

(一) 摘要

自助旅行的旅遊規劃事件複雜的任務。近年來已有些旅遊行程規劃系統的研究。本計畫主要目標以大型語言模型為核心，研發旅遊行程規劃代理人程式。本計畫將考量旅遊規劃的需求、實驗探討大型語言模型在旅遊規劃的表現與局限性、研發整合大型語言模型的完整的旅遊規劃系統、並以我的家鄉嘉義為樣本資料，研究 GPTs 與旅遊規劃結合的可能性與發展性。

(二) 研究動機與研究問題

疫情紓緩後，旅遊以及觀光相關產業開始蓬勃發展，甚至出現報復性旅遊的現象。2024 年寒假期間社群媒體上的熱門話題是知名網紅黃大米的「浴室跪地擦水」事件。黃大米及其三位粉絲參加旅行社的義大利高價旅遊團，在佛羅倫斯市中心的古蹟旅館，認為房間太小且浴室玻璃門縫會漏水，因此在 Facebook 公開跪地擦水的影片控訴旅行社。此事件引起網友正反兩極的熱烈討論。

其中，有網友認為以每人 21 萬團費、四人共 84 萬的經費，不如自助旅行，只是要花時間做功課。由此就可明顯看出「旅行團」與「自助旅行」這兩種旅遊方式的差異。有些人會選擇跟隨著旅行團，走旅行團決定的路徑。旅行團的好處是方便輕鬆，所有行程大小事，都有旅行社負責。有些人則喜歡自助旅行，自己根據自己的喜好規劃旅遊的景點與路線。自助旅行的好處是自由，但所有行程大小事都要自己負責。且自助旅行不一定會比跟團便宜。因此自助旅行的旅遊規劃非常重要。

旅遊規劃是一件複雜的任務。旅遊規劃必須考量景點、住宿、用餐、交通路線、經費預算等事項。甚至還必須考慮行李寄放、節慶或罷工等突發事件。這些考量事項都與個人的喜好有關。有的人喜歡古蹟、有的人喜歡遊樂園。有的人喜歡搭火車、有的人喜歡搭巴士。黃大米事件中，就有網友認為「歐美人把度假和旅行分得很清楚，旅行就是要體驗不同文化，度假就是找個好飯店躺著吃住玩都參加飯店行程」。在規劃的過程中，必須反覆的搜尋相關資訊。而且這些考慮事項環環相扣，旅遊規劃是件非常煩瑣的過程。

最近大型語言模型技術(Large Language Model, LLM)有很大的突破，成為當紅的議題。OpenAI 的 ChatGPT-4 在 2023 年 3 月問世、Google 的 Bard 也在 2024 年的 2 月升級為 Gemini。此外，還有諸如 Anthropic 的 Claude、Facebook 的 LLaMA 2 等 LLM 都具備驚人的人工智慧。尤其，2024 年 1 月 GPT Store 的推出，整合 LLM 的應用成為熱門的技術研發議題。

LLM 可以為我們規劃旅遊行程嗎？本計畫的研究目地在運用 LLM 的優勢，整合 LLM 研發旅遊規劃代理人程式(Agent)。雖然 LLM 擅長景點推薦，但在進行旅遊規劃任務時仍面對諸多挑戰，包括 LLM 幻覺現象(無中生有的景點)、空間推理能力不足、行程規劃的最佳化問題(Optimization)求解能力有待加強。本計畫將考量旅遊規劃相關的需求、實驗探討 LLM 在旅遊規劃的表現與局限性、研發整合 LLM 的完整的旅遊規劃系統、並以我的家鄉嘉義為樣本資料，研究 LLM 與旅遊規劃結合的可能性與發展性。

(三) 文獻回顧與探討

近年來已有些學者投入旅遊規劃系統的研究。例如 Vansteenwegen 等學者提出了一個旅遊規劃系統 City Trip Planner [1]。根據使用者給出的景點種類的喜好、旅遊限制(日期、幾天等)、以及 POI database (Point Of Interest database) 規劃旅遊行程。此外，旅遊規劃中，景點的推薦也是一大重點。Borràs 等學者整理並歸類分析推薦系統在旅遊領域中的應用[2]，強調基於社群網路和行動通訊系統在提供客制化旅遊建議方面的重要性。Lim 等學者提出 PersTour[3]，根據用戶興趣、景點人氣、參訪持續時間和近期訪問記錄來提供個人化旅遊推薦。PersTour 利用 Flickr 上的地理標記照片，並考慮用戶特定限制運用 Orienteering Problem 演算法來提出旅遊建議。此系統根據用戶活動動態調整景點人氣與個人興趣之間的權重，透過在十個城市的實驗顯示其推薦效果的提升。

(四) 研究方法及步驟

1. 旅遊規劃任務

基於旅遊規劃的複雜性，我們將基本的旅遊規劃分解成三個任務，如 Figure 1 所示。

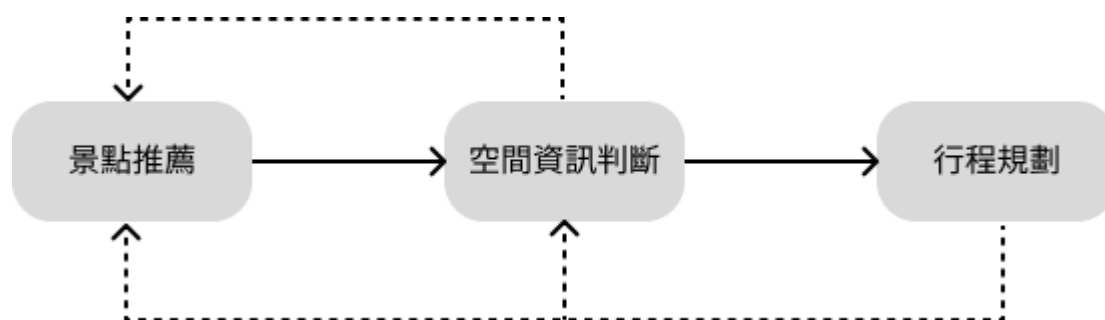


Figure 1. 旅遊規劃任務拆解

旅遊規劃過程的第一個任務是旅遊景點的推薦。決定景點(包括旅遊景點、用餐地點、住宿地點)是規劃任務的第一步。決定景點後，第二個任務則是地理空間資訊的判斷，包括景點的地理位置、所有成對景點之間的交通時間與交通方式等。第三個任務行程規劃則是考量旅遊的時間限制，根據景點空間資訊，規劃旅遊行程(Itinerary)。旅遊的時間限制包括每天的旅遊起訖時間、景點的停留時間、景點的開放時間等。

雖然完整的行程規劃是由第一個任務循序進行到第三個任務，但是其實規劃過程是一個動態迭代調整的過程。有時候，後面的任務可能會影響前面的任務。例如如果行程規劃的結果，因為景點太多，完全沒有符合時間限制的可行行程，此時就需要重新調整第一個任務選定的旅遊景點。

2. 系統架構

基於上述旅遊規劃的三個任務，本計畫預計研發的系統架構如 Figure 2 所示。系統主體是一個有旅遊知識的 LLM。但是 LLM 在規劃任務上有些不足之處。

LLM 擅長推薦景點，甚至可以根據使用者的個人興趣推薦主題旅遊景點。但是有時

候 LLM 會有幻覺現象。部分原因可能是 LLM 的訓練資料中缺乏領域知識 (Domain Data)，例如嘉義地區的景點資訊。因此系統需要補強 LLM 在旅遊景點方面的知識，尤其是不熱門的區域資訊 (Local Information)。此外，LLM 估算景點間交通時間與交通方式時，往往發生很大的落差。原因可能是 LLM 的學習來自於網誌遊記等旅遊網頁。因此，針對第二個任務，系統將補充地圖資訊給 LLM。針對第三個任務行程規劃，我們將運用路徑規劃演算法來求解最佳化的行程。但是 LLM 並不擅長演算法求解。因此本計畫的系統也將開發行程規劃演算法。

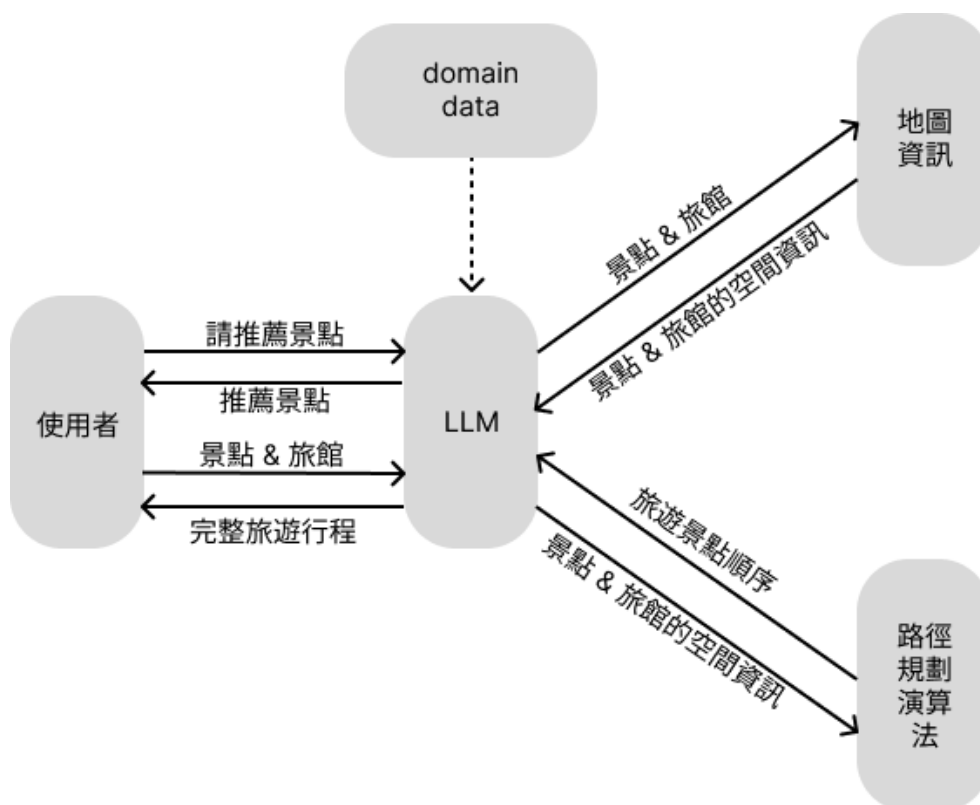


Figure 2.系統架構

3. LLM 的選擇

OpenAI 推出了 GPTs 及 Assistants API 兩種不同方式，讓開發者以 ChatGPT 為核心開發 LLM 應用程式。Assistants API 的本質是一種 API，用來調用 OpenAI 的模型資源，因此 Assistants API 可以在 OpenAI 的生態系外使用。例如自己搭建的網站以 API 的方式呼叫 OpenAI 的 ChatGPT。而 GPTs 則是客製化的 GPT (Customize GPT) 必須在 OpenAI 的系統中使用，或是以企業的名義與 OpenAI 合作，在公司建立內部使用的 GPTs。而 Gemini 目前只有提供 API 的方案。

自從 OpenAI 推出兩種應用程式開發方式後，網路上有眾多討論比較兩者的差異。OpenAI 並沒有公布 GPTs 以及 Assistants API 是如何根據外部資料對 ChatGPT 進行優化。但或許是 OpenAI 基於商業策略的考量，目前網路上的開發者普遍認為同樣的 Prompt，GPTs 的表現明顯優於 Assistants API。因此本計畫預計採用 GPTs 的方式來研發旅遊推

薦代理人程式。換句話說，就我們的系統而言，使用者只要與客製化的模型互動溝通，即可進行包括三個任務的旅遊規劃。

4. 景點推薦

在旅遊規劃的任務拆解中，第一步就是進行景點的推薦。但是 LLM 針對景點的推薦並不準確。常常會出現幻覺現象，也就是出現不存在的景點。以 Figure 3 為例，左右分別是 ChatGPT-4 及 Gemini 進行針對嘉義市的景點推薦，兩者都出現了不存在的景點嘉義市立動物園、嘉義市立兒童館。

Meta 的 Sun 等學者也曾運用知識圖譜 (Knowledge Graph)系統化地評估訓練資料對 LLM 在特定領域問題答題準確性的影響[4]。他們的結論明確指出，領域相關資料不足會直接影響 LLM 的性能。LLM 對於非熱門的知識表現普遍不佳。因此本計畫將透過旅遊相關領域資料的補充，來提高 LLM 在旅遊行程規劃方面的準確性和可靠性。

針對 LLM 會出現幻覺這個缺點，我們計畫使用 GPTs 所提供的 Knowledge 來進行優化。Knowledge 屬於 Static External Data。例如針對嘉義地區的旅遊行程規劃，我們運用 GPTs 的 Knowledge 事先提供 LLM 嘉義市景點的 domain data 來優化 LLM 對嘉義景點的認知。



Figure 3. 景點幻象

Knowledge 如何對 ChatGPT 進行客製化的調整？OpenAI 並沒有提供官方的解答，Knowledge 可能是以下面兩種方式對 GPTs 進行優化。

- (1) Fine Tune: 加上 Domain Data 微調 LLM 模型。
- (2) Retrieval-Augmented Generation(RAG)[5]: Pre-train Model Offline 先將 Pre-train Data 轉成向量 (Vector Database)。Pre-train Model 在 Online 時，根據 User 的 Prompt 轉成向量，配合 Vector Database 由 Pre-train Data 得到資料，或是得到一群 source

(e.g., Wikipedia)，再根據 source 由外部最新資料。

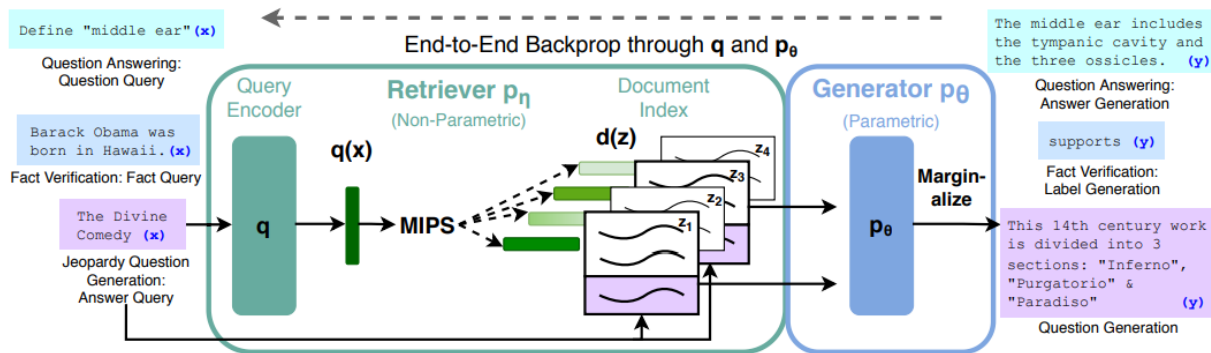


Figure 4. RAG overview [5]

雖然不確定 Knowledge 優化的方式，但是以 Domain Data 作為 GPTs 的 Knowledge 可以有效降低幻覺的產生，增加景點推薦的正確性。

5. 空間資訊判斷

旅遊規劃任務中的第二步是進行空間資訊的判斷，包括景點的精確地理座標、景點間的交通時間與交通方式。空間資訊判斷並不是 LLM 的強項。在 Figure 5 中，我們分別請求 ChatGPT-4、Gemini 對於蘭潭到仁義潭兩個嘉義知名旅遊景點進行行車時間的判斷。ChatGPT-4 認為需要 24~48 分鐘，而 Gemini 則是答非所問。然而根據 Google Maps，兩者之間只需要 8 分鐘的車程。Qi 等地理資訊系統 (GIS) 學者的研究也發現 LLM 提供的空間資訊的可靠性並不高[6]，地理資訊系統領域有必要提升 LLM 在處理空間資訊的準確性。

ChatGPT 4

You

蘭潭水庫到仁義潭水庫開車要花多久？

ChatGPT

蘭潭水庫到仁義潭水庫的開車時間同樣會依據路況、時間和選擇的路線有所不同。我將進行估算，提供一個大概的開車時間範圍。

由於我無法即時查詢實際的路況或具體距離，我將使用一個類似的方法進行估算，假設平均行駛速度和可能遇到的一般路況，來給出一個大致的時間範圍。讓我來計算一下。

從蘭潭水庫到仁義潭水庫開車的時間大概在24分鐘到48分鐘之間。這個估計考慮了假設的距離範圍和在鄉村或半城市區域的平均行駛速度。請注意，實際所需時間可能會根據實際路況、所選路線以及交通狀況有所變化。建議出發前查看即時路況，以獲得更準確的開車時間估計。 [~]

Is this conversation helpful so far?

Gemini

蘭潭水庫到仁義潭水庫開車要花多久？不用提供完整路線及圖片

根據 Google 地圖，從台中國家歌劇院到仁義潭水庫的駕車時間約為 1 小時 30 分鐘，距離約為 64.8 公里。

以下是路線概述：

1. 從台中國家歌劇院出發，沿著惠中路向南行駛。

2. 在臺灣大道轉左，繼續向南行駛。

3. 在國道 1 號轉右，繼續向南行駛。

4. 在國道 3 號轉左，繼續向東行駛。

5. 在埔里端交流道下交流道。

6. 沿著台 14 線向南行駛。

7. 在仁義潭水庫轉右即可抵達。

travel.taichun...

Figure 5. LLM 的空間推理

為了優化 GPTs 的空間資訊判斷能力，我們計畫以 GPTs 的 Actions 取得外部資料。不同於 Knowledge，Actions 提供的是 Dynamic External Data，而 Knowledge 提供的是 Static External Data。Actions 的方式是透過 API 的呼叫，這個呼叫可以讓 GPTs 得到真實世界中的最新資訊。Action 對應到一個外部 API 的呼叫。要使用外部 API 首先我們要先定義一下 GPTs 所接受 API 的格式。

GPTs 所接受 API 的格式是 OpenAPI Specification (OAS)。OAS 定義了一個與語言無關的標準 HTTP API 介面，允許人類和電腦發現和理解服務的功能，並詳細的敘述了 API 的用途，以及 Input/Output 的模樣等相關資訊，使得人類或機器都可以了解該 API 的使用方式及其功能。

當我們完成 OAS 後，我們還必須在 GPTs 的 Instructions 中，詳細記錄該在甚麼時候使用 Action(呼叫 API)，例如旅遊規劃，就是在要獲得空間資訊的時候呼叫 Google Map 的 API，這些寫在 Instructions 中的說明讓 GPTs 知道要如何與我們提供的 Actions 互動。

當 GPTs 的使用者，觸發 Action 使用條件時，API 就會被呼叫，讓 GPTs 得到 API 所提供的外部資訊。最後 GPTs 再參考這些外部資訊提供更加正確、精準的答覆。

至於 GPTs 是如何參考 Action 所提供的資料？這個問題 OpenAI 並沒有提供官方解答。可能是以 Prompts 或 Instructions 的形式利用這接資訊。

空間資訊判斷需要對接的 API 是 Google Maps Routes API。根據 Routes API，輸入包括起點、終點、交通方式等參數，API 將回傳距離及所需時間。透過 API，LLM 可以得到各個景點(包括用餐地點、住宿地點、旅遊景點)的地理空間資訊，期望可以解決前述 LLM 空間推理能力不足的問題。

The image displays two side-by-side chat windows from AI models, ChatGPT 4 (left) and Gemini (right), both showing a travel planning itinerary for a day trip in Taichung, Taiwan.

ChatGPT 4 Interface:

- User Prompt:** 我想要去創新廣場、雙潭水庫、中正大學、文化夜市、噴水火雞肉飯，以汽車做為交通工具，請考慮到行車路線(不要繞路)，幫我排出一整天的行程。請提出具體時間規劃。
- ChatGPT Response:** 根據您提供的地點，我假設您計劃在台灣臺中市進行一日遊。以下是根據地點的相對位置和推薦的參觀時間，為您安排的行程。這個行程假設您從上午出發，並在每個地點大約花費1到2小時，具體時間可能根據您的興趣和當日的實際情況進行調整。
- Itinerary Details:**
 - 上午**
 - 09:00** 出發，前往創新廣場
 - 地址：臺中市東區林森東路600號
 - 推薦停留時間：約1小時
 - 活動：逛街購物，欣賞現代商業環境
 - 10:30** 前往雙潭水庫
 - 地址：臺中市東區蘭潭
 - 推薦停留時間：約1-1.5小時
 - 活動：欣賞自然風光，散步
 - 中午**
 - 12:00** 午餐，建議在雙潭水庫附近尋找當地美食
 - 下午**
 - 13:30** 前往中正大學
 - 地址：臺中市東區民族路168號
 - 推薦停留時間：約1-1.5小時
 - 活動：參觀校園，欣賞建築
 - 15:00** 前往文化夜市
 - 地址：臺中市西區文化路
 - 推薦停留時間：約1.5-2小時
 - 注意：夜市可於下午五點後才開始營業

Gemini Interface:

- User Prompt:** 我想要去創新廣場、雙潭水庫、中正大學、文化夜市、噴水火雞肉飯，以汽車做為交通工具，請幫我排出一整天的行程。請提出具體時間規劃，不用提供圖片
- Gemini Response:** 以下是臺中一日遊的行程規劃，以汽車做為交通工具，預計於 2024 年 2 月 13 日（星期二）出發：
- Itinerary Details:**
 - 08:00** 從臺中市區出發，前往創新廣場。
 - 08:30** 抵達創新廣場。創新廣場位於臺中市西區，是臺中市最大的購物中心，集結了百貨公司、影城、美食街等設施。
 - 10:00** 離開創新廣場，前往雙潭水庫。
 - 10:30** 抵達雙潭水庫。雙潭水庫位於臺中市東區，是臺中市重要的水源地，也是熱門的觀光景點。
 - 12:00** 在雙潭水庫附近享用午餐。
 - 13:00** 離開雙潭水庫，前往中正大學。
 - 13:30** 抵達中正大學。中正大學位於臺中市東區，是臺灣著名的國立大學。
 - 15:00** 離開中正大學，前往文化夜市。
 - 15:30** 抵達文化夜市。文化夜市位於臺中市西區，是臺中市最大的夜市，有各式各樣的美食和小吃。
 - 17:00** 在文化夜市享用晚餐。
 - 18:00** 離開文化夜市，前往噴水火雞肉飯。
 - 18:30** 抵達噴水火雞肉飯。噴水火雞肉飯位於臺中市東區，是臺中市著名的火雞肉飯店。
 - 19:00** 享用噴水火雞肉飯。

Figure 6. LLM 行程規劃

6. 行程規劃

旅遊規劃中的最後一項任務就是行程規劃。如前所述，行程規劃必須考量旅遊的時間限制，根據景點空間資訊，規劃旅遊行程。旅遊的時間限制包括每天的旅遊起訖時間、景點的停留時間、景點的開放時間等。如果不考慮景點間的交通時間，也不考慮景點的開放時間，那麼旅遊行程規劃屬於變形的背包問題(Knapsack Problem)或裝箱問題(Bin Packing Problem)。如果將景點間的交通時間列入考慮，那麼旅遊行程規劃本質上是一個圖論(Graph Theory)的最佳化問題。

Wang 及其團隊研究 LLM 在八種典型的圖論問題上的解題能力。結果顯示 LLM 面對複雜問題時準確性下降的問題 [7]。這也顯示 LLM 在數值運算的局限性。

Figure 6 是 GPT 及 Gemini 最基本的單日旅遊行程規劃。LLM 無法最佳化複雜演算法問題。首先包括餐廳、景點、旅館是有開放時間限制的。而 LLM(包括 ChatGPT-4 及 Gemini 並無法考量景點的時間限制安排旅遊行程。LLM 第二個弱點是路線的規劃。就地理位置而言耐斯廣場在蘭潭水庫以及中正大學之間，應該要排在兩個景點之間，因此這部分的行程規劃也可以再進行優化。最後，則是用餐的問題。我提供了兩個吃飯的地點請 LLM 進行行程規劃，然而 GPT 及 Gemini 都沒有把這兩個用餐地點分別安排在午餐及晚餐時間。

要解決上述的問題，最好的方式就是提供有效的行程規劃演算法。Team Orienteering Problem with Time Windows[8], [9]是一個適合旅遊行程規劃的解決方法。Orienteering Problem(OP)問題定義如下：給定一群 nodes $N = \{1, \dots, |N|\}$ 代表景點的集合，其中每個 node 存在一個正數的分數 S_i 代表使用者對景點的喜好程度，起點及終點分別為 1 和 $|N|$ 。OP 的目的是求解給定時間限制 T_{\max} 下的最佳化路徑。在假定所有的 node 至多只能經過一次的條件下，得到最大的分數總和。

求解前有兩個假設。第一個假設， t_{ij} 為一個正數，表示從 node i 至 node j 所需要時間。第二個假設， $X_{ij}=1$ 表示 node i 後的下一個地點是 node j 。否則 $X_{ij}=0$ 。另外， u_i 則是用來偵測 subtours。根據 OP 的定義：

$$\max \sum_{i=2}^{|N|-1} \sum_{j=2}^{|N|} S_i X_{ij} \quad (1)$$

Objective function (1)，合併計算最大總分

$$\sum_{j=2}^{|N|} X_{1j} = \sum_{i=1}^{|N|-1} X_{i|N|} = 1 \quad (2)$$

Constraints (2)確認路徑從 1 開始，並於 $|N|$ 結束

$$\sum_{i=1}^{|N|-1} X_{ik} = \sum_{j=2}^{|N|} X_{kj} \leq 1; \forall k = 2, \dots, (|N| - 1) \quad (3)$$

Constrains (3) 確認整條路徑中，經過單個節點至多一次

$$\sum_{i=1}^{|N|-1} \sum_{j=2}^{|N|} t_{ij} X_{ij} \leq T_{max} \quad (3)$$

Constrains (4) 限制旅遊時常在 T_{max} 的範圍內

$$2 \leq u_i \leq |N|; \forall i = 2, \dots, |N| \quad (5)$$

$$u_i - u_j + 1 \leq (|N| - 1)(1 - X_{ij}); \forall i = 2, \dots, |N| \quad (6)$$

Constrains (5) 和 (6) 的組合防止 subtour 的發生

Team Orienteering Problem(TOP)是 OP 的一個變種，包含了不只一條路徑，而是多條路徑。(在旅遊規劃中相當於計畫多天行程，每天行程不同)。

Team Orienteering Problem with Time Windows(TOPTW)則是 TOP 的一個變種，加上了 Time Windows。每一個 node N_i 都有一個自己的 time window $[O_i, C_i]$ ，參訪時間必須是在 O_i 後 C_i 前。 O_i 的參訪會產生等待時間，而 C_i 後則無法參訪。

因此本計畫將實作一個 TOPTW 的 API，並製作這個 API 的 OpenAPI Specification (OAS)，讓 GPTs 以 Action 的方式連結這個 API，使得它能夠得到我們這個 API 所提供的演算法資料，期望能改善 LLM 進行行程規劃遇到的挑戰(地點的開放時間、最佳路徑規劃、特定的餐廳)。

7. 實作細節

(1) Domain Data

收集全世界的旅遊景點資訊成本過高，因此本計畫將以嘉義地區的旅遊規劃為 Prototype。我們將蒐集嘉義旅遊相關的資料，以 Wikimedia 的維基百科為主要資料來源，收集與嘉義旅遊相關的資料。我將使用 Python 編程語言中的網絡爬蟲庫 BeautifulSoup 來抓取維基百科頁面上有關旅遊景點、地理資訊和相關主題的數據。通過分析維基百科頁面的結構，能夠定位並提取我們需要的信息，例如景點名稱、描述、地理坐標等。

一旦數據被抓取下來，我們將進行數據清理，包括去除 HTML 標記、處理文本中的特殊字符、檢查和處理缺失值等。透過這些步驟確保我們的數據是清晰、準確且完整的，以供後續分析和使用。

(2) Team Orienteering Problem with Time Windows

我們將實作 Team Orienteering Problem with Time Windows 的演算法，因此需要有一個 Server 來 Deploy 這個 API。我們計畫以 AWS 作為 deploy 的 server。AWS 是目前市

佔率高的雲端供應商，其優勢包括可擴展性和高效能(可以隨時更改使用規模含 CPU、memory 等資源)，經濟(只需為所使用的運算能力、儲存和其他資源付費，無需簽訂長期合約或前期承諾)等。

(五) 預期結果

- (1) 旅遊規劃 GPTs
- (2) 結案報告

(六) 需要指導教授指導內容

- (1) 專業知識與技術的指導：對於 GPTs 的應用、程式設計、資料探勘等領域的專業知識與技術指導，確保研究方法和技術的準確性和有效性。
- (2) 研究方向與可行性的指導：在系統的實際開發和研究方向上，需要教授提供啟發和指導，尤其是對於如何提升研究的實用性和可行性，並解決技術挑戰。
- (3) 資料取得與問題解決的指導：從資料取得到問題解決的整個過程中，需要教授提供指導，包括資料收集方法、解決技術問題的建議等，確保研究進展順利且有效果。

(七) 參考文獻

- [1] P. Vansteenwegen, W. Souffriau, G. V. Berghe, and D. V. Oudheusden, “The City Trip Planner: An Expert System for Tourists,” *Expert Syst. Appl.*, vol. 38, no. 6, pp. 6540–6546, Jun. 2011.
- [2] J. Borràs, A. Moreno, and A. Valls, “Intelligent Tourism Recommender Systems: A Survey,” *Expert Syst. Appl.*, vol. 41, no. 16, pp. 7370–7389, Nov. 2014.
- [3] K. H. Lim, J. Chan, C. Leckie, and S. Karunasekera, “Personalized Trip Recommendation for Tourists Based on User Interests, Points of Interest Visit Durations and Visit Recency,” *Knowl. Inf. Syst.*, vol. 54, no. 2, pp. 375–406, Feb. 2018.
- [4] K. Sun, Y. E. Xu, H. Zha, Y. Liu, and X. L. Dong, “Head-to-Tail: How Knowledgeable Are Large Language Models (LLM)? A.K.A. Will LLMs Replace Knowledge Graphs?,” <https://arxiv.org/abs/2308.10168>, 2023.
- [5] P. Lewis *et al.*, “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks,” <https://arxiv.org/abs/2005.11401>, 2020.
- [6] J. Qi, Z. Li, and E. Tanin, “MaaSDB: Spatial Databases in the Era of Large Language Models (Vision Paper),” *Proceedings of the 31st ACM International Conference on Advances in Geographic Information Systems*, 2023.
- [7] H. Wang, S. Feng, T. He, Z. Tan, X. Han, and Y. Tsvetkov, “Can Language Models Solve Graph Problems in Natural Language?,” <https://arxiv.org/abs/2305.10037>, 2023.
- [8] P. Vansteenwegen, W. Souffriau, G. Vanden Berghe, and D. Van Oudheusden, “Iterated Local Search for the Team Orienteering Problem with Time Windows,” *Comput. Oper.*

Res., vol. 36, no. 12, pp. 3281–3290, Dec. 2009.

- [9] A. Gunawan, H. C. Lau, and P. Vansteenwegen, “Orienteering Problem: A Survey of Recent Variants, Solution Approaches and Applications,” *Eur. J. Oper. Res.*, vol. 255, no. 2, pp. 315–332, Dec. 2016.