **17 維的連續向量**，其構成如下：

1. **局部狀態 (Local State) - 8 維**: 描述控制器所在交叉路口自身的即時交通資訊。
   * **水平方向 (Horizontal)**:
     + [0] **佇列長度 (Queue Length)**: 水平幹道上等待通過路口的機器人數量。
     + [1] **首車等待時間 (First Vehicle Waiting Time)**: 如果佇列非空，此為佇列中第一個機器人已等待的時間；否則為 0。
     + [2] **平均等待時間 (Average Waiting Time)**: 水平方向所有等待機器人的平均等待時間。
     + [3] **下游飽和度 (Downstream Saturation)**: 水平方向下一個路口的佇列長度，用以預測潛在的回堵 (spillback) 風險。
   * **垂直方向 (Vertical)**:
     + [4]-[7] 對應垂直幹道的上述四個特徵。
2. **鄰居狀態 (Neighbor State) - 8 維**: 描述與當前路口直接相連的四個鄰居路口的關鍵資訊，幫助智能體理解更廣泛的交通態勢。
   * **上、下、左、右四個鄰居**各包含 2 維資訊：
     + [8, 9] **上方鄰居**: 垂直佇列長度、水平佇列長度。
     + [10, 11] **下方鄰居**: 垂直佇列長度、水平佇列長度。
     + [12, 13] **左側鄰居**: 垂直佇列長度、水平佇列長度。
     + [14, 15] **右側鄰居**: 垂直佇列長度、水平佇列長度。
3. **全局狀態 (Global State) - 1 維**: 引入一個宏觀指標，幫助智能體將局部決策與整體系統目標對齊。
   * [16] **揀貨站平均佇列 (Average Picking Station Queue)**: 所有揀貨站入口處的平均排隊長度。這是一個關鍵指標，因為揀貨站是整個系統的最終瓶頸。

### 狀態歸一化

由於上述 17 個特徵的單位和數值範圍各不相同（例如，計數、時間、飽和度），直接輸入神經網路會導致訓練不穩定。因此，在將狀態向量饋入 DRL 模型之前，本研究採用了一種**自適應歸一化 (Adaptive Normalization)** 技術。該技術會動態追蹤每個狀態特徵的運行均值和標準差，並將其標準化為均值為 0、標準差為 1 的分佈，確保了模型訓練的穩定性和效率。

# 3.4.2 動作空間 (Action Space)

與複雜的狀態空間相對，本研究中的動作空間設計得非常簡潔直觀。對於每一個由 DRL 控制的交叉路口，智能體（控制器）可以在每個決策時刻從一個包含 **6 個離散動作**的集合中選擇一個來執行。這些動作不僅包括了基本的交通相位切換，還引入了動態調整局部速度限制的能力。

### 離散動作集合

動作集合 A 被定義為： A = {0, 1, 2, 3, 4, 5}

這六個數字分別對應以下控制策略：

#### 基礎相位控制 (Basic Phase Control)

* **動作 0: KEEP\_CURRENT\_PHASE (保持當前相位)**
  + **描述**: 維持交叉路口當前的號誌狀態不變。
  + **目的**: 當現有車流順暢，或轉換相位的成本高於潛在收益時，此為最優選擇。
* **動作 1: SWITCH\_TO\_VERTICAL\_GREEN (切換為垂直綠燈)**
  + **描述**: 將號誌相位強制切換為垂直方向綠燈，水平方向紅燈。
  + **目的**: 當狀態顯示垂直方向的交通壓力大於水平方向時，應執行此動作以疏通垂直車流。
* **動作 2: SWITCH\_TO\_HORIZONTAL\_GREEN (切換為水平綠燈)**
  + **描述**: 將號誌相位強制切換為水平方向綠燈，垂直方向紅燈。
  + **目的**: 當狀態顯示水平方向的交通壓力大於垂直方向時，應執行此動作以疏通水平車流。

#### 動態速度控制 (Dynamic Speed Control)

為了更精細地管理交通流，避免在瓶頸路口發生回堵，模型還可以執行以下動作來動態調整離開該交叉口的機器人的速度限制。這些動作本身不改變號誌相位。

* **動作 3: SET\_SPEED\_NORMAL (設置速度為正常)**
  + **描述**: 將從此路口出發的機器人速度限制恢復為預設值 (1.0)。
  + **目的**: 在交通狀況緩解後，取消速度限制。
* **動作 4: SET\_SPEED\_SLOW (設置速度為慢速)**
  + **描述**: 將從此路口出發的機器人速度限制降低為慢速 (0.5)。
  + **目的**: 在預測到下游可能發生擁堵時，主動減緩車流，平滑交通波動。
* **動作 5: SET\_SPEED\_VERY\_SLOW (設置速度為極慢速)**
  + **描述**: 將從此路口出發的機器人速度限制降低為極慢速 (0.2)。
  + **目的**: 在下游發生嚴重擁堵時，最大程度地減少進入擁堵區域的流量，為其疏通提供時間。

### 決策間隔

控制器每隔 T\_decision = 10 個時間步才會根據當前狀態評估並選擇一個新動作。在這個間隔期內，交叉路口將維持上一個決策所設定的號誌相位與速度限制。

# 3.4.3 獎勵函數 (Reward Function)

獎勵函數是連接智能體動作與學習目標的橋樑，它通過一個純量回饋信號來告知智能體其決策的優劣。在本研究中，為了探索不同學習信號對模型性能的影響，我們設計並實現了兩種截然不同的獎勵模式，分別對應不同的實驗組別。這兩種模式的設計旨在平衡即時回饋的指導性與最終目標的全局性。

### 1. 步階獎勵 (Step Reward)

步階獎勵模式旨在為智能體提供一個密集的、即時的回饋信號。在每個決策間隔（10 ticks）結束時，系統會根據這段時間內交叉路口的局部觀測指標，計算一個綜合獎勵值。這種高頻率的回饋有助於智能體快速學習到基礎的交通控制啟發式規則。

步階獎勵 R\_step 由以下四個加權分量構成：

* **流量獎勵 (Flow Reward)**: 正向獎勵。獎勵智能體在決策間隔內成功引導通過交叉路口的機器人數量。
  + R\_flow = w\_flow \* (N\_horizontal + N\_vertical)
  + 其中 N 是通過的機器人數量，w\_flow 是權重。
* **能源懲罰 (Energy Penalty)**: 負向獎勵（懲罰）。懲罰因等待和啟停造成的額外能源消耗。
  + R\_energy = w\_energy \* E\_consumed
  + 其中 E\_consumed 是估算的能源消耗，w\_energy 是權重。
* **等待時間懲罰 (Waiting Time Penalty)**: 負向獎勵。懲罰所有仍在佇列中等待的機器人的累計等待時間。
  + R\_wait = w\_wait \* T\_cumulative\_wait
  + 其中 T\_cumulative\_wait 是累計等待時間，w\_wait 是權重。
* **相位切換懲罰 (Switching Penalty)**: 負向獎勵。對每一次號誌相位的切換施加一個固定的懲罰，以避免過於頻繁、無效的相位切換，鼓勵控制器維持暢通的交通流。
  + R\_switch = w\_switch (僅在發生切換時觸發)

**關鍵路口加權**: 為了讓智能體優先學習如何管理靠近揀貨站等瓶頸區域的交叉路口，對於這些被標記為「關鍵」的交叉路口，其計算出的 R\_step 會被乘以一個大於 1 的放大係數 w\_critical。

### 2. 全域獎勵 (Global Reward)

與步階獎勵不同，全域獎勵模式提供一個稀疏的、延遲的回饋信號。在這種模式下，智能體在整個評估回合（evaluation episode，例如 3000 或 8000 ticks）中不會收到任何獎勵信號。只有在回合結束時，系統才會根據整個倉儲的最終宏觀性能指標，計算一個單一的獎勵值。

這種模式的目的是迫使智能體學習一系列能夠對系統長期、全局目標產生積極影響的複雜行為，而非僅僅優化局部指標。

全域獎勵 R\_global 由以下三個核心指標的加權組合而成：

* **訂單完成效率 (Order Completion Score)**: 主要的正向獎勵。基於在規定時間內完成的訂單數量。
  + S\_order = w\_order \* N\_completed\_orders
* **總體時間成本 (Total Time Cost)**: 負向獎勵。懲罰所有訂單從生成到完成所花費的總時間。
  + C\_time = w\_time \* T\_total\_order\_time
* **總體能源成本 (Total Energy Cost)**: 負向獎勵。懲罰整個系統在運行期間的總估算能耗。
  + C\_energy = w\_energy\_global \* E\_total
* **回堵懲罰 (Spillback Penalty)**: 一個巨大的負向懲罰。如果在評估期間發生了導致系統死鎖的嚴重回堵，則施加此懲罰，以確保智能體學會避免災難性策略。
  + P\_spillback = w\_spillback (僅在發生嚴重回堵時觸發)

最終的全域獎勵為：R\_global = S\_order - C\_time - C\_energy - P\_spillback。

# 3.4 深度強化學習方法

為了應對第 3.1 節所定義的動態交通控制挑戰，並超越第 3.3 節基線控制器在適應性上的限制，本研究導入了深度強化學習（Deep Reinforcement Learning, DRL）作為核心解決方案。DRL 結合了深度學習在高維數據中提取特徵的強大能力與強化學習在序列決策問題中通過試錯進行學習的優勢，使其特別適合解決如倉儲交通控制這類具有複雜、非線性狀態空間的難題。

與依賴固定規則的基線控制器不同，DRL 智慧體能夠從與環境的互動中自主學習控制策略，發現人類設計師難以預見的有效模式。它們的目標是最大化一個長期累積的獎勵訊號，從而學習到能夠平衡即時需求與長遠系統效率的複雜行為。

本研究將實現並深入比較兩種代表性的 DRL 方法，它們在學習機制上存在本質差異，分別代表了基於價值函數梯度下降和基於族群演化優化的兩大技術路線：

1. **深度 Q 網路 (Deep Q-Network, DQN)**：一種基於價值的（Value-based）典型 DRL 演算法，它使用神經網路來近似最優的動作價值函數（Action-Value Function, Q-function），並通過時間差分學習（Temporal Difference Learning）進行更新。
2. **神經演化強化學習 (Neuroevolution Reinforcement Learning, NERL)**：一種結合了神經網路與演化演算法（Evolutionary Algorithms）的黑箱優化方法。它不依賴梯度下降，而是將神經網路的權重視為基因，通過選擇、交叉和變異等演化操作，在一個族群中直接對策略進行搜索和優化。

接下來的章節將分別對這兩種方法的架構設計、獎勵函數以及訓練配置進行詳細闡述。

# 3.4.1 深度 Q 網路 (DQN) 架構設計

深度 Q 網路（DQN）是本研究採用的第一種 DRL 方法。它是一種基於價值的演算法，其核心目標是學習一個稱為「Q 函數」的評估模型。Q 函數 的作用是預測在給定狀態 下，執行動作 所能帶來的長期累積獎勵。通過學習一個準確的 Q 函數，智慧體便可以在任何狀態下，通過選擇具有最高 Q 值的動作來做出最優決策。

本研究實現的 DQNController 在經典 DQN 的基礎上進行了擴展，整合了經驗回放（Experience Replay）、目標網路（Target Network）等關鍵技術以提高學習的穩定性，並引入了一套混合式決策框架，將傳統規則與神經網路模型相結合。

### 1. 核心組件

DQNController 由以下幾個核心組件構成： - **策略網路 (Policy Network)**：一個深度神經網路，用於近似 Q 函數。它的輸入是狀態 ，輸出是該狀態下每個可能動作的 Q 值。此網路的權重會在訓練過程中通過梯度下降不斷更新。 - **目標網路 (Target Network)**：一個與策略網路結構完全相同但參數不同步更新的神經網路。在計算目標 Q 值時，使用目標網路可以打破數據間的自相關性，有效防止學習過程中的震盪與發散。目標網路的權重會定期從策略網路複製而來，而非每個時間步都更新。 - **經驗回放記憶體 (Replay Memory)**：一個固定容量的佇列，用於存儲智慧體與環境互動的經驗元組 ，其中 是執行動作 後的新狀態， 是獲得的即時獎勵。在訓練時，控制器會從記憶體中隨機抽樣一個小批次（minibatch）的經驗進行學習，這打破了經驗之間的時間相關性，使訓練數據更符合獨立同分布的假設，從而大幅提升了學習的穩定性與效率。 - **混合決策邏輯**：除了神經網路，DQNController 還內建了一套與 QueueBasedController 類似的防鎖死與優先級規則。在做出最終決策前，系統會先檢查是否存在諸如機器人等待時間超限、潛在死鎖等緊急情況。若觸發這些規則，則直接執行預設的避險操作；否則，才將決策權交給 DQN 模型。這種設計確保了系統在學習初期或面對極端情況時，依然能保持最基本的運作穩定性。

### 2. 狀態表示 (State Representation)

為了讓模型能夠做出明智的決策，必須為其提供一個資訊豐富的狀態向量。本研究設計了一個 17 維的狀態向量，全面地描述了一個路口的局部與周邊交通態勢： - **路口自身狀態 (8 維)**：包括當前允許方向、信號持續時間、各方向排隊的機器人總數、高優先級任務機器人數量、以及各方向的平均等待時間。 - **相鄰路口狀態 (8 維)**：包括周圍有效鄰居的數量、鄰居路口的機器人總數、高優先級任務總數、平均等待時間、以及方向分佈等，為模型提供了更廣闊的視野。 - **全域系統狀態 (1 維)**：引入下游關鍵瓶頸——揀貨站的總排隊長度。這一全域資訊使得路口控制器能夠感知到下游的擁塞情況，從而做出更具預見性的決策，避免向上游傳播擁堵。

在將這些原始特徵輸入網路之前，我們會使用一個自適應正規化器（AdaptiveNormalizer）對其進行標準化處理，以確保不同尺度特徵的均衡性，加速模型的收斂。

### 3. 動作空間 (Action Space)

控制器在每個決策點可以選擇的動作集合。為簡化問題，我們定義了 3 個離散動作： - **動作 0**: 保持當前方向不變。 - **動作 1**: 將通行權切換至水平方向。 - **動作 2**: 將通行權切換至垂直方向。

### 4. 神經網路架構

策略網路與目標網路均採用一個結構相同的前饋神經網路（Feed-Forward Neural Network）。該網路包含兩個隱藏層，具體架構如下：

**輸入層 (17) -> 全連接層 (128) -> ReLU 激活 -> 全連接層 (64) -> ReLU 激活 -> 輸出層 (3)**

這個相對精簡的架構在表達能力與計算效率之間取得了良好的平衡，足以處理我們定義的狀態-動作空間。

**【圖表建議：圖 3.4.1 - DQN 控制器架構與決策流程圖】**

建議在此處繪製一張流程圖，清晰地展示 DQNController 的完整決策流程。圖中應包含以下步驟： 1. 接收到路口狀態 s。 2. 進入混合決策模塊，檢查防鎖死規則。 3. 若觸發規則，則直接輸出確定性動作。 4. 若未觸發，則將狀態 s 傳遞給策略網路。 5. 策略網路輸出各動作的 Q 值。 6. 通過 Epsilon-Greedy 策略選擇最終動作 a。 7. 同時展示經驗 被存入經驗回放記憶體的過程。 8. 獨立地展示從記憶體中採樣進行網路訓練的迴圈。

# 3.4.2 神經演化強化學習 (NERL) 架構設計

神經演化強化學習（NERL）是本研究採用的第二種 DRL 方法，它代表了一種與 DQN 截然不同的優化範式。NERL 不依賴於梯度下降來優化網路參數，而是將神經網路的權重和偏置視為個體的「基因」，並應用演化演算法（Evolutionary Algorithms, EA）直接在參數空間中進行搜索。這種方法屬於黑箱優化，它不關心獎勵函數是否可微，僅根據每個策略（即每個神經網路個體）在與環境互動後獲得的最終適應度分數（Fitness Score）來進行優化。

### 1. 核心思想與運作流程

NERL 的核心在於維護一個由多個神經網路組成的**族群 (Population)**。其基本運作流程遵循「評估 -> 選擇 -> 繁殖」的演化循環： 1. **評估 (Evaluation)**：族群中的每一個神經網路個體都會獨立地在模擬環境中運行一個完整的評估回合（Episode）。在此期間，該個體全權負責所有路口的交通決策。回合結束後，系統會根據其宏觀表現（如 3.4.3 節定義的全局獎勵）計算出一個單一的適應度分數。 2. **選擇 (Selection)**：根據所有個體獲得的適應度分數，演算法會從當前族群中選擇表現優異的個體作為「父代」(Parents) 參與繁殖。本研究採用**錦標賽選擇 (Tournament Selection)** 機制，即隨機選取 k 個個體進行小組競賽，其中適應度最高的個體勝出並被選為父代。這種方法在選擇壓力和族群多樣性之間提供了良好的平衡。 3. **繁殖 (Reproduction)**：通過對選出的父代進行**交叉 (Crossover)** 與**變異 (Mutation)** 操作來生成新一代的「子代」(Offspring)，以填充下一個族群。 - **精英保留 (Elitism)**：為了確保最優的基因能夠傳承下去，每一代中適應度最高的少數個體（稱為「精英」）會被直接、完整地複製到下一代族群中。 - **交叉**：模擬生物繁殖過程，將兩個父代網路的權重向量進行混合，以產生新的子代網路。本研究採用**均勻交叉 (Uniform Crossover)**，即對權重向量中的每一位，以 50% 的機率隨機選擇繼承自父代一或父代二。 - **變異**：在交叉之後，對子代網路的權重進行微小的隨機擾動。本研究採用**高斯變異 (Gaussian Mutation)**，即以一定的變異機率 (mutation\_rate) 選定部分權重，並為其疊加一個從平均值為 0、標準差為 mutation\_strength 的高斯分布中採樣的噪聲。變異是維持族群多樣性、防止早熟並探索新解的關鍵。

通過不斷重複上述循環，整個族群的平均適應度會逐步提升，最終收斂到能夠高效解決問題的策略網路上。

### 2. NERL 與 DQN 的關鍵區別

* **優化機制**: DQN 使用梯度下降和反向傳播，依賴於可微的損失函數；NERL 使用演化算法，是一種無梯度的黑箱優化方法，對獎勵函數的連續性和可微性沒有要求。
* **探索機制**: DQN 通過 ε-greedy 等策略在動作空間進行探索；NERL 的探索則是在策略的**參數空間**中通過交叉和變異來實現。
* **獎勵訊號**: DQN 對即時、頻繁的獎勵訊號（Step Reward）更敏感；而 NERL 的架構天然地適合處理延遲、稀疏的全局獎勵（Global Reward），因為它評估的是整個回合的最終表現。
* **並行性**: NERL 的評估過程具有極高的並行性，族群中的每個個體都可以被分配到獨立的處理器核心上同時進行評估，使其能夠有效利用多核心 CPU 或計算集群。

### 3. 架構細節

NEController 的狀態表示、動作空間以及底層的神經網路結構與 DQNController 完全相同，以確保兩種方法之間比較的一致性與公平性。兩者的根本差異僅在於驅動網路權重更新的學習演算法不同。

**【圖表建議：圖 3.4.2 - NERL 演化循環示意圖】**

建議在此處繪製一張循環圖，說明 NERL 的核心運作流程。圖中應包含： 1. **族群 (Population)**：展示多個神經網路個體。 2. **並行評估 (Parallel Evaluation)**：從族群中取出每個個體，在獨立的環境中運行並計算其適應度分數。 3. **選擇 (Selection)**：根據適應度分數，通過錦標賽選擇出父代。 4. **繁殖 (Reproduction)**：展示精英保留、交叉和變異操作，生成新一代族群。 5. 箭頭應清晰地指向下一個階段，形成一個完整的閉環。

# 3.4.3 獎勵函數設計

獎勵函數 (Reward Function) 是強化學習的核心，它向智慧體 (Agent) 提供回饋訊號，引導其學習出期望的行為。本研究設計了兩種截然不同的獎勵模式：**即時獎勵 (Step Reward)** 和 **全局獎勵 (Global Reward)**，以從不同維度探索交通控制策略的優化路徑。

### 1. 即時獎勵 (Step Reward)

此模式旨在為控制器在每個時間步 (tick) 的決策提供立即的回饋。其核心思想是基於**局部可觀測事件**來獎勵促進交通流動的行為，並懲罰導致延遲或頻繁變換的行為。獎勵訊號 是一個複合函數，其核心組成部分是旨在鼓勵交通流動的**流通獎勵 ()** 和**能源獎勵 ()**，以及旨在抑制低效行為的**等待成本 ()** 和**切換成本 ()**。具體而言，當機器人成功通過路口時，系統會根據其任務優先級給予相應的流通獎勵；若其能以低速、低能耗的方式通過，則會獲得額外的能源獎勵。相對地，在路口等待的機器人會產生等待成本，而路口信號燈的切換則會觸發一次性的切換成本，以維持系統穩定性。

此外，為了引導智慧體關注系統瓶頸，我們對**關鍵路口**（如通往主幹道的入口）的所有獎懲訊號乘以一個更高的權重 。綜合以上各點，在路口 的單個時間步 的即時獎勵 可以概念化地表示為：

其中 是路口 的權重（關鍵路口為 ，否則為 1）， 是一個訊號放大係數，而 和 分別是該時間步在路口 通過和等待的機器人集合。

### 2. 全局獎勵 (Global Reward)

與即時獎勵不同，全局獎勵模式在整個評估回合 (episode) 結束時，才根據**最終的宏觀系統表現**給予一個單一的、延遲的獎勵分數。這種模式迫使智慧體學習行為與其長期後果之間的關聯。全局獎勵 的設計旨在最大化一個**綜合效率分數**，其核心是一個成本效益分析。該分數的分子部分代表完成訂單所帶來的效益，由訂單總數 與單筆訂單的獎勵權重 的乘積構成。分母部分則代表總成本，由正規化後的總能源消耗 、時間成本 以及因下游堵塞造成的溢出懲罰 加總而成。此外，若整個回合未發生溢出，系統還會給予一筆額外獎勵 。完整的公式如下：

這個公式清晰地引導智慧體在**完成盡可能多訂單**的同時，必須努力**降低能源和時間的總成本**，並**避免造成系統瓶頸**。其中 是一個極小值，用以防止分母為零。

# 3.4.4 訓練參數與超參數設定

為了確保實驗的可複現性與結果的有效性，本節將詳細列出在訓練 DQN 與 NERL 控制器時所使用的全部關鍵參數與超參數。這些參數的設定綜合考量了演算法的穩定性、收斂速度以及計算資源的限制，並在初步實驗中進行了調校。

### 1. 通用參數

以下為 DQN 與 NERL 兩種控制器共用的參數：

| 參數名稱 | 程式碼變數 | 預設值 | 說明 |
| --- | --- | --- | --- |
| 最小綠燈時間 | min\_green\_time | 1 | 為避免訊號頻繁切換的最小持續時間 (tick) |
| 方向偏好因子 | bias\_factor | 1.5 | 給予水平方向的額外優先級權重 |
| 狀態空間維度 | state\_size | 17 | 輸入神經網路的狀態向量維度 |
| 動作空間維度 | action\_size | 3 | 神經網路輸出的可能動作數量 (保持/水平/垂直) |
| 最大等待時間閾值 | max\_wait\_threshold | 50 | 觸發防鎖死機制的機器人最大等待時間 (tick) |

### 2. DQN 特有超參數

以下為 DQNController 在訓練過程中使用的主要超參數：

| 參數名稱 | 程式碼變數 | 預設值 | 說明 |
| --- | --- | --- | --- |
| 學習率 | learning\_rate | 5e-4 | Adam 優化器的學習率 |
| 折扣因子 (Gamma) | gamma | 0.95 | 未來獎勵的折扣係數，值越接近1代表越重視長期回報 |
| 初始探索率 (Epsilon) | epsilon | 1.0 | 訓練初期隨機選擇動作的機率 |
| 最小探索率 | epsilon\_min | 0.01 | 探索率衰減的下限 |
| 探索率衰減率 | epsilon\_decay | 0.999 | 每個訓練步驟後，探索率乘以的衰減係數 |
| 經驗回放記憶體大小 | memory\_size | 100,000 | 存儲經驗元組的最大數量 |
| 批次大小 | batch\_size | 8,192 | 每次從記憶體中抽樣進行訓練的樣本數量 |
| 目標網路更新頻率 | (硬編碼) | 1,000 ticks | 每隔多少時間步將策略網路的權重複製到目標網路 |

### 3. NERL 特有超參數

以下為 NEController 在演化過程中使用的主要超參數：

| 參數名稱 | 程式碼變數 | 預設值 | 說明 |
| --- | --- | --- | --- |
| 族群大小 | population\_size | 20 | 每一代中包含的個體（神經網路）數量 |
| 精英保留比例 | elite\_ratio | 0.2 | 每一代中適應度最高的個體被直接保留到下一代的比例 |
| 錦標賽選擇大小 | tournament\_size | 4 | 在錦標賽選擇中，每次隨機挑選進行比較的個體數量 |
| 交叉率 | crossover\_rate | 0.8 | 兩個父代進行交叉操作產生子代的機率 |
| 變異率 | mutation\_rate | 0.2 | 個體基因（網路權重）發生變異的基礎機率 |
| 變異強度 | mutation\_strength | 0.15 | 高斯變異的標準差，控制變異的幅度大小 |
| 評估間隔 | evolution\_interval | 1,000 ticks | 每個個體進行評估的持續時間 (tick) |

這些參數共同定義了兩種 DRL 方法的學習行為。在第 4 章的實驗評估中，我們將基於這些設定來對控制器進行訓練和比較。

# 3.5 實驗設計與評估方法

為了客觀、量化地回答本研究的核心問題——即我們提出的深度強化學習交通控制策略相較於傳統方法，在提升倉儲系統運作效率方面是否具有顯著優勢——本章節將詳細闡述整體的實驗設計、模型訓練流程、效能評估框架以及結果的統計分析方法。

一個嚴謹的實驗設計是確保研究結論可靠性的基石。為此，我們將建立一個涵蓋多種控制策略的對照實驗矩陣，並在統一的模擬環境與系統負載下對其進行測試。所有實驗都將遵循標準化的訓練與評估流程，以消除無關變數的干擾，確保不同演算法之間的比較是公平且有意義的。

本章節的結構如下： - **3.5.1 訓練配置與實驗組別**：定義實驗中所包含的所有控制器類型（實驗組），並說明用於訓練的軟硬體環境配置。 - **3.5.2 模型訓練流程**：詳細描述 DRL 模型（DQN 與 NERL）的具體訓練步驟，確保實驗的可複現性。 - **3.5.3 評估方法與比較框架**：闡述在模型訓練完成後，用於最終效能評估的標準化流程，以及橫向比較所有實驗組的統一框架。 - **3.5.4 統計分析與結果驗證**：說明將用於分析實驗數據的統計學方法，以科學地驗證不同策略之間效能差異的顯著性。

# 3.5.1 訓練配置與實驗組別

為系統性地評估不同交通控制策略的效能，本研究設計了一套包含十二個獨立實驗組的對照實驗矩陣。此設計旨在將本研究提出的兩種深度強化學習方法（DQN 和 NERL），在不同配置下的表現，與兩種基線控制器進行全面的比較。

### 實驗組別定義

所有實驗組將在完全相同的模擬環境（如 3.2.1 節所述）下運行，唯一的變數為交叉路口所採用的交通控制器及其配置。實驗組別定義如下表所示。

**表 3.5.1：實驗組別定義與說明**

| 組別 | 控制器類型 | 獎勵模式 | NERL 變體 | NERL 評估時長 | 類別 | 說明 |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | TimeBased | - | - | - | 基線 | 固定時間週期的靜態控制器 |
| 2 | QueueBased | - | - | - | 基線 | 基於即時隊列長度的動態反應式控制器 |
| 3 | DQN | step | - | - | DRL | 使用即時獎勵訓練的深度 Q 網路 |
| 4 | DQN | global | - | - | DRL | 使用全局獎勵訓練的深度 Q 網路 |
| 5 | NERL | step | **A (探索型)** | 3,000 | DRL | 高變異率，鼓勵探索新策略 |
| 6 | NERL | global | **A (探索型)** | 3,000 | DRL | 高變異率，鼓勵探索新策略 |
| 7 | NERL | step | **B (利用型)** | 3,000 | DRL | 低變異率，精細優化現有策略 |
| 8 | NERL | global | **B (利用型)** | 3,000 | DRL | 低變異率，精細優化現有策略 |
| 9 | NERL | step | **A (探索型)** | 8,000 | DRL | 高變異率，延長評估時間 |
| 10 | NERL | global | **A (探索型)** | 8,000 | DRL | 高變異率，延長評估時間 |
| 11 | NERL | step | **B (利用型)** | 8,000 | DRL | 低變異率，延長評估時間 |
| 12 | NERL | global | **B (利用型)** | 8,000 | DRL | 低變異率，延長評估時間 |

### NERL 變體參數詳解

為探究演化過程中「探索」（Exploration）與「利用」（Exploitation）之間的平衡對最終策略效能的影響，本研究設計了兩種具有不同演化超參數的 NERL 變體： - **變體 A (探索型)**：此配置具有更高的變異率（mutation\_rate=0.3）和變異強度（mutation\_strength=0.2）。較大的變異幅度和頻率使得族群能夠在參數空間中進行更廣泛的搜索，有潛力發現更具創新性的解決方案，但可能面臨收斂較慢的風險。 - **變體 B (利用型)**：此配置採用較低的變異率（mutation\_rate=0.1）和變異強度（mutation\_strength=0.05）。較小的變異使得演化過程更傾向於在已發現的較優解附近進行精細微調，有助於策略的穩定收斂，但可能陷入局部最優。

此外，為研究個體評估時間的充分性對學習效果的影響，每個 NERL 變體都將分別在 3,000 ticks 和 8,000 ticks 兩種評估時長（eval\_ticks）下進行訓練。

### 硬體與軟體配置

為確保實驗結果的一致性與可複現性，所有實驗均在標準化的環境下執行。詳細的硬體與軟體堆疊資訊已在 3.2.3 節 中進行了闡述。總結而言，計算密集型的模型訓練與評估主要在配備 NVIDIA GeForce RTX 4090 GPU 的雲端伺服器上進行，所有實驗均使用 Python 3.10 和 PyTorch 2.1.0 作為核心開發與執行環境。

# 3.5.2 模型訓練流程

為確保 DRL 模型能夠充分學習並收斂到一個較優的策略，同時保證不同模型之間比較的公平性，本研究設計了一套標準化的模型訓練流程。此流程詳細規定了從模型初始化到最終模型保存的每一個步驟。

### 1. DQN 訓練流程 (對應組別 3, 4)

DQN 的訓練是一個線上（On-policy）持續學習的過程。單次完整的 DQN 訓練實驗流程如下：

1. **初始化**：
   1. 根據 3.4.4 節中定義的超參數及實驗組別指定的獎勵模式（step 或 global），創建一個 DQNController 實例。
   2. 創建一個模擬倉庫環境 Warehouse 的實例。
2. **訓練迴圈**：
   1. 啟動一個持續 N = 200,000 個時間步（ticks）的模擬。
   2. 在每一個時間步 t，IntersectionManager 會遍歷所有交叉路口。
   3. 對於每一個交叉路口 i：
      1. 控制器從環境中獲取當前狀態 s\_t。
      2. 使用策略網路和 ε-greedy 策略選擇一個動作 a\_t。
      3. 執行動作 a\_t，環境轉移到下一個狀態 s\_{t+1}，並由 UnifiedRewardSystem 計算出即時獎勵 r\_t (在 global 模式下此獎勵為 0)。
      4. 將經驗元組 (s\_t, a\_t, r\_t, s\_{t+1}) 存入經驗回放記憶體。
   4. **經驗回放**：每隔 k=32 個時間步，從記憶體中隨機抽樣一個批次（batch）的經驗進行學習。
   5. **目標網路更新**：每隔 M=1,000 個時間步，將策略網路的權重複製到目標網路。
3. **模型保存**：訓練完全結束後，將最終的策略網路權重保存為最終模型。

### 2. NERL 訓練流程 (對應組別 5-12)

NERL 的訓練是一個世代迭代的離線（Off-policy）學習過程。其核心流程對所有 NERL 組別是統一的，但會根據具體組別的配置，代入不同的超參數。

1. **初始化**：
   1. 根據 3.4.4 節以及**具體實驗組別**（5-12）的定義，創建一個 NEController 實例。此步驟將確定以下關鍵超參數：
      * **獎勵模式**: step 或 global。
      * **變異配置 (Variant)**: **A (探索型)** 或 **B (利用型)**，這將決定 mutation\_rate 和 mutation\_strength 的值。
      * **評估時長 (Eval Ticks)**: 3,000 或 8,000。
   2. 控制器隨機初始化一個包含 20 個網路個體的族群。
2. **演化迴圈**：
   1. 啟動一個持續 G = 30 個世代（Generations）的演化。
   2. 在每一個世代 g：
      1. **並行評估 (Parallel Evaluation)**：為族群中的 20 個個體，分別啟動 20 個獨立、並行的模擬環境。
      2. 每個個體 j 在其專屬的環境中運行一個完整的評估回合。回合的持續時間由該實驗組的 eval\_ticks 參數決定（3,000 或 8,000 ticks）。
      3. 回合結束後，UnifiedRewardSystem 根據該實驗組指定的獎勵模式（step 或 global）計算出個體 j 的適應度分數 f\_j。
      4. **演化操作**：當所有個體的適應度分數都計算完畢後，控制器根據該實驗組的變異配置（A 或 B）執行一次完整的演化操作（選擇、交叉、變異），生成一個全新的子代族群。
      5. 新的子代族群將成為下一個世代 g+1 的起始族群。
3. **模型保存**：
   1. 在每一個世代結束時，演算法都會將該世代中適應度最高的個體保存為當代最佳模型。
   2. 在所有 30 個世代的演化完全結束後，從所有世代的最佳模型中，選出那個擁有歷史最高適應度分數的模型，將其作為該實驗組的最終模型。

# 3.5.3 評估方法與比較框架

在 DRL 模型根據 3.5.2 節所述流程完成訓練後，為確保所有控制器在一個公平且無偏的基礎上進行比較，本研究設計了一套標準化的評估流程。此流程旨在模擬一個固定的、可重複的「測試日」場景，並通過一組預先定義的關鍵績效指標（KPIs）來量化各控制策略的表現。

### 1. 標準化評估流程

對於 3.5.1 節中定義的每一個實驗組（包括兩個基線控制器和四個已訓練的 DRL 控制器），我們將執行以下標準化評估程序：

1. **模型加載**：對於 DRL 實驗組，加載其對應的最終訓練模型，並將其設定為純粹的**評估模式**（Inference Mode）。在此模式下，DQN 的 ε-greedy 探索將被關閉（ε=0），而 NERL 將固定使用其歷史上適應度最高的最佳個體進行決策。
2. **環境重置**：初始化一個與訓練時相同，但使用一組**固定的、從未在訓練中出現過的隨機種子**的模擬環境。這確保了所有控制器面對的是完全相同的初始條件、相同的訂單序列與相同的隨機事件，從而消除了隨機性帶來的干擾。
3. **執行評估**：在標準化的環境中，運行一次完整的、固定時長（例如 T\_{eval} = 50,000 ticks）的模擬。
4. **數據記錄**：在模擬過程中，PerformanceReportGenerator 會以固定的時間間隔（例如每 10 ticks）記錄下所有關鍵效能指標的時間序列數據。
5. **重複執行**：為消除單次運行中極端隨機事件的影響，並獲得更具統計意義的結果，上述步驟 2-4 將對每個實驗組**重複執行 K 次**（例如 K=10），每一次都使用一組不同的、但各組共享的隨機種子。最終的效能將取這 K 次運行的平均值。

### 2. 效能比較框架

在 K 次重複評估運行完成後，我們將對每個實驗組的數據進行匯總和比較。比較框架將圍繞 3.2.4 節中定義的關鍵績效指標展開，這些指標可以歸為三大類：

**A. 核心效率指標** 這些指標直接反映了本研究最關心的系統效率。 - **總能量消耗 (Total Energy Consumption)**: 評估系統的能源使用效率，越低越好。 - **訂單完成總數 (Completed Orders Count)**: 衡量系統在固定時間內的總產出，越高越好。

**B. 流程品質指標** 這些指標從側面反映了系統運作的流暢度與服務品質。 - **平均訂單處理時間 (Average Order Processing Time)**: 衡量系統的響應速度，越低越好。 - **平均交叉口等待時間 (Average Intersection Waiting Time)**: 直接衡量交通控制策略的有效性，越低越好。 - **總停止-啟動次數 (Total Stop-and-Go)**: 反映交通流的平順性，越低越好。

**C. 資源利用指標** - **平均機器人利用率 (Average Robot Utilization)**: 反映了在達成產出的過程中，對機器人資源的利用程度。

最終，所有實驗組的平均 KPI 數據將被整理成一個綜合效能比較表，以便於在下一章節中進行深入的分析與討論。此外，所收集的時間序列數據也將被繪製成圖表，以直觀地展示不同策略在模擬過程中的動態行為差異。

# 3.5.4 統計分析與結果驗證

為了使本研究的結論不僅僅基於描述性的平均值比較，而是建立在更為嚴謹的統計學基礎之上，我們將採用假設檢定（Hypothesis Testing）來驗證不同控制策略之間觀測到的效能差異是否具有統計顯著性（Statistical Significance）。這一步對於區分真實的效能提升與隨機波動造成的偶然結果至關重要。

### 1. 假設檢定方法

由於我們對每個實驗組都進行了 K 次獨立的重複評估，這為我們提供了多組樣本數據。在比較兩個不同實驗組（例如，DQN-step vs. QueueBased）在某一特定效能指標（例如，總能量消耗）上的表現時，我們將採用**獨立樣本 t-檢定（Independent Samples t-test）**。

t-檢定的基本流程如下： 1. **建立虛無假設與對立假設**： - **虛無假設 ()**: 兩種控制策略在該指標上的真實平均值沒有差異。例如，。 - **對立假設 ()**: 兩種控制策略在該指標上的真實平均值存在差異。例如，。

1. **設定顯著水準**：
   * 我們將採用傳統的顯著水準（Significance Level） 。這意味著我們能接受的、錯誤地拒絕虛無假設（即第一類錯誤）的機率上限為 5%。
2. **計算 p-值**：
   * 使用 t-檢定計算出 **p-值（p-value）**。p-值表示在虛無假設為真的前提下，觀測到當前樣本數據或更極端數據的機率。

### 2. 結果判讀與結論

根據計算出的 p-值，我們將作出如下判讀： - **如果 p < 0.05**: 我們將**拒絕虛無假設**。這意味著觀測到的效能差異是統計顯著的，我們可以有 95% 的信心認為這兩種策略之間確實存在真實的效能差異。 - **如果 p ≥ 0.05**: 我們將**無法拒絕虛無假設**。這意味著觀測到的效能差異在統計上不顯著，它很可能是由隨機因素造成的，我們沒有足夠的證據證明兩種策略存在真實差異。

通過對所有關鍵效能指標和核心實驗組配對（如 DRL vs. 基線）進行 t-檢定，我們能夠為第四章的實驗結果分析提供強有力的統計支持，從而得出更為可靠和有說服力的研究結論。