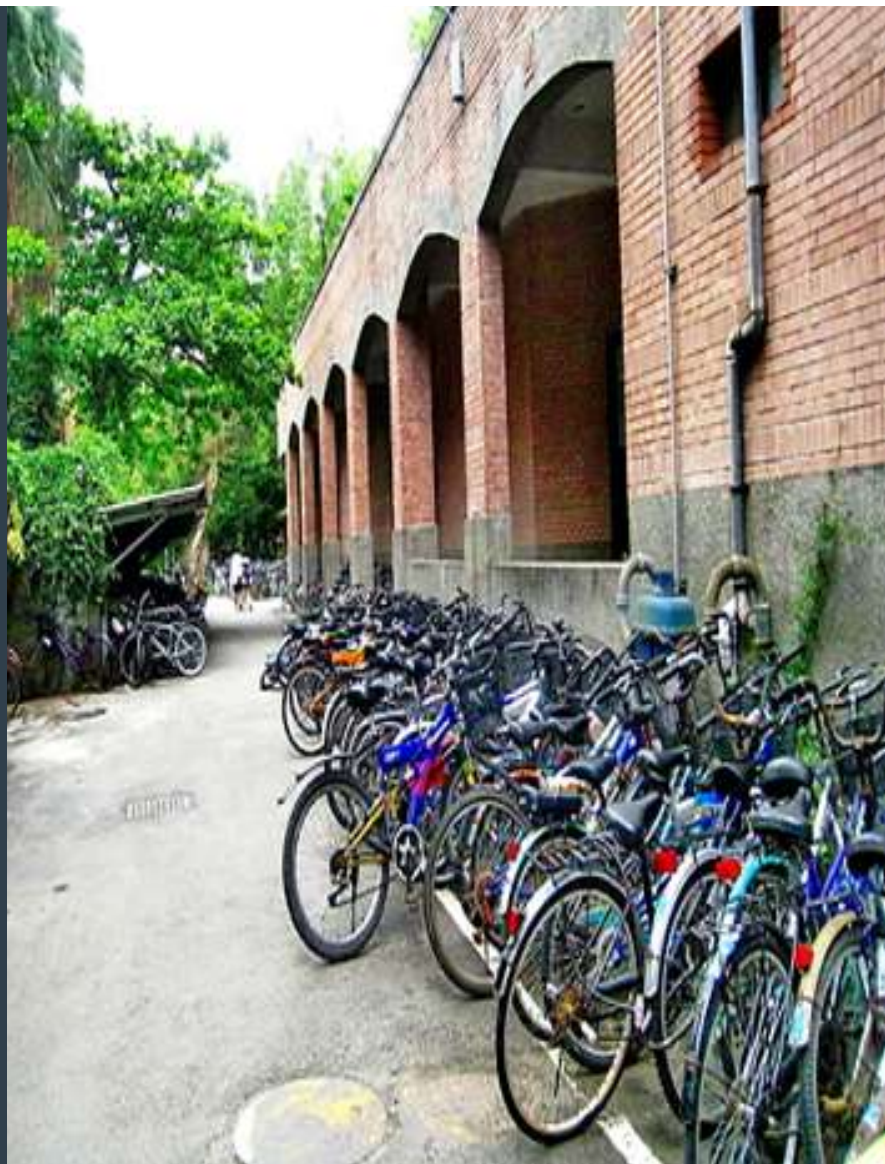
OR FINAL PROJECT

Team 5



DESCRIBE OUR PROBLEM

被水源阿伯拖車大概是所有台大學生共同的回憶，然而很不幸地，雖然水源阿伯一直努力拖吊，台大的違停腳踏車仍然很多，於是我們很好奇，如果水源阿伯希望能夠盡可能地把所有影響交通的違停腳踏車都拖吊走，應該怎麼規劃每次的行車路徑與拖吊地點才能有效解決違停問題。

我們將問題定義為：若有一位水源阿伯，有一台能載 TC 台腳踏車的卡車，一天只能出動兩次，在台大有 M 個地點會有違停車輛 N_i 台，且每個地點之間的移動固定需要 T_ij 分鐘（即距離與車速維持相等），由於每個地點違停的嚴重程度不同（例如在大一女前面違停會嚴重阻礙通行，但在新體前面違停對交通毫無影響），所以賦予 M 個地點不同的 Weight_i 權重。雖然有些地點違停不嚴重，但如果長期都不去拖車會讓學生覺得違停也沒關係，所以每週還是要去該地點拖 F_i 次。配合大一女現行拖吊政策，我們規定在特定幾天一定要到特定地點拖吊。水源阿伯雖然身強體壯，但超時工作仍對身心健康不好，所以規定一次拖吊不能超過 Time分鐘。而會佔用時間的除了在不同地點之間移動各需要 T_ij 分鐘以外，還有每搬一台腳踏車上去卡車需要 TB 分鐘。除非卡車塞滿或是要超過時間限制Time了，否則水源阿伯造訪一個地方就一定會把現場所有違停車都搬走。假定學生違停後如果沒被拖吊，就會很開心地把車騎回宿舍，隔天會有新的一群人違停。

我們想探討，水源阿伯應該依照什麼順序、去哪幾個地點拖吊，才能最大化拖吊效果（拖吊車數 乘以 該地點權重）。

這個問題不僅止於水源阿伯在台大校園拖吊，還有像是垃圾車應該去哪裡收垃圾等問題，也可以用類似解法。

FORMULATION

我們將問題分成兩階段，第一階段先選擇所有要去的拖吊地點，第二階段會找出最短路徑，從水源拖車場出發，各經過上述地點一次，再回到水源拖車場。

在第一階段，我們建立了一個 Integer Programming，從中尋求下述資料

1. 要去拖吊哪些地點
2. 哪個地點是最後一點，與應該在那邊拖幾台車（因只有最後一個地點可以因為卡車載運限制或超時而不拖滿）

以下是我們的 IP:

ASSUMPTIONS AND CONSTANTS

1. TC, the truck's capacity
2. M, the number of bike parking spots in NTU
3. N_i, the number of parking violations on the i^{th} spot
4. T_{ij}, the time spent for the truck to travel from spot i to j
5. $Weight_i$, the weight of each spot
6. F_i, the least times spot i be visited per week
7. $Time$, the maximum time for one towing guard to be on shifts per week
8. TB, the time required to move a bike onto the truck

PHASE I

$$\begin{cases} x_{dm}, \text{tows all violation bikes at the } m^{th} \text{ spot on day } d. \\ z_{dm}, \text{partially tows at the } m^{th} \text{ spot on day } d. \end{cases}$$

q_{dm}, the number of towed bikes at spot m on day d

$$\max \quad \sum_d \sum_m q_{dm} \times W_m$$

$$\text{s.t.} \quad q_{dm} \leq x_{dm} \cdot N_{dm}$$

$$\sum_m z_{dm} \leq 1, \forall d$$

$$z_{dm} \leq x_{dm}, \forall d, m$$

$$q_{dm} \geq x_{dm}$$

$$x_{dm} \cdot N_{dm} - q_{dm} \leq M_1(z_{dm})$$

$$\sum_m q_{dm} \leq TC, \forall d$$

$$\sum_d x_{dm} \geq F_m, \forall m$$

$$\sum_d \sum_m q_{dm} \cdot W_m \leq B$$

FORMULATION

在第二階段，我們拿著已知的要拖吊的點（若有多個解，則必須獲取每個解）與 objective value 安排最短路徑（即 TSP），並限制第一點必須為水源，若有任一點是未完全拖吊，則必須放在最後一點（是否有這個 constraint 會在執行前先以程式判斷是否動態加入）。

PHASE II

$$\min \quad \sum_{(i,j)\in E} T_{ij} \times y_{ij}$$

$$\text{s.t.} \quad \sum_{i\in V, i\neq k} y_{ik} = 1, \forall k \in V$$

$$\sum_{j\in V, j\neq k} y_{kj} = 1, \forall k \in V$$

$$y_{ij} \in \{0,1\}, \forall (i,j) \in E$$

$$u_0 = 1, u_{last} = N$$

$$2 \leq u_i \leq (N-1), \forall i \in V/\{1,N\}$$

$$u_i - u_j + 1 \leq (n-1)(1-x_{ij}), \forall (i,j) \in E, i,j \neq 1,N$$

若每個解所對應的 TSP 最佳解都超過時間上限，則回到第一階段，但限制 Objective value 必須小於目前的值（具體而言，會造成 objective value 為目前解減去一台權重最小的違停腳踏車，並令其為 Big Number），如此則能確保我們只會找到次佳解，而不會錯過比較好的解或是剛剛在第二階段已經證明不可行的解。

若在第二階段有找到符合時間上限的解，則結束迴圈，回傳最佳解的路徑。

DATA

由於我們沒有台大各地點的違停數量以及其他必要資訊，故我們使用兩種方法來解決之。我們首先根據組員的日常生活經驗為每個地點做評分，再發行問卷詢問其他台大師生，我們總共收到 68 份有效回應，並綜合兩者結果得出我們所使用數據。我們相信，若這個模型在水源阿伯（或其他類似場景）有蒐集歷史資料的情況下，能給出足夠精確的答案，但我們並沒有這些數據。

我們嘗試過尋找這個問題在其他場景的 Open Data（例如垃圾車、收包裹等），但也都沒有找到足夠精細的資料。

以下為我們設定的index對應的停車點：

1	小福
2	普通大樓
3	化工系館
4	新體育館
5	文學院
6	大門
7	化學系
8	新生大1樓
9	博雅館
10	舊體育館
11	電機一館
12	新月台
13	校外捷運公館站
14	水源校區
15	科技大樓周邊
16	電機二館
17	總圖書館
18	計算機中心
19	女八九宿舍
20	社會系館
21	博理館

21	博理館
22	機械系1
23	資工所
24	應力所
25	心理系
26	語言中心
27	圖資系
28	萬才館
29	工綜館
30	行政大樓
31	共同教室
32	研一、大一女舍
33	小小福
34	鹿鳴廣場
35	舟山路
36	管理學院
37	行政大樓門口
38	女一舍門口
39	農綜館
40	地質系
41	駐警隊

42	學生心理輔導中心
43	女生1235宿舍
44	明達館
45	知武館
46	水工所
47	航測館
48	中菲大樓
49	生物產業機電學系

OBSTACLES

思考解法時的困難

在成功完成這份project的過程中，我們並非一次就想到最後的解決方法，我們也經歷了一些絆腳石。原本組員們一致認為是可行的做法，在實踐時才發現其中的困難點，不過這些問題反而給我們更縝密思考的機會，在不斷的討論與修正後，才得以產出最後完整的結果。以下為我們曾經遇到的困難：

1.原本以為可以用TSP來當作解決辦法，但是後來發現若使用此方法的話，並非每個點都會走到，原因在於牽扯到順序問題，如此和TSP就會有解法上的落差，因為TSP要走所有的點，但是我們並不打算走完全部的節點，相反地，我們只會選擇部分的拖車地點進行拜訪。
之後我們想出用兩階段的推演來解決問題，我們將限制式分為第一階段和第二階段來計算，等於是我們需要生成兩個program來分別解決第一階段和第二階段。

2.由於卡車的負重有限，我們無法確認是否會有一個地點（可能是終點）的違停車輛不會被拖完。因此我們考慮過要限制: 每個地點要不是全部違停車輛都拖完，否則就都不拖。但是，我們最後並沒有這樣做，選擇保有最後一個地點可以不用拖完的彈性。

3.我們一開始選擇用路程距離進行限制，也就是假設油箱有限，一趟不能走超過某個距離上限。然而，在實際應用層面似乎沒有很合理，尤其放在台大校園的場景下，距離似乎不是大問題。因此，思考了很久後，我們最後決定用時間進行限制，也比較符合實際考量。

4.違停車輛是否會動態改變（第n天會影響第n+1天）也是我們考慮的點，但後來我們決定撇除這個因素，因為DP操作上過於複雜，所以就選擇了靜態的違停車輛數。

5.水源阿伯每天要出門幾趟也是我們考慮的條件之一，我們考慮程式的複雜性與資料的特性後，決定讓水源阿伯一天只會出門拖吊一趟。

OBSTACLES

6.關於Phase II ：由於Phase II可行與否會決定我們該不該要iterate，來找到下一個可行解，然而，我們發現一周當中若有1天的路徑為infeasible（儘管其他路徑都feasible），我們便需要全部重新iterate。
為了讓運作起來更加合理，我們決定擴大限制式的時間尺度: 只要一周總和的時間都符合，那個解答便算是feasible。

7.如果Phase I有數個解，我們就會需要尋找所有可行的整數解，並將這些解帶入Phase II計算，這會很明顯地增加時間複雜度。

寫程式時遇到的困難

1.Constraint的問題：當我們在討論解法時，我們有設定一個參數 z，它是在控制每個拖車點是否為最後一個拖車點，如果是的話，我們將允許這個拖車點能不把所有車拖完，也就是說其他有前往的拖車點，違停的腳踏車一定都會被拖完。但是當時沒有思考到我們忽略了一項限制，那就是我們忘記加上「除了最後的拖車點外都需將該點的腳踏車全數拖完」這個限制式，我們一開始只有限制最後一個點可以不把全部違停的車輛拖完，而這樣的疏忽是我們在實際執行寫好的程式碼時，發覺結果不如預期才發現的，最後我們也有想出解決辦法來讓程式的結果較合理。

2.浮點數精確度問題：在寫程式時，我們的許多參數都是設成gurobi的binary型態，即為理論上這個變數的值應該只有0和1，但是當我們在檢查參數是不是等於0或是1的時候，才發現 gurobi 中 binary 型態的值其實不是interger 的0和1，而是floating number，所以1有可能是0.9999999999999996，而0也有可能是0.0000000000004，因此我們在執行程式的時候，0.9999999999999996可能不會等於1，0.0000000000004也可能不會等於0，我們便修改程式碼來讓結果符合我們的期待。

SOLUTION

我們將運算第一階段的過程記錄下來，首先，Phase I印出的值為跑出的最大效益為多少，即「Phase I : 5028.63100963236」，這代表說我們我們讓水源阿伯在這次拖吊的過程能夠得到最大5028.63100963236的效益。但在進入第二階段後，若該解對應的 TSP 最佳解超過時間上限，我們就會重新跑第一階段，我們也將所有重跑過的值記錄下來，直到程式印出「Success！」，該解才是我們得出最後的最佳解。可以發現Phase I的值的確越來越小，原因如上所述，這樣可以讓我們以最少次數找到解答。

第二個資訊為能得到相同最大效益的組合數，即「Solution Count：10」，這代表有10個拖吊地點的組合都能達到最佳效益，而我們就會拿這10組的資料去Phase II跑，以計算最短的拖吊時間。

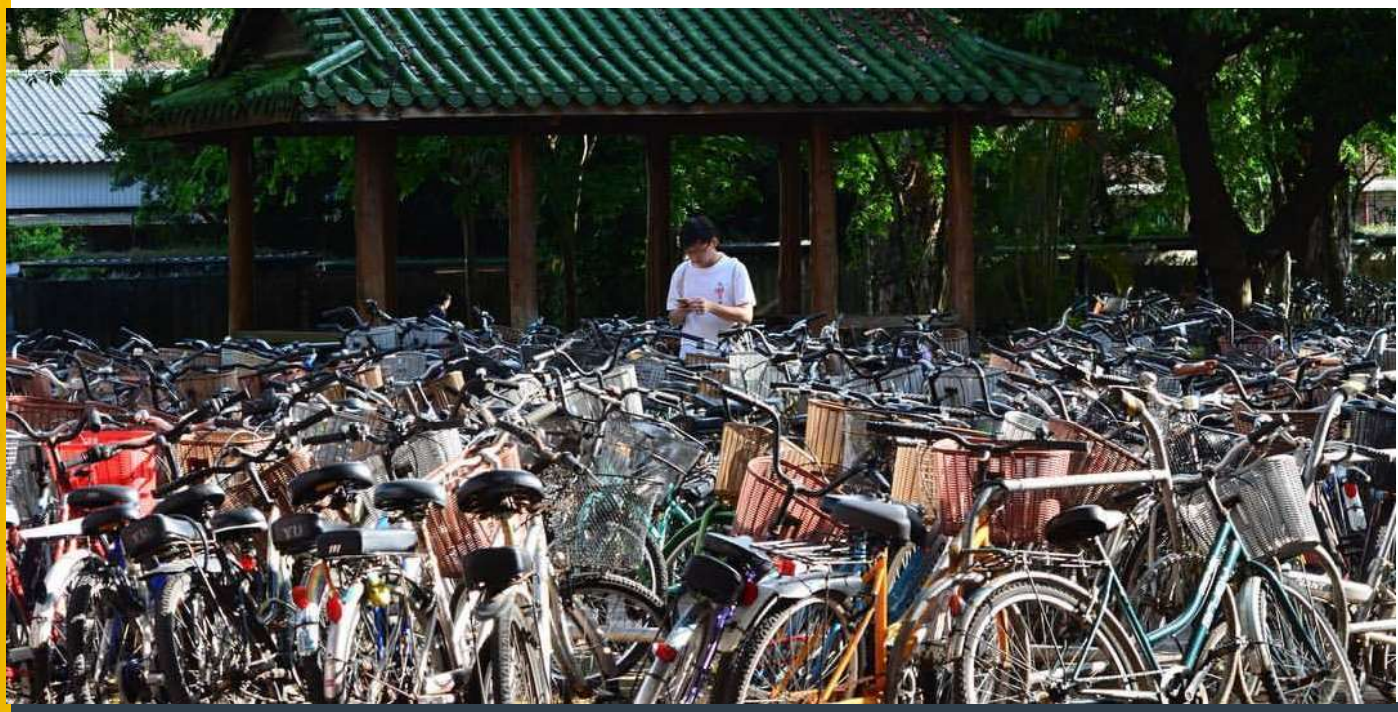
第三個資訊是水源阿伯能花費最少的拖吊時間為多少，即「Time gap [4, 15, 16, 18, 21, 15, 16, 13, 15, 16]」，這個陣列中有10個數是因為「Solution Count：10」，而這些數值代表這10個組合各別花的最短路徑時間和我們限制的時間之間的時間差。

當我們在Phase I找到符合條件的最佳拖吊地點組合後，經過Phase II的運算，我們將該最佳解一周內每天要拖吊的地點與拖吊的違停腳踏車數印出來，而地點的排序就是最佳行駛路徑的順序。以「At 17 catch 46」為例，意即在編號為17的地點拖吊46台違停腳踏車，我們也在每天都印出總共拖吊的車輛數，即「total_bike = 200」，我們可以看到每天的總拖吊數都等於卡車最大的負荷量，可知我們沒有浪費任何一個能夠裝載違停車輛的機會。「total_benefit = 5021.455000000001」是說這組最佳解能夠得到最大5021.455000000001的效益，而「total_time = 1617」意思為一周最短的拖吊時間為1617分鐘。

SOLUTION

綜上所述，我們得到的最佳解之效益為5021.455000000001，一個禮拜所花的拖吊總時間為1617。接者是關於每天的路徑，第一天水源阿伯會先去總圖書館拖吊46台 -> 普通大樓拖吊22台 -> 校外捷運公館站拖吊96台 -> 博理館拖吊1台 -> 地質系拖吊3台 -> 心理系拖吊2台 -> 小福拖吊30台，第二天則是先去科技大樓周邊拖吊11台 -> 校外捷運公館站拖吊110台 -> 資工所拖吊1台 -> 女生1235宿舍4台 -> 總圖書館拖吊74台，第三天會先去校外捷運公館站拖吊102台 -> 圖資系拖吊4台 -> 總圖書館拖吊74台 -> 航測館拖吊3台 -> 中菲大樓拖吊10台 -> 水公所拖吊7台，第四天的路線為普通大樓拖吊24台 -> 總圖書館拖吊50台 -> 校外捷運公館站拖吊96台 -> 小福拖吊30台，第五天會是先去校外捷運公館站拖吊99台 -> 化工系館拖吊5台 -> 小福拖吊29台 -> 工綜館拖吊2台 -> 舟山路拖吊3台 -> 社會系館拖吊1台 -> 總圖書館拖吊61台，第六天將先去校外捷運公館站拖吊106台 -> 普通大樓拖吊27台 -> 應力所拖吊2台 -> 舟山路拖吊3台 -> 總圖書館拖吊62台，最後一天的路徑則是先去電機一館拖吊3台 -> 校外捷運公館站拖吊115台 -> 大門拖吊26台 -> 總圖書館拖吊55台 -> 生物產業機電學系拖吊1台，以上即為我們的結果。

有了以上的結果，我們能夠幫助水源阿伯提供高工作效率，因為他可以花最少的時間來最大化他的工作效益，此外我們也做出路徑規劃，讓水源阿伯知道該怎麼走才是最佳路徑。



SUGGESTIONS

水源阿伯可以紀錄每個點拖吊的車輛數以及大致的違停情況，如此把歷史資料放進我們設計的model時，能夠跑出更準確的答案。若是學生知道熱門地區容易會被拖吊後，漸漸都乖乖把車停在白線內，而使得該地的違停情況趨緩，水源阿伯也可以因此調整每個停車區的權重，讓拖車效益維持最大化，保持工作的效率。

關於路徑規劃的部份，若是未來有設置新的停車區，我們可以新增新的站點再把資料放進model執行，而每天行走不同的路徑，也能增加學生對於何時被拖吊的不確定性，進而願意多花一些時間找到空位好好停車，如此可以讓校園的交通秩序更加優良。