Universitat Politècnica de Catalunya

Treball de Fi de Grau en Enginyeria Informàtica

Especialitat en Ciències de la Computació

Mòdul d'assignació i routing d'un sistema de transport a la demanda

Gonzalo Recio Domènech

dirigit per

Dra. Mª Paz Linares Herreros

inLab FIB | Departament d'Estadística i Investigació Operativa

29 de gener de 2019

Resum

El Servei Públic de Transport Especial de Barcelona ofereix a les persones amb mobilitat reduïda la possibilitat de programar desplaçaments setmanals en taxi a una determinada hora. Actualment, les assignacions i rutes d'aquest servei es realitzen mitjançant un procediment manual i, per tant, sorgeix la necessitat d'optimitzar-les per a aprofitar millor els recursos destinats a aquest servei, fomentant la compartició de vehicle entre usuaris per tal de minimitzar el seu cost.

Ens trobem davant d'una generalització del problema d'optimització combinatòria conegut amb el nom de *Pickup and Delivery Problem with Time Windows* o *Dial-a-Ride Problem*, el qual consisteix en determinar una planificació òptima per a una flota de vehicles que ha de transportar usuaris respectant restriccions de temps i capacitat. En el nostre cas, el principal objectiu és minimitzar el nombre total de vehicles i, com a objectiu secundari, la distància total de viatge. Degut a la tipologia dels serveis, a més, s'han de considerar restriccions afegides com ara de flota de vehicles heterogènia. Aquest problema és NP-hard en la versió d'optimització, i utilitzar mètodes exactes per a escenaris de la vida real és computacionalment impracticable.

Així doncs, l'objectiu d'aquest Treball de Fi de Grau, que es realitza dins d'un projecte de l'inLab FIB, és desenvolupar un mòdul d'assignació i *routing* que s'integrarà en un nou sistema de gestió del Servei Públic de Transport Especial el qual contindrà un algorisme heurístic que determini bones solucions del problema properes a l'òptima en pocs segons d'execució.

Es presenta una metaheurística basada en una heurística constructiva inicial juntament amb una Cerca Tabú com a fase de millora per a resoldre el problema en temps assequible. Els resultats obtinguts mostren que la metaheurística implementada aconsegueix resultats pròxims a l'òptim amb temps d'execució raonables.

Resumen

El Servicio Público de Transporte Especial de Barcelona ofrece a personas con movilidad reducida la posibilidad de programar desplazamientos semanales en taxi a una determinada hora. Actualmente, las asignaciones y rutas de este servicio se realizan mediante un procedimiento manual y, por lo tanto, surge la necesidad de optimizarlas para aprovechar mejor los recursos destinadosa este servicio, fomentando la compartición de vehículo para así minimizar su coste.

Nos encontramos ante una generalización del problema de optimitzación combinatoria conocido con el nombre de *Pickup and Delivery Problem with Time Windows* o *Dial-a-Ride Problem*, el cual consiste en determinar una planificación óptima para una flota de vehúculos que tienen que transportar a usuarios respectando restricciones de tiempo y capacidad. En nuestro caso, el principal objetivo es minimizar el número total de vehículos y, como objectivo secundario, la distancia total de viaje. Debido a la tipologia de los servicios, además, se tienen que considerar restricciones adicionales como las de flota de vehículos heterogenea. Este problema es NP-hard en la versión de optimitzación, y utilizar métodos exactos para escenarios de la vida real es computacionalmente impracticable.

Así pues, el objectivo de este Trabajo de Fin de Grado, que se realiza dentro de un proyecto llevado a cabo por inLab FIB, es desarollar un módulo de asignación y *routing* que se integrará en un nuevo sistema de gestión del Servicio Público de Transporte Especial el cual contendrá un algoritmo heurístico que determine buenas soluciones del problema cercanas a la óptima en pocos segundos de ejecución.

Se presenta una metaheurística basada en una heurística constructiva inicial juntamente con una Busqueda Tabú como fase de mejora para resolver el problema en tiempo asequible. Los resultados obtenidos muestran que la metaheurística implementada consegue resultados próximos al óptimo con tiempos de ejecución razonables.

Abstract

The Public Service of Special Transport of Barcelona gives persons with reduced mobility the opportunity to program weekly taxi trips within the city at a specific hour. Currently, vehicle assignation and routing are planed manually, therefore, the need to optimize them arises in order to take more advantage of the allocated resources and increase car-sharing to minimize the service cost.

We are facing a generalization of the well-known combinatorial optimization problem called *Pickup and Delivery Problem with Time Windows* or *Dial-a-Ride Problem*, which consists of determining an optimal schedule for a fleet of vehicles that have to transport costumers considering time and capacity constraints. In our case, the main goal is to minimize the fleet of vehicles, and as a secondary objective, the total travelled distance. Due to the service typology, it has to be taken into account additional restrictions such as considering heterogeneous fleet. This problem is NP-Hard in the optimization version, and exact approaches for real-life scenarios are computationally not practicable.

Thus, the goal of this Final Degree Thesis, which is a part of a project carried out by inLab FIB, is to develop an assignation and routing module that is going to be integrated in a new management system of the Public Service of Special Transportation, which will contain an heuristic algorithm that provides good solutions of the problem near to optimum in a few seconds of execution time.

A metaheuristic based on an initial constructive heuristic together with a Tabu Search for improvement purpose is presented for solving this problem. The obtained results show that the implemented metaheuristic achieves very good solutions not far from the optimum in a reasonable execution time.

Agraïments

Primer de tot, m'agradaria agrair a la M^{a}Paz Linares per l'excel · lent supervisió d'aquest treball. No només m'ha ajudat en moments de dificultat tècnica, sinó que s'ha implicat totalment en el projecte dedicant temps i esforços per a millorar aquest Treball de Fi de Grau fins a la seva millor versió possible. Sense el seu bon criteri i guiatge al llarg de tot el projecte, aquest treball no seria el que és ara. Sincerament, és una gran persona amb talent descomunal amb qui realment m'agradaria seguir treballant i conservar en l'entorn personal.

En segon lloc, a l'equip i membres de l'inLab amb els quals he treballat colze a colze en el projecte i que m'han donat suport, ajuda i ànims. En especial, al Sergio Walls, Juanjo, Gisela, Josefran, Juan, Gina i Josep, per fer tota aquesta etapa tant amena i inoblidable.

També a la família i amics íntims per tot el suport i en concret als meus pares per les oportunitats que m'han donat i el recolzament incondicional que han tingut cap a mi.

No podia faltar en Josep Casanovas i la Marta Cuatrecasas per donar-me l'oportunitat d'entrar a l'inLab i poder aprendre tant d'aquest magnífic entorn de treball, del qual sempre quedarà com una de les meves millors experiències.

Índex

Índex de taules

Índex de figures

Llista d'Algorismes

Glossari

AMB Àrea Metropolitana de Barcelona.

AMPL A Mathematical Programming Language.

Angular *Framework* de *Typescript* utilitzat per a desenvolupar el *Front-end* d'aplicacions *web*.

Back-end Part d'un sistema *software* que s'encarrega de la lògica i es comunica amb el servidor i capa de persistència.

CVRP Capacitated Vehicle Routing Problem.

DARP Dial-a-Ride Problem.

Django *Framework* de *Python* utilitzat per a desenvolupar aplicacions web.

FIB Facultat d'Informàtica de Barcelona.

Front-end Part del *software* amb la que interactuen els usuaris.

HVRP Heterogeneous Vehicle Routing Problem.

HVRPPDTW Heterogeneous Vehicle Routing Problemwith Pickup and Delivery with Time-Windows.

ILP Programació Lineal Entera (en anglès, Integer Linear Programming).

IMPD Institut Municipal de Persones amb Discapacitat.

KPI Key Performance Indicator.

LP Programació Lineal (en anglès, Linear Programming).

PDPTW Pickup and Delivery Problem with Time-Windows.

PMR Persona amb Mobilitat Reduïda.

Routing Relatiu a la planificació de rutes en una xarxa de transport.

TDD Test-Driven-Development.

VRP Vehicle Routing Problem.

VRPPD Vehicle Routing Problem with Pick-Up and Delivery.

VRPPDTW Vehicle Routing Problem with Pickup and Delivery with Time-Windows.

VRPTW Vehicle Routing Problem with Time-Windows.

Capítol 1

Introducció

Aquest Treball de Fi de Grau en Enginyeria Informàtica (d'especialitat en Computació) de la Facultat d'Informàtica de Barcelona ha estat desenvolupat en la modalitat B amb un conveni de Cooperació Educativa a l'`inLab FIB` i dirigit per la Dra. Mª Paz Linares Herreros del Departament d'Estadística i Investigació Operativa.

L'objectiu principal d'aquest Treball de Fi de Grau és optimitzar les rutes d'un servei de transport a la demanda. Això implica la formalització del problema, l'estudi de mètodes actuals per a resoldre'l, la implementació d'un o més algorismes i la integració en el sistema més complet.

1.1 Context i motivació

El *Servei Municipal de Transport Especial* de Barcelona és un servei públic de l'Institut Municipal de Persones amb Discapacitat (`IMPD`) i l'Àrea Metropolitana de Barcelona (`AMB`) complementari al transport públic i pensat per ser utilitzat únicament quan no hi ha transport públic regular adaptat. Aquest servei poden demanar-lo les persones amb mobilitat reduïda (PMR), i ofereix la possibilitat de realitzar viatges entre els municipis de Barcelona, Badalona, Esplugues de Llobregat, l'Hospitalet de Llobregat, Sant Adrià de Besòs i Santa Coloma de Gramenet. El servei està disponible tots els dies de l'any i disposa d'una flota integrada d'autobusos i taxis [1].

Servei Públic de Transport Especial

Els usuaris que posseeixin la Targeta de Transport Especial poden demanar serveis porta a porta de transport especial per a una hora determinada que desitgin. Els tipus de serveis que es presten són els següents:

- **Serveis Fixos**: serveis que es repeteixen setmanalment (en hores, orígens i destinacions) i que es poden programar fàcilment amb antelació a l'inici de temporada. Es sol · liciten i programen les rutes anualment quan comença la temporada (principis de setembre).

- **Serveis Esporàdics**: serveis que no es repeteixen ni en hores ni en orígens ni en destinacions i no es poden programar fàcilment amb molta antelació. Es demanen sobre la marxa amb una antelació màxima d'uns dos dies.

A nivell operatiu, aquests serveis es subcontracten a una operadora de vehicles (busos i taxis) la qual cobra el kilometratge de cada viatge a l'IMPD.

Sistema actual

Un cop introduït el servei que ofereixen l'IMPD i l'AMB, explicarem com es gestiona avui en dia.

Els serveis **esporàdics** es tramiten amb un sistema que té enregistrat als usuaris PMR i els assigna a vehicles de l'operadora contractada per a realitzar els viatges. Actualment, aquest procés d'assignació és manual: se sol · liciten els serveis esporàdics per via telefònica i els gestors d'AMB del sistema s'encarreguen d'assignar-los, la qual cosa genera una alta saturació de les línies telefòniques a certes hores del dia.

Pel que fa als serveis **fixos**, la planificació també és realitza manualment. L'IMPD envia les sol · licituds dels serveis fixos a l'operadora i aquesta proposa una planificació anual de rutes calculada a mà. Aquesta planificació sembla ser poc eficient a dia d'avui, amb la qual cosa l'IMPD considera que pot estar pagant de més a l'operadora pel fet de no generar unes rutes optimitzades i, per tant, creuen que podrien arribar a donar servei a més usuaris amb el mateix pressupost.

Amb l'objectiu de millorar la gestió actual, l'AMB i l'IMPD s'han posat en contacte amb el laboratori de recerca i innovació inLab FIB per tal de crear un nou sistema informàtic que millori l'eficiència i disminueixi costos del servei. A continuació s'enumeren les principals millores i objectius que l'IMPD i l'AMB volen conseguir amb la renovació del sistema:

- Crear un nou *software* Web App per a usuaris PMR i gestors AMB a través del qual poder sol · licitar serveis esporàdics (i així alliberar la congestió telefònica).
- Permetre que el sistema admeti la participació de més d'una empresa operadora de taxis (actualment està monopolitzat en una única operadora).
- Realitzar una assignació dels serveis **esporàdics** automàtica i equitativa entre les operadores en funció de la flota.
- Realitzar una assignació dels serveis **fixos** automàtica que optimitzi les rutes i maximitzi la compartició de vehicles per tal de no desaprofitar recursos.

1.2 Stakeholders

Un cop entrat en context, les parts implicades i afectades pel sistema que es desenvoluparà són les següents.

Institut Municipal de Persones amb Discapacitat

L'IMPD és un organisme autònom a través del qual l'Ajuntament de Barcelona treballa per garantir els drets socials de les persones amb diversitat funcional. Els serveis de l'IMPD ofereixen, d'una banda, recursos per a la millora de la qualitat de vida de les persones i d'una altra, contribuir a la cerca de noves solucions i al suport a la resta de l'organització municipal per tal d'aconseguir una ciutat més accessible i inclusiva [2].

L'IMPD vol renovar el sistema que s'utilitza per a la gestió del *Servei Públic de Transport Especial*, amb l'objectiu de millorar-lo. L'IMPD és un dels principals impulsors del projecte.

Àrea Metropolitana de Barcelona

L'AMB és l'administració pública del territori metropolità de Barcelona i s'encarrega d'operar el sistema de Servei de Transport Públic Especial. Posa a disposició uns gestors atenent sol · licituds telefòniques i el nou *software* que es desenvoluparà pretén millorar la gestió actual. Juntament amb l'IMPD, imposa regles de negoci i finança part del projecte.

Operadores

Són les empreses contractades per l'IMPD que ofereixen part de la seva flota de vehicles per al *Servei Públic de Transport Especial*. Actualment només es treballa amb una única operadora, però amb el nou sistema podran entrar a participar noves empreses i autònoms. Amb el nou sistema, deixaran de ser elles qui proposen les rutes calculades a mà per tal de cobrir els serveis fixos.

Persones amb Mobilitat Reduïda

Són els usuaris que utilitzaran el sistema i es beneficiaran de les millores que el producte final aportarà, com per exemple, poder sol · licitar serveis mitjançant la nova Web App sense haver de trucar.

inLab FIB

L'inLab FIB és el laboratori d'innovació i recerca de la *Facultat d'Informàtica de Barcelona* (UPC) que s'encarrega de desenvolupar el nou sistema, del qual, a més, en serà propietari.

Dins de l'inLab FIB hi ha l'equip de desenvolupament, el Product-Owner i una persona experta en temes de mobilitat i transport.

- **Equip de desenvolupament**: està format per 5-6 persones, i és l'encarregat de desenvolupar i documentar la part tècnica del nou sistema. Dins l'equip

de desenvolupament, s'hi troba l'autor d'aquest Treball de Fi de Grau, que dedicarà especialment més temps a la part algorísmica.

- **Product-Owner**: és l'encarregat de gestionar la part de negoci amb el client i transmetre-la a l'equip tècnic, així com fixar uns objectius i una planificació temporal del projecte. L'actor que desempenya aquest rol és en Jordi Montero, professor de la UPC i col · laborador a l'`inLab FIB`.
- **Expert en mobilitat i transport (Routing)**: La persona experta en mobilitat i transport és la que s'encarregarà de liderar i aportar el seu coneixement a la part del mòdul d'assignació i *routing* del sistema. La persona que realitza aquest rol és la Dra. Mª Paz Linares Herreros, professora de la UPC i col · laboradora a l'`inLab FIB`, que, a més, és la directora i ponent d'aquest Treball de Fi de Grau.

1.3 Formulació del problema

A la introducció s'han presentat els dos tipus de serveis que ofereix el *Servei Públic de Transport Especial*. El nostre problema se centrarà principalment en els **serveis fixos**, els quals es programen a l'inici de *temporada* a partir d'una demanda fixa que es repetirà setmanalment. Amb aquesta informació el nou sistema assignarà automàticament els serveis fixos als diferents vehicles (microbusos compartits i taxis) de les operadores per a cobrir els desplaçaments setmanals dels usuaris.

L'objectiu del problema és trobar una assignació que maximitzi la demanda a la que es pot arribar a donar servei amb el mateix pressupost fixat per a la contractació de la flota.

Aquesta assignació té en compte restriccions a nivell d'usuari (els orígens i destins, si requereixen de vehicle adaptat o no, finestres de temps de sortida i/o arribada del viatge.) i restriccions a nivell de flota (les característiques dels vehicles, capacitats i la dimensió de la flota definida per les operadores).

Aquest és un problema d'optimització conegut amb el nom de *problema d'enrutament de vehicles*, en anglès, *Vehicle Routing Problem* (`VRP`), subjecte, però, a algunes variacions que el fan bastant més complex.

1.3.1 Vehicle Routing Problem

La pregunta que resol el *Vehicle Routing Problem* (`VRP`) és la següent: "Quin és el conjunt de rutes òptimes per a una flota de vehicles per tal de repartir béns a un conjunt donat de clients?"

La versió simple del `VPR` té en compte que els clients són punts de demanda fixa, que la flota de vehicles és homogènia (mateixes capacitats) i que els vehicles parteixen d'un mateix punt. L'objectiu és minimitzar el cost total de repartir els béns a tots els clients.

En aquest cas, una solució és factible si la quantitat de clients assignats a cada ruta no supera la capacitat dels vehicles.

El cas real al que s'enfronten l'`IMPD` i l'`AMB` té unes implicacions afegides que podem trobar a la literatura com a variants del `VRP` per separat.

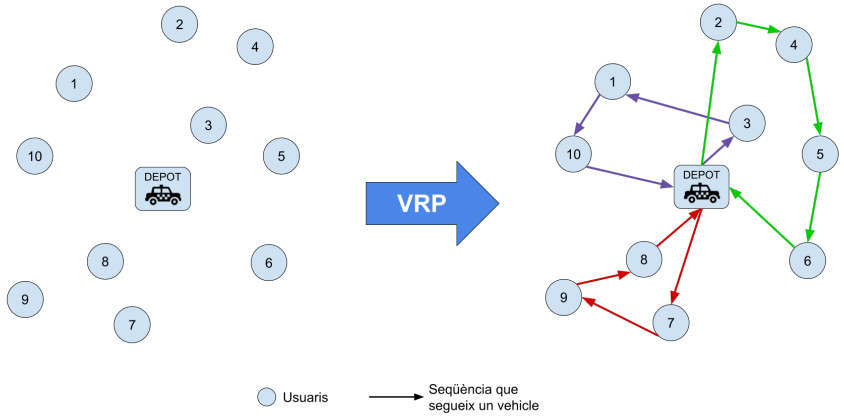

Figura 1.1: Figura que il·lustra el Vehicle Routing Problem.

1.3.2 Variants del VRP

- **Vehicle Routing Problem with Pick-Up and Delivery (VRPPD)**

  Una important restricció afegida al problema que apareix amb els serveis fixos és respectar l'ordre de recorregut dels orígens i destins dels usuaris. Primer s'haurà de passar a buscar un usuari al seu punt d'origen i després portar-lo al seu destí, mai al revés.

  Per tant, s'hauran d'afegir aquestes restriccions de precedència entre vèrtexs del graf al problema bàsic del `VRP`.

- **Vehicle Routing Problem with Time-Windows (VRPTW)**

  Els usuaris trien una hora a la que volen ser recollits o lliurats a un destí. Per a complir aquestes hores de sortida i d'arribada dels serveis dels usuaris PMR, se sol d'afegir una finestra de temps ($\pm\delta$) per a relaxar aquesta la restricció horària, ja que sinó podríem arribar a no obtenir solucions factibles. Com més ample sigui la finestra de temps, millors resultats es poden obtenir ja que es té més marge per a crear possibles noves rutes, però a la vegada pot empitjorar la qualitat del servei pel que fa al temps de viatge.

- **Heterogeneous Vehicle Routing Problem (HVRP)**

  Per al servei fix es compta amb dos tipus de vehicles: els microbusos i els taxis. Tots dos vehicles poden ser compartits pels usuaris (viatjar més d'un usuari a la vegada), però això afegeix complexitat al problema, degut a que són vehicles que tenen diferent punt de partida, deferents capacitats, costos, velocitats, etc. A més ens trobem amb taxis adaptats i no adaptats, hi alguns dels usuris només

poden viatjar en vehicles que estiguin adaptats. Aquestes tipologies de la flota de taxis s'ha de tenir en compte també.

Aquestes restriccions addicionals de les variants comentades s'han d'afegir al problema VRP que es vol resoldre.

1.3.3 Formulació final: Heterogeneous Vehicle Routing Problem with Pickup and Delivery with Time Windows

Posant totes les restriccions addicionals descrites a la secció 1.3.2, la variant del VRP a la que ens enfrontem és un *Heterogeneous Vehicle Routing Problem with Pickup and Delivery with Time Windows* (HVRPPDTW). Aquest problema també rep el nom de Pickup and Delivery Problem with Time-Windows (PDPTW) o Dial-a-Ride Problem (DARP).

La formulació matemàtica es veurà i descriurà detalladament al capítol 2 ja que és una part complexa i de pes en aquest treball.

1.4 Objectius

Amb la necessitat de millorar l'eficiència i reduir costos del servei fix per part de l'IMPD i l'AMB, l'objectiu principal d'aquest projecte és optimitzar l'assignació i les rutes dels serveis fixos, els quals ara mateix es planifiquen a mà per part de l'operadora de vehicles.

A continuació s'han definit una sèrie de fites les quals, si s'assoleixen, s'haurà complert amb l'objectiu principal.

- Descriure la formulació matemàtica del problema per al cas específic dels serveis fixos que ofereix el *Servei Públic de Transport Especial*.
- Realitzar un estudi de la complexitat algorísmica del problema per a decidir quin mètode és més adient utilitzar per a resoldre'l.
- Desenvolupar un algorisme que resolgui el problema.
- Validar i avaluar els resultats de l'algorisme.
- Realitzar una proposta d'integració de l'algorisme al sistema complet.

1.5 Abast

El sistema que es desenvoluparà a l'inLab FIB implica el desenvolupament d'un *software* nou i el disseny d'una nova lògica de negoci del qual una part serà un mòdul d'assignació i *routing* per als serveis fixos. Aquest mòdul és el que es desenvoluparà en aquest Treball de Fi de Grau.

Així doncs, aquest Treball de Fi de Grau ve delimitat per la part algorísmica del sistema relacionada amb l'optimització de les rutes per als serveis fixos que ofereix el *Servei Públic de Transport Especial*. L'estudi del problema, formalitzar-lo matemàticament, resoldre'l, validar els resultats i facilitar la integració de l'algorisme al sistema entren dins l'abast d'aquest projecte.

1.5.1 Obstacles

Durant el procés de desenvolupament d'un projecte solen aparèixer obstacles que endarrereixen la planificació inicialment establerta i que poden impedir el compliment de tots els objectius.

Dependència amb la resta del sistema

El mòdul a desenvolupar que s'encarrega de la part algorísmica, es troba subordinat a la resta del sistema, ja que s'haurà d'integrar amb ell a posteriori. El mal funcionament de la resta de components amb els que s'ha d'integrar podria ser un greu obstacle per a la realització completa d'aquest Treball Final de Grau.

Limitacions temporals

El projecte té un seguit de dates límit per a cada fase consensuades amb el client que s'han de complir amb rigorositat. Una mala planificació temporal pot ocasionar el no assoliment amb plenitud de tots els objectius.

Modificació dels objectius del client

Aquest projecte compta amb més d'un client, l'`IMPD` i l'`AMB`. En algun moment del desenvolupament pot succeir que no tinguin els objectius clars o alineats i, a arrel d'això, s'originin modificacions dels objectius inicials del projecte.

Altres rols dins del projecte

Com a integrant de l'equip de desenvolupament, també assumeixo altres rols programant components del nou *software* que ocuparan gran part del meu temps dins el projecte. Tot i així, tota tasca més enllà de la part algorísmica, com ara el desenvolupament software, quedarà fora de l'abast d'aquest Treball de Fi de Grau.

1.6 Metodologia

Per a desenvolupar el projecte, dins l'equip de desenvolupament s'utilitzaran metodologies àgils de cicles curts (entre 2 i 3 setmanes) basades en Scrum [3]. Aquest

desenvolupament iteratiu ens permetrà definir objectius a cada cicle, tenir més flexibilitat i un desenvolupament més ràpid. A més, es mantindrà un continu contacte amb el client el qual ens proporcionarà un *feedback* que es podrà integrar a cada iteració.

Per tal de validar i contrastar la correctesa de l'algorisme a desenvolupar, es realitzaran experiments amb diferents instàncies del problema i configuració de paràmetres i s'extrauran indicadors de la qualitat de les solucions així com també d'eficiència computacional.

Pel que fa a la validació de la integració amb la resta del sistema, s'aplicaran metodologies de *testing* basades en TDD (*Test Driven Development*) [4] per assegurar el correcte funcionament del mòdul.

L'algorisme que utilitzi el mòdul es validarà amb instàncies les dimensions de les quals permetin calcular l'òptim de manera exacta amb un *solver* de programació lineal entera. D'aquesta manera, es podrà realitzar un anàlisi comparatiu per a contrastar la qualitat de les solucions de l'algorisme implementat i observar quant s'allunya del resultat òptim.

Per a reforçar la validació, dins l'equip del projecte està la directora d'aquest treball, experta en temes d'optimització, que supervisarà la feina desenvolupada a cada fase del projecte.

Capítol 2

Estudi del problema

En aquest capítol es descriu la formalització matemàtica del problema que s'ha de resoldre. Primer es presenta el *Capacitated* VRP, que és la versió més simple del problema, i després es descriuen el VRP *with Time Windows* i el VRP *with Pickup and Delivery with Time Windows*, el qual, aquest últim, s'assembla al problema dels serveis fixos d'`IMPD`. Seguidament es realitza un estat de l'art sobre com es resol actualment el problema i finalment es conclou amb la selecció d'un mètode per a començar a implementar una solució.

2.1 Formalització matemàtica del problema

En aquesta secció estudiarem la formalització matemàtica del `VRP` i les seves versions esteses, ampliant-lo fins al `VRPTW` i el `VRPPDTW`. Finalment, es presentarà la formalització final adaptada al cas dels serveis fixos amb les restriccions i modificacions afegides que demana el client (`IMPD` i `AMB`).

Notació general

En totes les formalitzacions del `VRP` i demés variants hi ha una notació i elements comuns. Aquest apartat pretén ser un glossari on definir aquests conceptes i facilitarne la consulta.

El `VRP` i les seves variants es poden descriure com un problema de grafs [5]: Sigui $G = (V, A)$ un graf complet, amb $V = \{0, \ldots, n\}$ com a conjunt de vèrtexs i A com a conjunt d'arestes. Els vèrtexs $i = 1, \ldots, n$ representen els usuaris i el vèrtex 0 el punt de partida dels K vehicles (en el nostre cas pot ser un punt d'origen fictici).

També tenim una matriu de costos on cada element $c_{ij} \geq 0$ representa el cost (en el nostre cas la distància de viatge) associat a l'aresta $(i, j) \in A$.

- $G = (V, A)$, és el graf amb el que es representa la xarxa de transport. V és el conjunt punts on es troben els usuaris (coordenades) i A el conjunt de possibles arestes entre els nodes. Un arc (i, j) representa que existeix un camí entre

el node i i j. En el nostre cas, el graf amb el que treballarem serà **dirigit** i **complet**.

- *depot*: node de sortida i/o arribada dels vehicles (poden ser ficticis). Excepte en el cas de que s'especifiqui el contrari, el *depot* de partida es veurà representat pel node 0 i el d'arribada pel node $n+1$ (tot i que pot haver-hi només un).
- $N = V \setminus \{0, n+1\}$
- d_i: demanda del node $i \in V$
- c_{ij}: cost de viatge (distància) de l'arc $(i,j) \in A$.
- t_{ij}: temps de viatge de l'arc $(i,j) \in A$.
- K: nombre de vehicles
- C: capacitat dels vehicles. En cas de tenir una flota heterogènia, C_k representa la capacitat del vehicle $k \in K$.
- $\Delta^+(i)$: conjunt de vèrtexs que surten del node i.
- $\Delta^-(i)$: conjunt de vèrtexs que entren al node i.

Tot i així, a cada apartat apareixerà nova notació que s'explicarà més detalladament quan s'hagi d'utilitzar en alguna de les variants del `VRP`.

2.1.1 CVRP

La versió bàsica del `VRP` també coneguda com a *Capacitated Vehicle Routing Problem* (`CVRP`) consisteix en un conjunt de nodes que són de recollida o lliurament (al node i se li assigna una demanda d_i). Els vehicles són idèntics, surten i tornen d'un mateix *depot* i només es contemplen restriccions de capacitat. L'objectiu del `CVRP` és trobar una col · lecció d'exactament K circuits simples (cada un corresponent a una ruta de vehicle) de cost mínim. El cost de les rutes pot ser, per exemple, una funció ponderada sobre el nombre de rutes i la seva distància de recorregut o temps de viatge. Aquestes rutes han de complir que cada vehicles visiti el *depot*, cada usuari sigui visitat per un únic vehicle i que es respecti la capacitat del vehicle al llarg de cada circuit.

Aquest problema es pot formular com un problema de programació lineal entera tal i com es mostra a continuació.

Notació

Sigui $S \subseteq V \setminus \{0\}$, definim la funció $r(S)$ com el mínim nombre de vehicles necessaris per a cobrir la demanda S. Aquest valor és el resultat de resoldre el *Bin Packing*

Problem (BPP)[1], problema NP-Hard, però es pot substituir per la cota **mínima** trivial $\lceil \frac{d(S)}{C} \rceil$, on $d(S) = \sum_{i \in S} d_i$.

Formulació

Amb la notació anterior, la formulació lineal entera proposada per [5] és la següent:

$$\text{(CVRP)} \qquad \min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \tag{2.1}$$

subjecte a:

$$\sum_{i \in V} x_{ij} = 1 \qquad \forall j \in V \setminus \{0\} \tag{2.2}$$

$$\sum_{j \in V} x_{ij} = 1 \qquad \forall i \in V \setminus \{0\} \tag{2.3}$$

$$\sum_{i \in V} x_{i0} = K \tag{2.4}$$

$$\sum_{j \in V} x_{0j} = K \tag{2.5}$$

$$\sum_{i \notin S} \sum_{j \in S} x_{ij} \geq r(S) \qquad \forall S \subseteq V \setminus \{0\},\ S \neq \emptyset \tag{2.6}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in V \tag{2.7}$$

on la variable indicadora $x_{ij} = \begin{cases} 1 & \text{si l'arc (i,j) està a la solució} \\ 0 & \text{altrament} \end{cases}$

Les restriccions (2) i (3) indiquen que un arc l'ha de recórrer exactament un vehicle. Les restriccions (4) i (5) indiquen que el nombre de vehicles que surten del punt inicial és el mateix que els que tornen i que és igual a K. La restricció (6) imposa que les rutes han d'estar connectades i que la demanda a cada ruta no pot excedir la capacitat del vehicle, on $r(S)$ és el mínim nombre de vehicles necessaris per a servir S. Per acabar, la restricció (7) imposa que les variables siguin binàries.

La formulació anterior és un model de flux de vehicles de 2 índexs (*Two-index vehicle flow models*). Aquesta modelització no permet saber quin vehicle recorre cada arc de la solució obtinguda i quan es complica el problema convé treballar amb variables de tres índexs (*three-index vehicle flow formulation*, x_{ijk}) que ens donen informació sobre quin vehicle k travessa l'aresta $(i, j) \in A$ a la solució.

De moment no entrarem a fer un anàlisi en profunditat de com resoldre aquest problema, ja que és una versió molt simple del problema que finalment s'acabarà tractant.

Tot i així es veu que aquest model d'optimització és costós de resoldre mitjançant relaxació de programació lineal entera degut a les restriccions del tipus (6), la qual és exponencial al nombre de vèrtexs[2].

[1] Bin Packing Problem, problema de combinatòria NP-Hard que consisteix en encabir un conjunt d'objectes de diferents volums en el mínim nombre de contenidors de volum fixat

[2] El nombre total de subconjunts d'un conjunt de mida n és 2^n.

2.1.2 VRPTW

El VRP *with Time Windows* és una extensió del `CVRP` en el que s'imposen restriccions de capacitat i s'associa a cada usuari i un interval de temps $[a_i, b_i]$, anomenat finestra temporal (*time-window*). El servei de cada usuari ha de començar dins d'aquesta finestra temporal associada i els vehicles s'aturen durant un temps de servei (s_i) atenent la petició. El `VRPTW` consisteix en trobar una col · lecció d'exactament K circuits amb cost mínim, tals que que cada vehicle visita el *depot*, tot usuari es servit exactament per un vehicle i que es compleixin les restriccions de temps i capacitat.

Notació

Per a aquesta variant haurem d'afegir notació complementaria necessària per a tenir control sobre les restriccions temporals. Partirem de la notació de la secció anterior, on la xarxa de transport és un graf $G = (V, A)$ i el punt inicial (*depot*) es representa pels nodes 0 i $n + 1$. Com ja s'ha comentat, per a cada node $i \in V$ hi ha associada una finestra de temps $[a_i, b_i]$ i, en concret, $[a_0, b_0] = [a_{n+1}, b_{n+1}] = [E, L]$, on E i L representen l'instant més d'hora possible de sortida del *depot* i l'instant d'arribada al *depot* més tard possible, respectivament. A més, per cada $i \in V$ tenim associat un temps de servei s_i que fa referència al temps que el vehicle triga en recollir (o lliurar) un usuari.

De la mateixa manera que al `CVRP`, definim demandes i temps de servei nuls al punt inicial de partida i punt final d'arribada, per tant, $d_0 = d_{n+1} = s_0 = s_{n+1} = 0$.

Degut a aquestes restriccions temporals, es poden eliminar arcs $(i, j) \in A$ si $a_i + s_i + t_{ij} > b_j$, és a dir, en el cas que sigui inviable en termes de temps arribar als dos nodes tenint en comptes les finestres temporals. També poden eliminar-se per limitacions de capacitat si $d_i + d_j > C$. Per aquest motiu treballem amb la notació Δ^+ i Δ^-, explicada al principi de la secció 2.1.

En aquesta variant del problema, a més de la matriu de costos (c_{ij}), també s'utilitza la matriu de temps de viatge en la que els elements $t_{ij} \geq 0$ representen el temps que triga un vehicle en recórrer l'aresta $(i, j) \in A_k$.

Finalment, si es vol permetre que alguns vehicles es quedin al *depot*, en el cas que vulguem minimitzar el nombre de vehicles a utilitzar, l'arc $(0, n + 1)$ s'ha d'afegir al conjunt A amb costos $c_{0,n+1} = t_{0,n+1} = 0$.

Formulació

Amb la notació anterior, una formulació lineal entera del `VRPTW` proposada per [5] és la següent:

$$\text{(VRPTW)} \qquad \min \sum_{k \in K} \sum_{(i,j) \in A} c_{ij} x_{ijk} \tag{2.1}$$

subjecte a:

$$\sum_{k\in K}\sum_{j\in\Delta^+(i)} x_{ijk} = 1 \qquad \forall i \in N \tag{2.2}$$

$$\sum_{j\in\Delta^+(0)} x_{0jk} = 1 \qquad \forall k \in K \tag{2.3}$$

$$\sum_{i\in\Delta^-(j)} x_{ijk} - \sum_{i\in\Delta^+(j)} x_{ijk} = 0 \qquad \forall k \in K,\, j \in N \tag{2.4}$$

$$\sum_{i\in\Delta^-(n+1)} x_{i,n+1,k} = 1 \qquad \forall k \in K \tag{2.5}$$

$$x_{ijk}(w_{ik} + s_i + t_{ij} - w_j k) \leq 0 \qquad \forall k \in K,\, (i,j) \in A \tag{2.6}$$

$$a_i \sum_{j\in\Delta^+(i)} x_{ijk} \leq w_{ik} \leq b_i \sum_{j\in\Delta^+(i)} x_{ijk} \qquad \forall k \in K, i \in N \tag{2.7}$$

$$E \leq w_{ik} \leq L \qquad \forall k \in K, i \in \{0, n+1\} \tag{2.8}$$

$$\sum_{i\in V} d_i \sum_{j\in\Delta^+(i)} x_{ijk} \leq C \qquad \forall k \in K \tag{2.9}$$

$$x_{ijk} \in \{0,1\} \qquad \forall k \in K, (i,j) \in A \tag{2.10}$$

on la variable $x_{ijk} = \begin{cases} 1 & \text{si l'arc } (i,j) \text{ el travessa el vehicle } k \text{ i està a la solució} \\ 0 & \text{altrament} \end{cases}$

i la variable w_{ik}, $i \in V$, $k \in K$, especifica l'inici del servei al node i quan és servit pel vehicle k.

La funció objectiu (1) representa el cost total de les rutes.

Donat que $N = V \setminus \{0, n+1\}$ és el nombre de clients, les restriccions del tipus (2) imposen que cada client només s'assigni a una única ruta de vehicle.

Les restriccions (3)-(5), imposen que es compleixi la conservació de flux (que tot el que entra a un node, ha de sortir), i obliga a que tots els vehicles surtin del *depot* i finalitzin la ruta també al *depot*.

Les restriccions (6)-(8) fan referència a les restriccions temporals (*time-windows*). Així mateix, la (9) procura que no es sobrepassin les capacitats dels vehicles.

Finalment, la condició (10) imposa que les variables x_{ijk} siguin binàries.

Convé remarcar que hi ha una restricció no lineal, en contret, la número (6). Tot i així, es pot linealitzar reescrivint la restricció per si es volgués resoldre amb un *solver* de programació lineal entera o simplement per facilitar la resolució d'aquest. Aquest procediment està explicat a la secció 4.2.1.

2.1.3 VRPPDTW

En el `VRP` *with Pickup and Delivery with Time Windows* (`VRPPDTW`), es considera una flota heterogènia de vehicles que des de diferents terminals han de transportar un conjunt de càrregues des d'origens a destins, respectant les restriccions de capacitat i de temps. Per tant, cada petició es defineix amb un punt de recollida (*pickup*), un de

deixada (*delivery*) corresponent i la demanda que s'ha de transportar entre aquests punts. Aquestes parelles de nodes de *pickup* i *delivery* se solen anomenar *PD-pairs*. Addicionalment, els nodes tenen associada una finestra temporal que s'ha de complir per a recollir i lliurar les demandes respectives.

El `VRPPDTW` és una generalització del `VRPTW` explicat en la secció anterior (2.1.2), en el que tots els *pickups* (o bé els *deliveries*) són al mateix *depot*.

Aquesta versió del problema implica visitar cada punt de *pickup* i *delivery* exactament un cop, sense excedir la capacitat dels vehicles i assegurar que cada punt de *pickup* i de *delivery* corresponents estiguin a la mateixa ruta de vehicle tot imposant la precedència entre ells (primer passar pel *pickup* i després pel *delivery*).

Notació

En aquesta versió, es redefineix part de la notació explicada fins ara. En el problema es tenen n peticions. Sigui un petició i, la identificarem amb dos nodes, i i $n+i$, corresponents al node de *pickup* i al de *delivery*, respectivament (és possible que dos nodes representin el mateix punt geogràfic). A continuació, definim el conjunt de nodes de *pickup* $P = \{1, \ldots, n\}$ i el conjunt de nodes de *delivery* $D = \{n+1, \ldots, 2n\}$. D'aquesta manera, definim $N = P \cup D$. Si la petició i consisteix en transportar d_i persones des de i fins a $n+i$, definim $l_i = d_i$ i $l_{n+i} = -d_i$, com la càrrega (*load*) que porta o deixa el vehicle.

Sigui K el conjunt de vehicles, com que pot donar-se el cas en que no tots els vehicles puguin servir totes les peticions, cada vehicle k té un conjunt específic associat $N_k = P_k \cup D_k$, on N_k, P_k i D_k són subconjunts apropiats de N, P i D, respectivament.

Per cada vehicle k, es defineix un graf $G_k = (V_k, A_k)$. Definim $V_k = N_k \cup \{o(k), d(k)\}$, on $o(k)$ és el *depot* d'origen del vehicle k i $d(k)$ és *depot* de destí corresponent.

Formulació

Per a la formulació d'aquesta versió, s'utilitzen dues noves noves variables a part de les indicadores de flux x_{ijk}: les de temps T_{ik} que especifiquen en quin instant el vehicle k comença el servei al node i, i les variables de càrrega L_{ik} que indiquen la càrrega del vehicle k després de servir la petició al node $i \in V_k$.

A continuació es presenta la formulació matemàtica del `VRPPDTW` proposada per [5], afegint les noves restriccions tenint en compte les del `VRPTW` també:

$$\text{(VRPPDTW)} \qquad \min \sum_{k \in K} \sum_{(i,j) \in A_k} c_{ijk} x_{ijk} \tag{2.1}$$

subjecte a:

$$\sum_{k \in K} \sum_{j \in N_k \cup \{d(k)\}} x_{ijk} = 1 \qquad \forall i \in N \tag{2.2}$$

$$\sum_{j \in N_k} x_{ijk} - \sum_{j \in N_k} x_{j,n+i,k} = 0 \qquad \forall k \in K,\, i \in P_k \tag{2.3}$$

$$\sum_{j \in P_k \cup \{o(k)\}} x_{o(k)jk} = 1 \qquad \forall k \in K \tag{2.4}$$

$$\sum_{i \in N_k \cup \{o(k)\}} x_{ijk} - \sum_{i \in N_k \cup \{d(k)\}} x_{jik} = 0 \qquad \forall k \in K,\, j \in N_k \tag{2.5}$$

$$\sum_{i \in D_k \cup \{o(k)\} n+1)} x_{i,d(k),k} = 1 \qquad \forall k \in K \tag{2.6}$$

$$x_{ijk}(T_{ik} + s_i + t_{ij} - T_{jk}) \leq 0 \qquad \forall k \in K,\, (i,j) \in A_k \tag{2.7}$$

$$a_i \leq T_{ik} \leq b_i \qquad \forall k \in K, i \in V_k \tag{2.8}$$

$$T_{ik} + t_{i,n+1,k} \leq T_{n+1,k} \qquad \forall k \in K, i \in P_k \tag{2.9}$$

$$x_{ijk}(L_{ik} + l_j - L_{jk}) = 0 \qquad \forall k \in K, (i,j) \in A_k \tag{2.10}$$

$$l_i \leq L \leq C_k \qquad \forall k \in K, i \in P_k \tag{2.11}$$

$$0 \leq L_{n+1,k} \leq C_k + l_i \qquad k \in K, n+i \in D_k \tag{2.12}$$

$$L_{o(k),k} = 0 \qquad \forall k \in K \tag{2.13}$$

$$x_{ijk} \in \{0,1\} \qquad \forall k \in K, (i,j) \in A_k \tag{2.14}$$

on la variable $x_{ijk} = \begin{cases} 1 & \text{si l'arc } (i,j) \text{ el travessa el vehicle } k \text{ i està a la solució} \\ 0 & \text{altrament} \end{cases}$

La funció objectiu lineal (1) minimitza el cost de les rutes.

Les restriccions de (2) i (3) imposen que cada petició (els nodes de *pickup* i de *delivery*) es servida un únic cop i pel mateix vehicle.

Les restriccions (4)-(6) obliguen que cada vehicle $k \in K$ comença al seu respectiu $o(k)$ i acaba a $d(k)$, i que si entren a un node, surten d'ell (també conegudes com a restriccions de flux).

Les restriccions (7)-(9) s'encarreguen que es compleixin els limits temporals, de finestres de temps, de precedència entre els *PD-pairs* (que els vehicles visitin el node de *pickup* abans del de *delivery*). A més, les restriccions del tipus (7) permeten que un vehicle hagi d'esperar-se abans de visitar un node si arriba abans que la finestra temporal estigui oberta.

Les darreres restriccions, (10)-(13), procuren que la capacitat dels vehicles no es sobrepassi i defineixen la capacitat inicial d'aquests.

Per tal de minimitzar la flota, es pot afegir un alt valor a $c_{o(k),j,k}$, per a $j \in P_k$. També s'ha d'incloure l'arc $(o(k), d(k)) \in A_k$ amb cost 0, per tal de permetre vehicles sense utilitzar.

2.1.4 Formalització final adaptada

En aquest apartat s'explica com és el cas concret dels serveis fixos que ofereixen `IMPD` i `AMB` i es realitzen si cal fer adaptacions a la formalització estàndard del `VRPPDTW`.

`IMPD` i `AMB` apliquen una normativa que s'ha de complir i que afecta a diferents *stakeholders* del sistema. Aquestes només vénen explicades en aquesta secció com a restriccions que ens imposa el client a l'hora de realitzar el càlcul del *routing* dels

vehicles.

A continuació s'exposen els requisits del sistema actual dels serveis fixos que el client imposa i les adaptacions que s'han de dur a terme sobre la formalització del `VRPPDTW`:

- Els usuaris **trien a quina hora volen ser recollits** (suposant que el taxi realitzarà un recorregut directe a la destinació). Per tant, se'ns limita la finestra temporal del node i de *pickup* i podem jugar amb el node $n+i$ de *delivery*. Per exemple, podem posar la *time-window* del punt de lliurament a $1.25 \cdot t_{ij} \pm \delta$, indicant que podem allargar la durada del trajecte en un 25%.

- S'ha de tenir en consideració que hi ha usuaris que només poden desplaçar-se en **vehicle adaptat**, en canvi, la resta pot viatjar tant en vehicles adaptats com no adaptats. Això es pot solucionar utilitzant la notació N_k proposada a la formalització on podem assignar un subconjunt de peticions segons el tipus de vehicle. D'aquesta manera, els vehicles no adaptats tindrien com a conjunt de peticions $P_k = \{i \in P \mid \text{l'usuari associat a } i \text{ no necessita taxi adaptat}\}$ i els vehicles adaptats $P_k = P$, és a dir, poden servir qualsevol petició. De la mateixa manera es construirien els conjunts D_k.

- No disposem de la informació de quins vehicles estan operant a cada hora, per tant no es pot saber quins taxis amb quines capacitats respectives estaran en circulació a cada franja horària. De manera que de moment treballarem considerant que es tracta d'una flota homogènia amb vehicles, en la que $\forall k \in K$, $C_k = C$ (on $C = 4$ places, per exemple).

- Hi ha un conjunt d'usuaris PMR que **no poden compartir taxi** amb altres usuaris, en aquest cas, la demanda associada a aquesta petició i serà $d_i = C$, per tal que ocupin tota la capacitat i no permetin la compartició de vehicle.

- A més del cost de les rutes, també es vol **minimitzar la flota** de vehicles. Per a fer-ho, es pot incloure la modificació que es comenta al final de la formulació del `VRPPDTW`: afegir un alt valor a $c_{o(k),j,k}$, $\forall j \in P_k$ i afegir l'arc $(o(k), d(k)) \in A_k$ amb cost 0, i d'aquesta manera permetre que hi hagi vehicles sense utilitzar.

- Haurem de resoldre un `VRPPDTW` per a cada dia de la setmana o per cada franja de temps dins d'un dia. Els paràmetres temporals a_i, b_i i les variables temporals T_{ij} prendran com a valor un nombre comprès entre 0 (que fa referència a les 00:00h d'un cert dia) i 86400 (que és el nombre total de segons en un dia és de $24h \cdot 60min \cdot 60s = 86400$ segons). Per tant, $a_i, b_i, T_{ij} \in \{0, \ldots, 86400\}$.

- La funció objectiu que volem minimitzar és el cost total del servei, el qual es paga per kilometratge, la matriu c_{ij} constarà de la distància entre anar del node i al j dins de la xarxa de transport.

Amb aquestes adaptacions, veiem que podem utilitzar la formulació descrita a la secció del `VRPPDTW` anterior sense afegir restriccions de més.

Anàlisi de la complexitat

Una vegada definit el model que representa el problema a tractar, a continuació es realitza una anàlisi de la complexitat en funció del nombre d'usuaris n.

Donat que $|A_k| = \mathcal{O}(n^2)$, tenim que el nombre de restriccions del nostre problema creix assimptòticament en l'ordre de $\mathcal{O}(Kn^2)$ en funció de les n peticions i el nombre de vehicles K.

Donat que $K \geq K_{min}$, $K \leq n$, i que $K_{min} \geq \lceil \frac{d(V)}{C} \rceil \geq \lceil \frac{n}{C} \rceil$. Tenim que $n \geq K \geq \lceil \frac{n}{C} \rceil$, on C és la capacitat dels vehicles (o la màxima de les capacitats en una flota heterogènia) la qual serà un valor menor que 10. Per tant, es pot veure que $K \sim \mathcal{O}(n)$.

Així doncs, s'observa que el nombre de restriccions que es generen en funció de n usuaris és de l'ordre de $\mathcal{O}(n^3)$.

2.2 Estat de l'art

Tal i com s'ha explicat anteriorment a la secció 1.3, ens trobem davant d'una variant més complexa del `VRP`, concretament, un *Heterogeneous Vehicle Routing Problem with Pick-Up and Delivery with Time Windows* (`HVRPPDTW`), donat que ens trobem amb una flota heterogènia on no tots els vehicles poden atendre totes les peticions.

Pel que fa a la tractabilitat del problema, el `VRP` és una generalització del conegut problema `TSP` (Travelling Salesman Problem). En el `TSP` es vol determinar el cost mínim d'un circuit que visiti tots els vèrtexs d'un graf (circuit Hamiltonià). El `TSP` és el cas concret del `VRP` quan el nombre de vehicles és $K = 1$. Se sap que el `TSP` és un problema `NP`, en la seva versió decisional, i `NP-Hard` en la versió d'optimització, ja que existeix una reducció[3] del problema `NP-Complet` de satisfacció de booleana `SAT` a `TSP`.

$$\texttt{SAT} \leq_p^m \texttt{dHAMPATH}^4 \leq_p^m \texttt{HAMPATH}^5 \leq_p^m \texttt{TSP} \quad [6]$$

Per tant, la versió d'optimització del `VRP` és també `NP-Hard`. Això significa que no es pot obtenir la solució òptima en un temps polinòmic en funció de l'entrada, excepte si `P = NP`.

Resoldre un problema `NP-Hard` com el `VRP` en la versió d'optimització pot ser realment costós en termes de temps d'execució segons la mida de l'*input* i, per tant, podem considerar entre calcular la solució òptima **exacta** o bé computar solucions **aproximades**.

[3]Denotem amb $\leq_p^m$ que existeix una funció de reducció de temps polinòmic (p) la qual és "many-to-one" (m), és a dir que no requerim que la funció de reducció considerada sigui injectiva .

[4]`dHAMPATH` - Problema del camí Hamiltonià dirigit

[5]`HAMPATH` - Problema del camí Hamiltonià

2.2.1 Mètodes exactes

Solucionar un problema de programació lineal entera requereix explorar totes les combinacions possibles de l'arbre de solucions. Per a explorar les solucions se sol fer ús d'algorismes basats en *Branch-And-Bound* [5], però el cost de fer-ho així és pràcticament exponencial, $\mathcal{O}(2^n)$. S'hauria de realitzar un estudi del nombre de variables i del nombre de restriccions que s'originen per a veure si és factible en termes de cost computacional i mida de l'entrada calcular la solució exacta.

2.2.2 Mètodes aproximats

Els mètodes aproximats donen solucions properes a l'òptima amb un temps d'execució raonable (generalment polinòmic). En una primera consulta a la literatura, els mètodes més comuns per a resoldre problemes de programació lineal entera i, en concret, el VRP, són els següents.

Heurístiques

Existeixen heurístiques per al VRP que ens ajuden a apropar-nos a cada iteració a una solució propera a l'òptima. Exploren un espai de solucions més limitat, però a canvi requereixen de poc temps d'execució [5].

Les heurístiques clàssiques tracten el problema obtenint primer una solució inicial basada en cerca local o veïnats limitats, a la que després apliquen una heurística de millora. Moltes d'aquestes heurístiques van ser desenvolupades abans dels anys noranta i estan enfocades en la resolució del VRP. Les dues tècniques principals estan basades en el concepte d'estalvi i en procediments d'inserció. L'algorisme més conegut és la proposta de Clarke and Wright (1964) [7], millorada posteriorment per Salomon al 1987 [8] en particular pel problema amb finestres de temps (VRPTW). Entre les heurístiques de millora més conegudes podem trobar el 2-opt Exchange proposat per Potvin i Rousseau al 1993 [9], o el lambda-exchange introduit per Osman al 1993 [10].

Metaheurístiques

L'ús de mètodes metaheurístics funciona molt bé per a instàncies a gran escala d'un problema. Els més usats i estudiats per aquest problema són els algorismes genètics, el *Simulated Annealing* i la Cerca Tabú. Aquests mètodes són coneguts en el camp de l'optimització i de l'intel · ligència artificial [11] i s'utilitzen per a fer cerca local i global en l'espai de solucions per tal d'obtenir un mínim local.

Cal remarcar que una de les metaheurístiques més utilitzades a la literatura per resoldre el VRP i les seves variants són els *algorísmes genétics* [12] i la Cerca Tabú proposada per Glover i Lacuna al 1997 [13], que ha estat sotmesa a múltiples modificacions i/o extensions al llarg del temps [14, 15, 16].

k-Aproximacions (relaxació de ILP)

Els problemes de programació lineal entera pertanyen, en general, a la classe de complexitat `NP-Hard` en la versió d'optimització (obtenir la solució mínima/màxima). Una prova d'això és que el problema `3SAT` es pot reduir al de Integer Linear Programming ($\texttt{3SAT} \leq_p^m \texttt{ILP}$). Una manera de trobar un k-aproximació del resultat òptim és *relaxar* les variables a nombres reals (es passa de resoldre el problema amb variables enteres $x_{ij} \in \{0, 1\}$ a considerar-les reals $0 \leq x_{ij} \leq 1$) i les restriccions convertir-les en inequacions. A partir d'aquesta transformació, es pot resoldre com un problema de `LP` amb un *solver* que utilitzi l'algorisme `Simplex`[6] amb un cost computacional acceptable i després randomitzar les variables x_{ij} per a tornar a tenir $x_{ij} \in \{0, 1\}$ [17, 18].

Aquest mètode ens permet després avaluar quant de distant és la solució obtinguda pel nostre algorisme A sobre l'entrada X en cas pitjor del mínim òptim global (una k-aproximació).

$$\frac{A(X)}{opt(X)} \leq k$$

2.2.3 Conclusions

Així doncs, ens trobem davant de dues tendències: treballar amb mètodes exactes o mètodes heurístics. Donades les dimensions del problema, en el qual tindrem moltes restriccions i un volum de dades elevat, necessitem un mètode de resolució que sigui eficient en termes de temps i memòria. D'altra banda, tenim unes limitacions de pressupost que no permeten incloure en el conjunt de recursos un *solver* eficient de pagament que ens resolgui una solució exacte, ja que *solvers* com ara CPLEX, ofereixen una versió de prova que limita a uns pocs centenar de variables/restriccions, límit que fàcilment se supera tenint en compte que el nostre problema genera un nombre de restriccions de l'ordre de $\mathcal{O}(n^3)$.

En el següent gràfic de la figura 2.1 es pot veure que amb instàncies de només 5 usuaris ja se supera el llindar de 1000 restriccions, les quals ja no es poden resoldre amb una llicència de prova dels *solvers*.

Pel mateix motiu, es descartaria l'opció de resoldre el problema fent relaxació lineal per utilitzar una k-aproximació, ja que tot i que el cost computacional sigui polinòmic amb un algorisme basat en `Simplex`, necessitaríem pagar la llicència d'un *solver*, la qual és força cara i se surt del pressupost.

[6] algorisme popular per a resoldre programació linal de temps polinòmic en el cas promig.

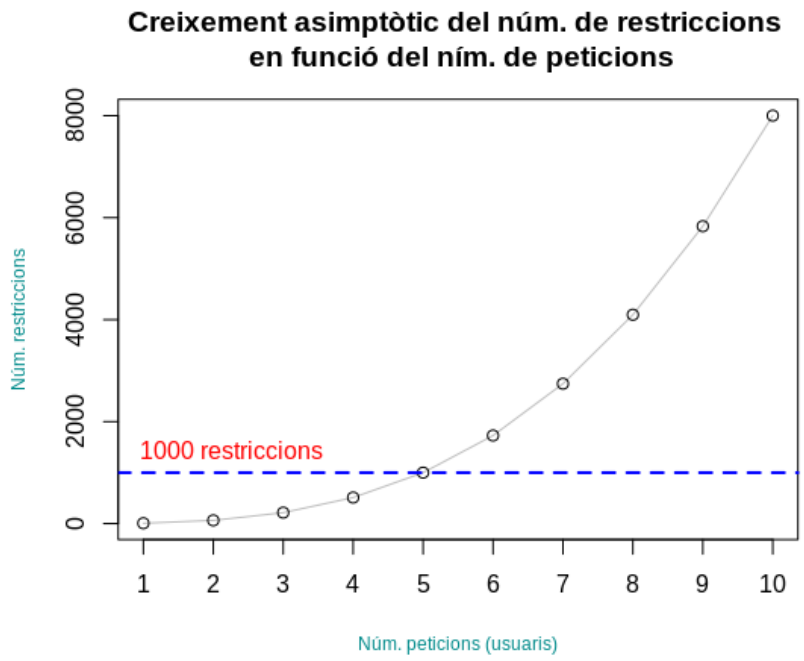

Figura 2.1: Gràfic que mostra el creixement asimptòtic del nombre de restriccions en la formalització com un problema de programació lineal (creixen en un ordre de $\mathcal{O}(n^3)$, on n és el nombre de peticions).

Per aquest motiu, els mètodes heurístics com la Cerca Tabú, *Simulated Annealing* o algorismes genètics apunten a ser els mètodes candidats per a implementar una solució del problema.

2.3 Selecció de l'algorisme

En els primers articles referents al `PDPTW`, s'hi troben heurístiques per a resoldre el 1-`VRPPDTW`, que és el cas en que la flota està formada per un únic vehicle (com un `TSP` amb *pickup* i *delivery* amb TW) [19, 20], o abordar el m-`VRPPDTW` detectant primerament clústers i resoldre cada un d'ells com un 1-`VRPPDTW` [21].

Els algorismes *cluster-first*, *route-second* poden semblar ideals per al nostre cas, però res garanteix que dos usuaris ubicats en una mateixa zona (clúster) vulguin desplaçar-se a una mateixa destinació i fer-ho en finestres de temps compatibles. Donat que cada usuari ve representat per un conjunt de dues localitzacions (*pickup* i *delivery*), és difícil generar clústers de bona qualitat sense incorporar informació de *routing*.

Per això, ens centrarem en implementar una heurística més genèrica que trobi una solució bona sense haver de fer suposicions de la distribució geogràfica de les peticions.

2.3.1 Mètodes candidats

A continuació, llistem els treballs més interessants i rellevants d'aquestes dues últimes dècades que presenten mètodes de cerca local i a la següent secció seleccionarem en quin ens basarem per a implementar la nostra solució.

- ***Fast local search algorithms for the handicapped persons transportation problem*, P. Toth and D. Vigo (1996)** [**14**]: Es resol un problema molt semblant al nostre, planificar rutes per a un servei destinat a persones discapacitades. Utilitzen cerca local per a trobar un òptim local.
- ***Solving the pickup and delivery problem with time windows using reactive tabu search*, W. P. Nanry and J. W. Barnes (2000)**[**22**]: Presenten una *Cerca Tabu Reactiva* eficient i per a resoldre instàncies grans del problema.
- ***Metaheuristic for the pickup and delivery problem with time windows*, Haibing Li i Andrew Lim (2001)** [**16**]: Utilitzen una combinació de *simulated annealing* amb cerca *Tabu*.
- ***Pickup and delivery problem with time windows: Algorithms and test case generation*, H. C. Lau i Z. Liang (2002)** [**23**]: Proposen un mètode de dues fases, una de construcció heurística i una de millora amb cerca *Tabu*.
- ***Genetic Algorithm for Mulicriteria Optimization of a Multi-Pickup andDelivery Problem with Time Windows*, I. H. Dridi, R. Kammarti, M. Ksouri i P. Borne (2010)** [**12**]: Algorisme genètic proposat recentment com a mètode de resolució del `VRPPDTW`.

Abans d'explicar els algorismes de cerca local proposats pels articles seleccionats a la secció de l'estat de l'art, explicarem breument en que consisteixen els mètodes de cerca local.

Recordem que en aquests mètodes es parteix d'una solució inicial qualsevol i una funció que es vol maximitzar o minimitzar (en el nostre cas seria minimitzar el cost de les rutes). A partir de la solució inicial, se li apliquen unes modificacions (p.e., permutacions) que generen noves solucions (veïnat). Aquestes modificacions reben el nom d'*operadors* i no són alteracions qualsevol, doncs no han de permetre sortir-se de l'espai de solucions (les noves solucions han de mantenir la factibilitat del problema). A continuació, dins d'aquest veïnat de solucions se selecciona la "millor" entre elles i a partir d'aquesta es torna a generar un veïnat per així successivament anar apropant-se a un òptim local. El mètode acaba quan s'ha assolit un criteri d'aturada.

Seguidament es mostra el pseudocodi:

Algorisme 1 Pseudocodi cerca local

$S \leftarrow S_0$ solució inicial
while no s'assoleix criteri d'aturada **do**
    Generar veïnat $\mathcal{N}(S)$
    Trobar "millor" solució $S_{best} \in \mathcal{N}(S)$
    $S \leftarrow S_{best}$
end while

A continuació s'expliquen els mètodes que s'han utilitzat en els articles seleccionats per a resoldre el `VRPPDTW` i, posteriorment, s'escollirà una estratègia per a començar a implementar una solució pròpia.

Fast local search algorithms for the Handicapped Persons Transportation Problem (1996) [14]

En aquest article, P. Toth i D. Vigo s'enfronten a un problema força similar al que es resol en aquest TFG. Es tracta d'optimitzar rutes d'un servei de transport per a persones amb discapacitat a la ciutat de Bolonya.

Expliquen una cerca local que consta de 4 operadors de millora banstant comuns a la literatura per a generar el veïnat (*neigbors*).

D'una banda, tenen operadors *intra-ruta*, que només alteren els nodes d'una pròpia ruta (p.e., intercanviar ordre dels nodes que recorre la ruta) i d'altra banda, hi ha els operadors *inter-ruta*, on s'alteren les peticions entre rutes diferents.

- **Operadors *intra-ruta***: aquesta operació intenta millorar la solució alternat la seqüència de nodes de *pickup* i *delivery* d'una ruta. La modificació s'obté de moure un node a un altre punt de la ruta, sempre que es preservi la factibilitat de la ruta. Es generen tots els moviments de nodes possibles mantenint la factibilitat i el millor moviment que faci reduir el cost, si n'hi hagués un, és el que s'acaba aplicant.
- **Operadors *inter-ruta***: Se'n distingeixen de 3 tipus:
  - *Trip insertion*: es treu una petició i (els dos nodes *pickup* i *delivery*) d'una ruta i es col · loquen els 2 nodes en les posicions més convenients de, si és possible, una altra ruta.
  - *Trip exchange*: s'intercanvien dues peticions i i j de dues rutes diferents. Cada petició es treu de la seva respectiva ruta actual i es col · loca en la millor posició de la seva nova rutes.
  - *Trip double insertion*: es treuen dues peticions i i j de dues rutes diferents. Es tria una tercera ruta diferent a les anteriors i en aquesta es col · loquen primer en la millor posició factible els nodes de la petició i i a continuació els de la petició j.

Amb aquests operadors conclouen explicant que obtenen bons resultats i comenten que és l'algorisme que utilitzen a Bolonya per a gestionar les flotes de taxis per a persones amb discapacitat.

Aquesta heurística millora en un 20% el cost del càlcul a mà que realitzaven per a assignar les rutes.

Metaheuristic for the pickup and delivery problem with time windows (2001) [16]

En aquest treball, els autors H. Li i A. Lim proposen una metaheurística basada en una Cerca Tabú dins d'un procediment de *simulated annealing* per a resoldre el `VRPPDTW` amb múltiples vehicles. Això és, una Cerca Tabú que es reinicia seguint un *simulated annealing* quan el procés de cerca no millora després d'un seguit d'iteracions, i així donar la possibilitat d'explorar altres òptims locals (*Tabu-embedded Simulated Annealing*). La cerca Tabú permet evitar fer cicles en el procés de cerca.

Defineixen 3 operadors per alterar les parelles de nodes de *pickup* i *delivery* (*PD-pairs*): *PD-Shift*, *PD-Exchange* i *PD-Rearrange*, els quals a la practica són iguals que els que proposen Toth i Vigo [14]. Els dos primers els utilitzen per a generar veïnat i el *PD-Rearrange* amb el propòsit de millorar la solució escollida a cada iteració.

- ***PD-Shift***: aquest operador mou els parells de nodes *pickup* i *delivery* a una altra ruta, mantenint la factibilitat de les restriccions imposades pel problema.
- ***PD-Exchange***: intercanvia dos *PD-pairs* de dues rutes.
- ***PD-Rearrange***: consisteix en reordenar la seqüència de *PD-pairs* mantenint la factibilitat.

Amb aquest algorisme, el qual combina *simulated annealing* amb cerca *Tabu*, els autors resolen instàncies de 100 peticions de manera eficient i instàncies amb diferents propietats de distribució i conclouen que és un algorisme fàcil d'adaptar a altres generalitzacions del `VRPPDTW`.

Solving the pickup and delivery problem with time windows using reactive tabu search (2000)[22]

W. P. Nanry i J. W. Barnes, proposen una Cerca Tabú reactiva, la qual reajusta els paràmetres de cerca i pot detectar valls locals en la topologia de les solucions i, a més, sortir-ne d'elles per tal de no quedar-se explorant un únic mínim local.

Els operadors que utilitzen per a generar solucions veïnes són similars als explicats als articles anteriors: *Single paired insertion*, *Swapping pairs between routes* i *Within route insertion*.

Es tracta d'una implementació més complexa que adapta paràmetres de cerca en funció de les solucions que va explorant, però que obté resultats eficientment molt propers a l'òptim (menys d'un 0.05% de diferència relativa de mitjana per a instàncies de 25, 50 i 100 peticions), i inclús arriba a aconseguir l'òptim per a algunes instàncies.

Pickup and delivery problem with time windows: Algorithms and test case generation (2002) [23]

Els autors d'aquest article, H. C. Lau i Z. Liang, proposen un mètode de dues fases per a resoldre el problema: Fase de construcció heurística i fase de millora amb Cerca Tabú.

Utilitzen diverses heurístiques per a construir una solució inicial, entre elles destaquen les heuristiques d'inserció (*Insertion Heuristic* i *Partitioned Insertion Heuristic*).

En referència a la cerca *Tabu*, els operadors que proposen per generar el veïnat són els mateixos que s'han explicat en els anteriors papers: *Single Pair Insertion*, *Swapping Pair Between Routes* i *Within Route Insertion*.

Com a conclusió, fan una comparativa experimental amb l'algorisme de W. P. Nanry i J. W. Barnes [22] que hem explicat anteriorment (*Reactive Tabu Search*) i els resultats obtinguts són força bons i molt propers als de Nanry i Barnes.

Genetic Algorithm for Mulicriteria Optimization of a Multi-Pickup and Delivery Problem with Time Windows (2010) [12]

Els autors d'aquest treball, I. H. Dridi, R. Kammarti, M. Ksouri i P. Borne , proposen resoldre el `PDPTW` amb un algorisme genètic per a minimitzar la distància i el cost de viatge total.

En aquest mètode, un *cromosoma* és una seqüència de nodes (p.e. [0, 1, 3, 4, 6, 0]) que representa l'ordre del nodes que ha de visitar un cert vehicle (el *depot* ve representat pel node 0).

Defineixen un **operador de creuament** entre *cromosomes* en el que donat dos *cromosomes pare*, aquests es parteixen en un mateix punt i es creen dos *cromosomes fills* resultants de creuar aquestes dues parts.

Aquesta operació fàcilment pot generar una solució infactible, i per aquest motiu defineixen un procediment de correcció que reordena la seqüència de cada *cromosoma* per a corregir les restriccions de precedència i de capacitat. A més s'ha de vigilar que els parells de nodes *pickup* i *delivery* no se separin en dues rutes diferents.

També implementen l'**operador de mutació**, que consisteix en alterar la seqüència de nodes del *cromosoma* intercanviant-ne dos nodes escollits a l'atzar, sempre que es preservin les restriccions de precedència i capacitat.

Com a conclusió, mostren resultats experimentals utilitzant el mètode que proposen obtenint una reducció del nombre de vehicles (2 vehicles per a cobrir 50 peticions i 4 vehicles per a 100 peticions), així com també de la funció objectiu.

2.3.2 Conclusió i selecció del mètode

Hem vist que les heurístiques més utilitzades a l'hora de optimitzar el `PDPTW` són mètodes de cerca local basats en la cerca *Tabu*, *simulated annealing* o algorismes genètics: Entre aquests, però, notem que, per simplicitat i robustesa, el que clarament

destaca i és més utilitzat és la **cerca *Tabu***. D'altra banda, els algorismes genètics presenten inconveniències a l'hora d'aplicar els operadors de cerca ja que fàcilment se surten de l'espai de solucions factibles i impliquen realitzar correccions a cada iteració.

A la taula 2.1, es resumeixen breument els pros i els contres de cada mètode amb l'objectiu de destacar les aportacions de cada un i així facilitar la elecció de l'algorisme a implementar.

		Valoració	
Algorisme	**Autors**	**Punts a favor**	**Punts en contra**
Cerca Local	P. Toth, D. Vigo	Fàcil d'implementar, bon temps d'execució per a instàncies grans i aconsegueixen minimitzar un 20% el cost	Pot succeir que la cerca s'encalli en un mínim local
Simulated Annealing + Tabu Search	H. Li, A. Lim	Bons resultats i temps d'execució. El fet d'utilitzar Simulated Annealing permet explorar diferents mínims locals	Implementació més elaborada
Reactive Tabu Search	W.P. Nanry, J.W. Barnes	Obtenen molt bons resultats de manera eficient per a instàncies grans del problema. Amb aquesta implementació eviten encallar-se en mínims locals i explorar-ne altres	Implementació bastant complexa i elaborada
Heurística + Tabu Search	H.C. Lau, Z. Liang	Solucions pròximes a les obtingudes amb Reactive Tabu Search. Bons temps d'execució	Obtenen els millors resultats quan utilitzen una heurística força complexa per a generar la solució inicial, però se'n poden utilitzar d'altres
Algorisme Genètic	I. H. Dridi, R. Kammarti, M. Ksouri, P. Borne	Obtenen una bona minimització de vehicles per a instàncies grans del problema	Complexitat d'implementació degut a que els operadors de creuament de cerca generen solucions infactibles

Taula 2.1: Comparativa i resum de pros i contres sobre els mètodes analitzats.

A partir de l'anàlisi realitzat a la secció 2.3.1, i considerant l'àmplia literatura que hi ha sobre l'ús de la Cerca Tabú per a resoldre el `PDPTW` juntament amb els bons resultats experimentals obtinguts amb aquesta, triarem aquest mètode per a implementar l'algorisme que solucionarà el nostre problema.

La implementació que millor ens convé per la bona qualitat dels resultats i per ser una proposta assequible d'implementar, és una adaptació de l'heurística constructiva juntament amb la Cerca Tabú proposada per H. C. Lau i Z. Liang. A la secció 3.2 del capítol 3 es detalla com s'ha implementat i configurat la Cerca Tabú i es defineixen quins operadors s'utilitzen per explorar solucions veïnes. També considerarem si convé utilitzar una heurística de construcció per a obtenir una bona solució inicial.

Capítol 3

Desenvolupament de l'algorisme

En aquest capítol primer s'explica el disseny del mòdul d'assignació i *routing* i de quines parts consta. Seguidament s'explica la implementació que es du a terme i la justificació del llenguatge de programació utilitzat per a implementar-lo.

3.1 Disseny del mòdul d'assignació

El mòdul que s'ha d'implementar s'ha de desenvolupar considerant-lo una peça desacoblada del sistema, de manera que pugui integrar-se com si es tractés d'un servei extern o llibreria.

La figura 3.1 mostra un esquema de l'arquitectura del mòdul que s'implementarà i el flux de funcionament d'aquest.

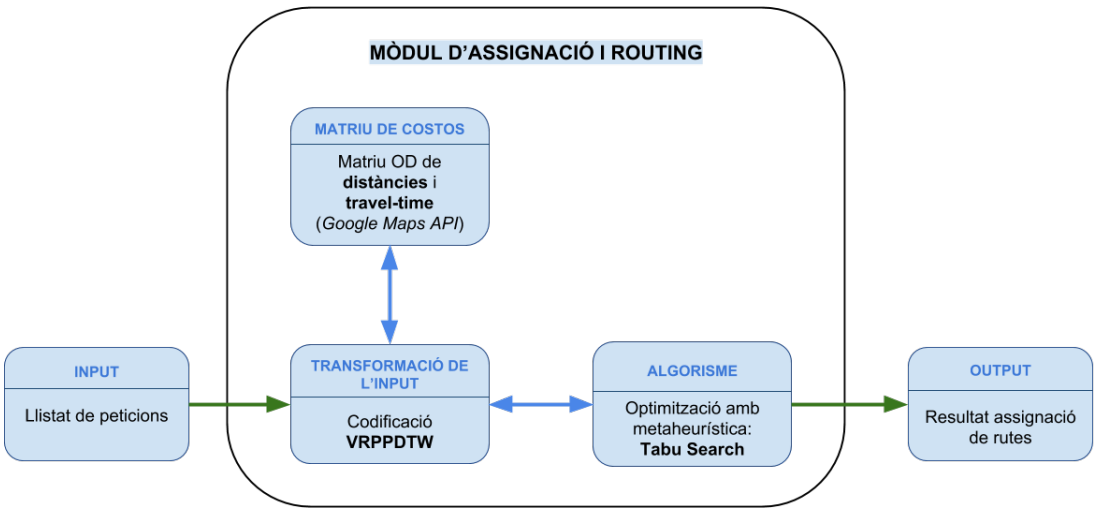

Figura 3.1: Esquema de flux del mòdul d'assignació.

A continuació es descriuen en detall cada una de les parts implicades en aquest

mòdul. La peça que constitueix l'algorisme es descriu a la secció 3.2 juntament amb la implementació detallada.

3.1.1 *Input*

Les dades necessàries i bàsiques per a representar una petició, per tal que pugui ser assignada a una ruta per a un cert dia, són: origen, destí, hora de recollida, si l'usuari requereix de vehicle adaptat i si l'usuari no pot compartir vehicle. Per tant, el format d'una petició de servei fix haurà de contenir, com a mínim, la següent informació:

```
{
    "id": 1,
    "origin": "41.390833,2.163056",
    "destin": "41.401136,2.123810",
    "hour": 10,
    "minute": 30,
    "adapted": false,
    "sharing": true
}
```

El camp *id* de les peticions serveix per a identificar-les i han de ser valors consecutius començant per 1.

Les coordenades d'*origen* i *destí* vénen en format de graus decimals (DD)[1] en forma d'*string*, primer la latitud i despres la longitud i separades per una coma, p.e. "41.40338,2.17403".

Les *hores* i *minuts* són enters compreses entre els rangs 0-23 i 0-59, respectivament.

Les capacitats dels vehicles, amplada de les finestres de temps i el temps de servei mig són paràmetres del mòdul d'assignació i *routing*, i no pas de l'*input*.

3.1.2 *Output*

El format de l'assignació resultant també serà en format clau-valor. Constarà d'un llistat de vehicles dels quals s'especificarà si és adaptat o no i la ruta que ha de seguir. La ruta estarà formada per un llistat de nodes a visitar, els quals contindran la coordenada en format de graus decimals, l'hora en la que ha de visitar el node (hora i minuts), a quina petició del *input* fa referència (id) i quin tipus d'acció s'hi realitza (*pickup* o *delivery*).

A continuació es mostra un exemple de com seria aquest format.

```
{
    "vehicles": [
```

[1] https://support.google.com/maps/answer/18539?hl=en&ref_topic=3092444

```
        {
            "adapted": true,
            "route": [
                {
                    "request_id": 5,
                    "coord": "41.390833,2.163056",
                    "action": "pickup",
                    "hour": 8,
                    "minute": 15,
                }, ...
            ]

        }, ...
    ]
}
```

3.1.3 Matriu de costos

El graf sobre el que es treballa es representa com una matriu d'adjacències, en la que recull totes les coordenades de les n peticions, tant la d'origen com la de destí (per tant hi haurà $2n$ coordenades en total), i es genera una matriu origen-destí (matriu OD) de dimensió $2n \times 2n$. En aquesta matriu es té el temps de viatge i la distància d'anar de qualsevol coordenada a qualsevol altra.

Per a aquest treball, la informació s'obté de la API de *Google Maps*. *Google Maps* permet obtenir tant una estimació del temps de viatge directe com la distància entre dos punts geogràfics utilitzant la xarxa de transport. A partir d'aquesta informació es construeixen les matrius de components t_{ij} i c_{ij} per a cada possible parella de nodes $(i, j) \in N \times N$. Notem que serà una matriu asimètrica, ja que la xarxa de transport no és bidireccional i a més *Google* té en compte el trànsit per al càlcul del temps de viatge.

Dels serveis que ofereix la API de *Google Maps* s'utilitzarà la *Distance Matrix* API[2], la qual proveeix la distància i el temps de viatge per a una matrius d'orígens i destins.

Exemple de petició a la API de *Distance Matrix*:

```
"https://maps.googleapis.com/maps/api/distancematrix/json?
origins=41.390833,2.163056|41.390833,2.163056&
destinations=41.3760081,2.1603171|41.3808688,2.1430154&
key={API_KEY}"
```

A continuació es mostra un fragment de la resposta a la petició anterior. Veiem que ens retorna el temps de viatge en segons i la distància de viatge en metres.

[2]https://developers.google.com/maps/documentation/distance-matrix/intro

```
{
    "status":"OK",
    "origin_addresses": [
        "Carrer de Calàbria, 18, 08015 Barcelona, Spain",
        "Carrer de Provença, 3, 08029 Barcelona, Spain",
    ],
    "destination_addresses": [
        "Pl. Catalunya-Portal de l'Àngel, 08002 Barcelona, Spain",
        "Carrer de la Diputació, 50, 08015 Barcelona, Spain",
    ],
    "rows": [
        {
            "elements": [
                {
                    "distance": {
                        "text": "2.4 km",
                        "value": 2404
                    },
                    "duration": {
                        "text": "9 mins",
                        "value": 518
                    },
                    "status":"OK"
                }, ... ]
        }, ... ]
}
```


Utilitzar l'API de *Google Maps* té un seguit d'implicacions i limitacions. D'una banda, les peticions estan restringides a 25 orígens o 25 destins com a màxim i a més no es pot superar els 100 elements OD per petició[3]. Per tant, el que es farà serà fer peticions de sub-matrius de 10×10 i així anar omplint les nostres matrius de temps t_{ij} i de distància c_{ij} de mida $2n \times 2n$. Addicionalment, es tracta d'una API de pagament que ofereix una quota de prova gratuïta mensual, la qual se sobrepassaria quan el sistema resolgués instàncies grans dels problema. Com a alternativa, es pot utilitzar *OpenStreetMap*, el qual és un servei col · laboratiu i gratuït, però, per contrapartida, el temps de viatge entre dos punts el calcula en *free-flow* (sense considerar trànsit a la xarxa de transport). Tanmateix, per a validar l'algorisme, s'ha optat per utilitzar l'API de *Google Maps*, que ofereix millors estimacions de temps de viatge.

Respecte als valors que la API proporciona, es podria pensar que treballar amb unitats com metres i segons implica operar amb nombres molt grans, i que es podria treballar amb kilòmetres o minuts. Però per tal d'evitar operacions amb nombres petits (propers a zero) les quals ens poden generar sensibilitat en alguns càlculs com

[3]https://developers.google.com/maps/documentation/distance-matrix/usage-and-billing

divisions o multiplicacions, es deixaran les unitats en metres i segons, per així no a afectar l'estabilitat numèrica de l'algorisme.

Un cop obtinguda aquesta matriu, convé guardar-la (p.e. en un fitxer de text extern) ja que són dades que no són gratuïtes.

3.1.4 Tranformació de l'*input*

Aquesta part del mòdul té la funció de transformar l'*input* en una instància del problema. Les peticions que es reben en format clau-valor, es processen, i es generen les estructures necessàries per a resoldre el problema (`VRPPDTW`).

Les n peticions que es reben tenen un atribut *id* que comença des de 1 i acaben fins a n, tots consecutius. Això ens permet identificar les peticions dels usuaris com un conjunt $1, \ldots, n$. A partir d'aquí, podem utilitzar la formulació que proposen Toth i Vigo [5] per a disposar els nodes d'una manera eficient: donada una petició i, els seus dos nodes seran i i $n+i$, que correspondran, respectivament, al node de *pickup* i al de *delivery*.

A partir d'això, es construeix el conjunt P de nodes de *pickup* i el deconjunt D de nodes de *delivery*, ambdós de mida n i representats per vectors o arrays. El conjunt de tots els nodes N serà la concatenació de P i D ($N = P \cup D$).

També es construeixen els vectors a_i i b_i relatius a les finestres de temps $[a_i, b_i]$, així com els vectors de d_i i l_i que fan referència a les demandes de cada petició i i la càrrega d'usuaris que implica el fet de passar pel node i, respectivament. Si la petició i consisteix en desplaçar d_i usuaris de i a $n+i$, es definirà $l_i = d_i$ i $l_{n+i} = -d_i$.

Aquesta disposició dels nodes permet accedir de manera instantània a la informació de cada node i de les matrius de costos. Donat un node de *pickup* i, el seu node respectiu de *delivery* es trobarà a la posició $n+i$ dels vectors.

Les rutes que ha de seguir un vehicle es representaran com a vectors que contenen una seqüència dels nodes que ha de visitar. A continuació es mostra un exemple on (1,6) i (2,7) serien parelles de nodes de *pickup* i *delivery*:

vehicle i:

0	1	6	2	7	0

La seqüència anterior expressa que el vehicle i ha de passar primer a recollir pel node 1, després deixar al node 6, recollir al node 2 i lliurar al node 7.

Tots els vehicles han de començar i acabar les rutes al *depot* (node 0). El *depot* serà un node suficientment allunyat per tal que el cost d'anar del *depot* a qualsevol node dels usuaris (i viceversa) sigui molt alt ($c_{0i} \uparrow, c_{i0} \uparrow, \forall i \in N$), de manera que qualsevol solució que intenti encabir les peticions en el menor nombre de vehicles serà la que minimitzi la funció objectiu. Si durant l'optimització una ruta queda buida (

0	0

), aquesta tindrà un cost de 0, doncs $c_{00} = 0$.

3.2 Implementació de l'algorisme

En aquesta secció s'explica la proposta inicialment presa per a començar a implementar l'algorisme d'optimització. Tanmateix, aquestes idees poden modificar-se de cara a l'algorisme final ja que es durà a terme un **desenvolupament iteratiu**. Això és, implementar una proposta inicial, provar-la amb un seguit d'instàncies, i avaluar-la per tal de realitzar una nova proposta per a millorar els resultats, tal i com es mostra a la figura 3.2. Aquesta metodologia és freqüent en la implementació d'heurístiques, ja que es poden millorar contínuament i inclús arribar a descartar idees inicials a partir dels resultats obtinguts.

Figura 3.2: Diagrama del desenvolupament iteratiu que es durà a terme en la implementació de l'algorisme.

Per tant, al final de la secció s'explicarà quines modificacions poden haver sorgit de la proposta inicial justificant-ho amb els resultats d'algunes execucions i es descriurà quin serà l'algorisme resultant que es validarà posteriorment.

3.2.1 Proposta inicial

Un cop ja es té la matriu de costos i el problema representat en estructures de dades, passarem a implementar la heurística de la cerca local que permeti trobar un òptim local. Per a fer-ho, com ja s'ha comentat a la seccions 2.3.2, ens centrarem en implementar una **Cerca Tabú**.

La Cerca Tabú és un mètode de cerca local que utilitza memòria per a impedir que es realitzin certs moviments (moviments tabú) i, d'aquesta manera, evitar que es repeteixin aquests moviments a curt termini [13]. Aquesta memòria sol rebre el nom de *llista tabú*. Els moviments enregistrats en aquesta llista no es podran realitzar durant

el procés de cerca durant un seguit d'iteracions. Passat aquest nombre d'iteracions tabú, els moviments s'eliminen de la llista i ja poden tornar-se a utilitzar novament.

A continuació es mostra un pseudocodi de com funciona:

Algorisme 2 Pseudocodi Tabu Search

$S \leftarrow S_0$ solució inicial
$tabuList \leftarrow \emptyset$
while no s'assoleix criteri d'aturada **do**
    Generar veïnat $\mathcal{N}(S)$
    Trobar "millor" solució $S_{best} \in \mathcal{N}(S)$ tal que $S_{best} \notin tabuList$
    $S \leftarrow S_{best}$
    Actualitzar $tabuList$
end while
return S

Donada una instància de n peticions, com a solució inicial trivial es pot generar una ruta per a cada petició, és a dir, assignar un vehicle a cada petició que sigui adaptat o no si s'escau (hi haurà $k = n$ vehicles). És una solució inicial que no implica ningun cost més enllà de crear les estructures de dades pertinents i que contempla el cas pitjor: que s'hagi d'assignar un vehicle a cada petició d'usuari degut a tenir restriccions temporals molt ajustades:

vehicle 1	0	1	$n+1$	0
vehicle 2	0	2	$n+2$	0
	...			
vehicle i	0	i	$n+i$	0
	...			
vehicle n	0	n	$2n$	0

Una alternativa no tan naïf per a generar la solució inicial seria utilitzar una heurística ràpida que ens construeixi unes rutes inicials més elaborades. Això permet començar amb una solució que utilitzi menys vehicles ($k < n$) cosa que milloraria l'eficiència de l'algorisme. La decisió de si utilitzar una heurística es veurà més endavant en cas que es necessiti millorar el temps d'execució del mètode.

Tot algorisme de cerca necessita d'uns operadors que generin solucions veïnes. Basant-nos en els treballs que s'han analitzat a la secció 2.3, s'implementaran **quatre** operadors de cerca (*Within Route Insertion*, *Single Pair Insertion*, *Double Pair Insertion* i *Swap Between Pairs*) que a continuació es descriuran detalladament i s'analitzarà el cost de cada un. Recordem que, pel que fa a les anàlisis de complexitat d'aplicar un d'aquests operadors sobre la solució actual, es considera que la instància té n peticions i k rutes (o el que és equivalent, vehicles).

3.2.2 Operadors de cerca

Within Route Insertion **(WRI)**: per cada ruta de la solució actual, es prova de reordenar la seqüència dels nodes i ens quedem amb la ruta que minimitzi la funció objectiu. Donat que cada ruta pot tenir $\mathcal{O}(n)$ nodes, i cada node el podem recol·locar en $\mathcal{O}(n)$ llocs de la ruta, i això es calcula per a cada una de les k rutes, el cost computacional d'aquesta operació és de $\mathcal{O}(n^2k)$.

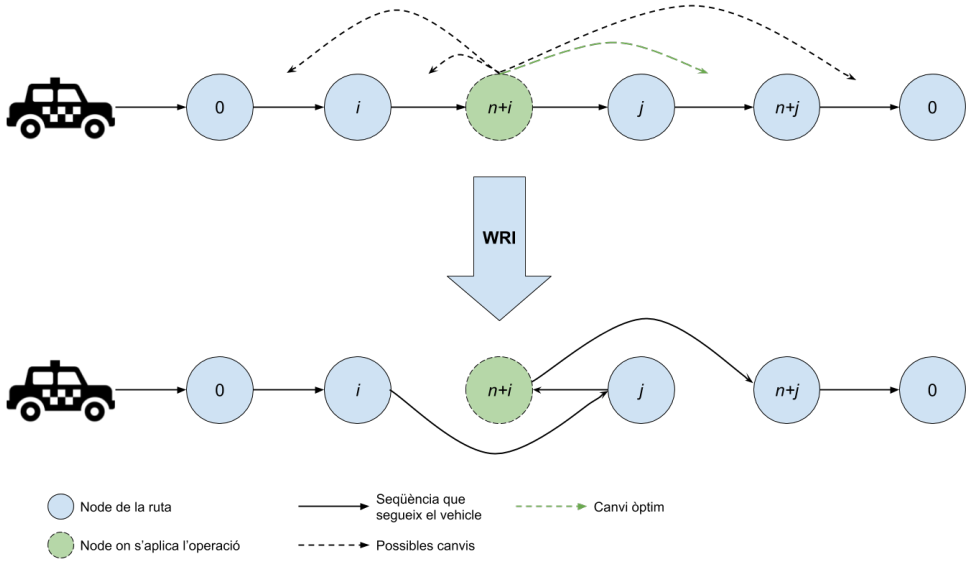

*Figura 3.3: Ruta resultant d'aplicar l'operació Within Route Insertion (**WRI**).*

Single Pair Insertion **(SPI)**: per a cada parell de rutes i i j, per cada parell de nodes *pickup* i *delivery* $(x, n+x)$ de la ruta i, intentem col·locar el *PD-pair* a la millor posició de la ruta j. Primer intentem col·locar el node de *pickup* i, en cas de trobar una posició factible, intentem col·locar a posteriori el node de *delivery*. De totes les possibles maneres factibles de col·locar els nodes ens que quedem amb la que doni menor cost a la ruta j.

Per cada inserció hem de mantenir la factibilitat de la solució (com sempre, respectar finestres de temps i capacitats dels vehicles), però a més també s'ha de tenir en compte que hi ha vehicles (no adaptats) que no poden admetre a les seves rutes peticions d'usuaris que necessiten vehicles adaptats (flota heterogènia).

Així doncs tenim, com a molt, $\sum_{t=1}^{n} t = \frac{n(n-1)}{2} = \mathcal{O}(n^2)$ possibles maneres de col·locar un *PD-pair* de la ruta i dins la ruta j. També tenim $k(k-1) = \mathcal{O}(k^2)$ possibles parelles de rutes (s'aplica l'operació en les dues direccions: de la ruta i a la j però després també de la j a la i). Així doncs, el cost computacional d'aplicar el **SPI** és de $\mathcal{O}(n^2k^2)$.

Observem que, per cada parell de rutes, generem una solució veïna. Per tant, amb aquest operador generarem $\mathcal{O}(k^2)$ solucions veïnes. A la pràctica el nombre de veïnat serà molt menys que k^2, ja que les restriccions de finestres de temps, capacitat i de vehicle adaptat limitaran bastant poder dur a terme insercions entre rutes.

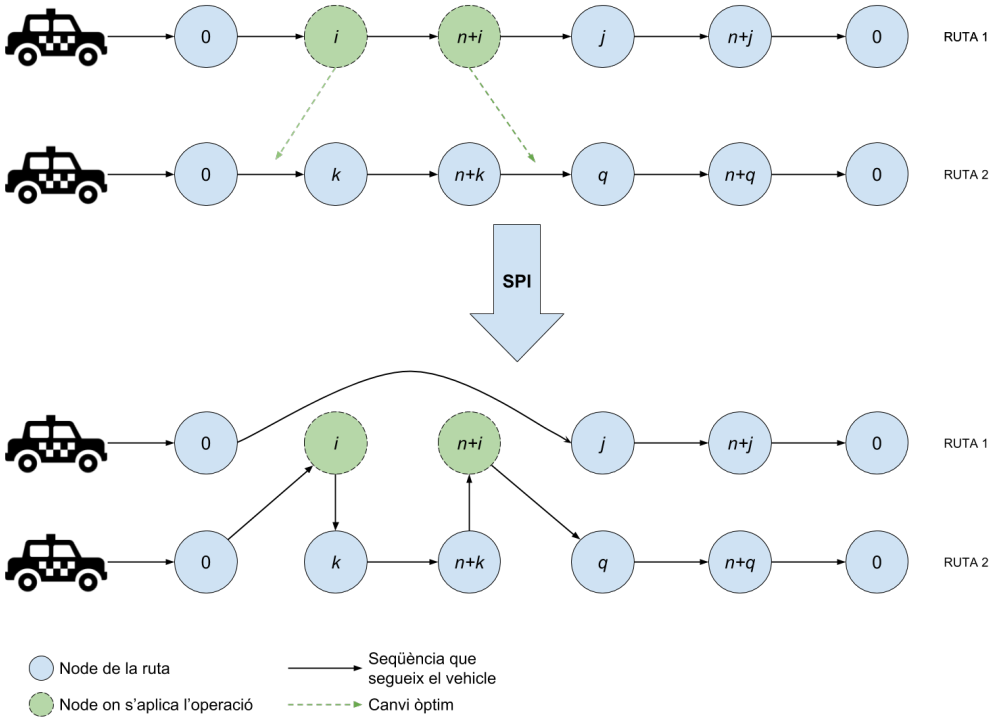

*Figura 3.4: Operació Single Pair Insertion (**SPI**) entre dues rutes.*

Double Pair Insertion **(DPI)**: donades tres rutes i, j i t diferents, agafar un *PD-pair* de i i un altre de j i inserir-los en la ruta t. Aquest operador serveix per a intentar encabir les peticions en unes poques rutes i alliberar vehicles per així minimitzar la flota. Si només es pot encabir un dels *PD-pairs*, s'insereix igualment, tot i que l'altre no es pugui. Funciona bé per a reduir el nombre de vehicles en les primeres iteracions.

Es vol que aquest operador sigui computacionalment eficient per tal de buidar rutes ràpidament. Per tant, generarem un únic veí (el primer que es generi al realitzar l'operació i que millori el cost actual). Pel que fa al cost computacional, haurem de recórrer les $\mathcal{O}(k)$ rutes de manera lineal fins a trobar algun *PD-pair* de les rutes i i j ($\mathcal{O}(n)$) que es pugui inserir en la primera posició factible de la ruta t ($\mathcal{O}(n)$). Per tant, el cost serà de $\mathcal{O}(n^2k)$.

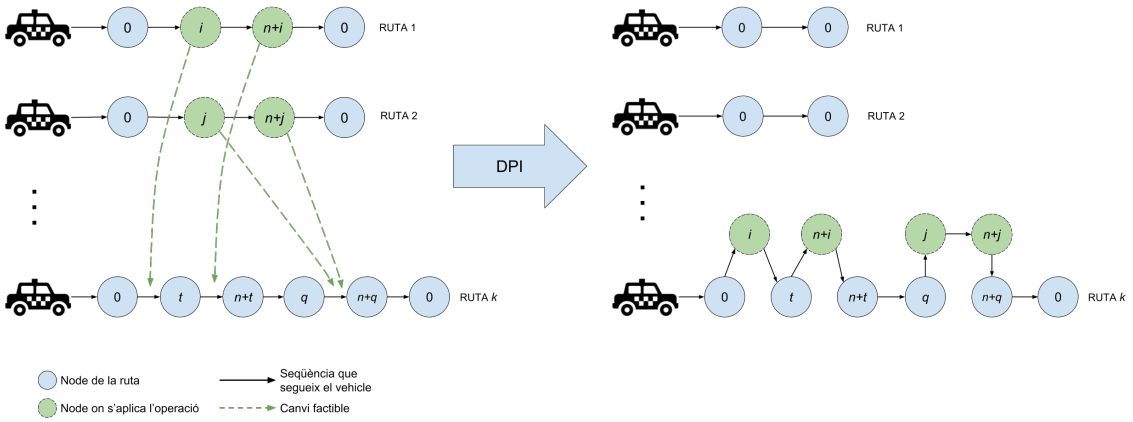

*Figura 3.5: Operació Double Pair Insertion (**DPI**) entre tres rutes, en la que es buiden dues de les rutes implicades.*

***Swap Between Routes* (SBR)**: per cada parell de rutes i i j (tals que $i \neq j$), agafem un *PD-pair* de cada ruta i intentem intercanviar-les. Els nodes a intercanviar s'eliminen de les rutes actuals i s'insereixen a les posicions que impliquin un menor cost a l'altra ruta.

El cost d'inserir els nodes en la millor posició d'una ruta és $\mathcal{O}(n^2)$, tal i com hem vist amb l'operador **SPI**. Per cada parell de nodes, el *swap* implica realitzar dues insercions (una a la ruta i i l'altra a la ruta j). Per tant, com que donades dues rutes es poden realitzar $\mathcal{O}(n^2)$ intercanvis, i donat que tenim $\mathcal{O}(k^2)$ possibles parelles de rutes, el cost computacional total d'executar aquest operador és de $\mathcal{O}(n^4k^2)$.

Obtenim una solució veïna per cada parell de rutes, això és, $\mathcal{O}(k^2)$ noves solucions que es generen amb aquest operador. Com amb els altres operadors, es probable que aquest nombre de veïns sigui inferior a $\mathcal{O}(k^2)$ si les restriccions del problema són molt ajustades.

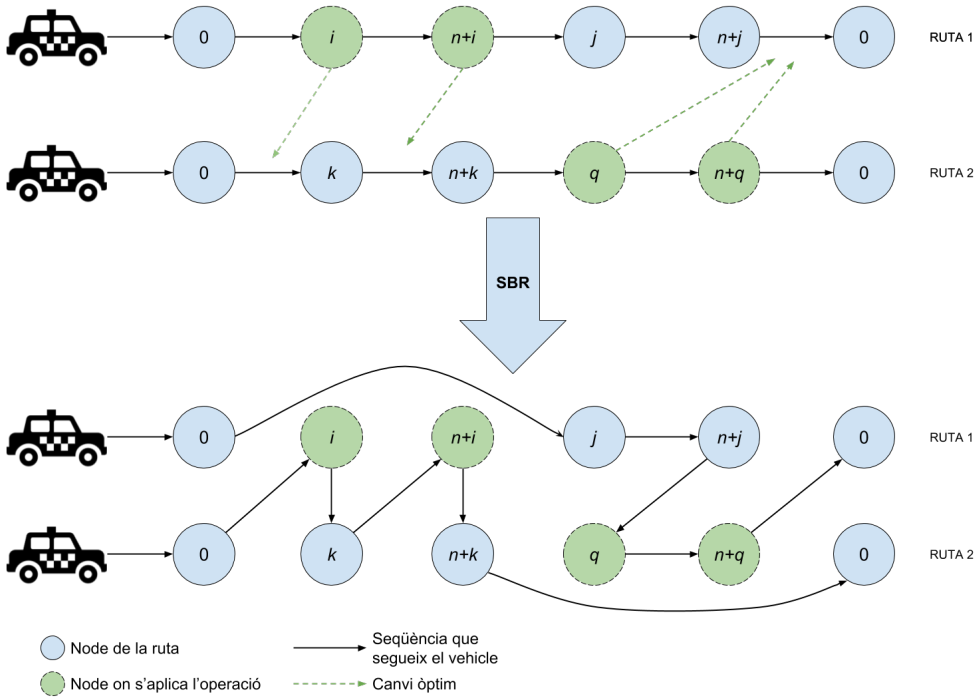

*Figura 3.6: Operació Swap Between Routes (**SBR**) entre dues rutes.*

En resum, per a cada iteració de la cerca, la complexitat de cada operador és la que es mostra a la taula 3.1 en funció de les n peticions i les k rutes (o vehicles).

Operador	Complexitat	Veïns generats
Within Route Insertion (WRI)	$\mathcal{O}(n^2k)$	1 (millora les rutes de la solució actual)
Single Pair Insertion (SPI)	$\mathcal{O}(n^2k^2)$	$\mathcal{O}(k^2)$
Double Pair Insertion (DPI)	$\mathcal{O}(n^2k)$	1 (genera el primer veí que es troba)
Swap Between Routes (SBR)	$\mathcal{O}(n^4k^2)$	$\mathcal{O}(k^2)$

Taula 3.1: Resum de la complexitat computacional i de l'ordre de generació del veïnat dels operadors en funció de n peticions i k rutes (o vehicles).

Cal observar que el nombre de vehicles k pot arribar a ser igual que n, però això implicaria que les rutes siguin de mida 2 (atendre una única petició). Per filar més prim es podria haver utilitzat $\bar{n} = n - k$ per indicar el nombre de nodes que tindrà una ruta, donat que si hi ha k rutes, almenys hi haurà 2 nodes a cada ruta, i, per tant, la ruta amb més nodes tindrà $\bar{n} = 2n - 2k = \mathcal{O}(n - k)$ nodes. Per tant, el cost computacional del **WRI**, **SPI**, **DPI** i **SBR** es podria reescriure com $\mathcal{O}(\bar{n}^2k)$, $\mathcal{O}(\bar{n}^2k^2)$, $\mathcal{O}(\bar{n}^2k)$ i $\mathcal{O}(\bar{n}^4k^2)$, respectivament, sent $\bar{n} = n - k$.

3.2.3 Cerca Tabú

L'estratègia de la cerca a cada iteració serà començar amb una solució inicial, i aplicar el **WRI** per veure si es pot millorar la solució actual. A continuació, es genera el veïnat amb els operadors **SPI**, **DPI** i **SBR** a partir de la solució obtinguda d'aplicar el **WRI**, i així successivament.

Pel que fa al criteri d'aturada, l'algorisme acabarà quan s'hagi superat un nombre d'iteracions, que serà configurable com a paràmetre, o si s'arriba a un punt de no millora.

Al final de cada iteració, la *llista tabú* enregistrarà els moviments que han generat la solució actual en format de tupla. D'aquesta manera, el moviment que s'hagi realitzar per a millorar la solució actual no es desfarà durant un seguit d'iteracions.

Pels moviments del tipus **SPI** que consisteix en intercanviar una petició p entre les rutes i i j (sent $p \in \{1, \ldots, n\},\ i, j \in \{1, \ldots, k\},\ i < j$), es registrarà un tupla del tipus (**SPI**, i, j, p).

Per als moviments de tipus **SBR** que consisteixin en intercanviar les peticions p i q entre les rutes i i j (sent $p, q \in \{1, \ldots, n\},\ i, j \in \{1, \ldots, k\},\ p < q$, $i < j$), s'afegirà a la llista tabú una tupla (**SBR**, i, j, p, q)

Les operacions **DPI**, com que a la pràctica són dues insercions del tipus **SPI**, el que s'enregistrarà a la llista tabú seran aquestes insercions (**SPI**) com si s'haguessin realitzat per separat.

La mida de la llista tabú serà parametritzable i també ho serà el nombre d'iteracions que un moviment serà tabú dins aquesta llista.

Així doncs, la primera proposta de Cerca Tabú que s'ha implementat és la següent:

Algorisme 3 Pseudocodi Cerca Tabú (primera versió)

$S \leftarrow S_0$ solució inicial
$tabuList \leftarrow \emptyset$
while no superat el nombre d'iteracions o s'obtingui millora **do**
  $S \leftarrow \mathbf{WRI}(S)$
  Generar veïnat $\mathcal{N}(S) = \mathbf{SPI}(S) \cup \mathbf{DPI}(S) \cup \mathbf{SBR}(S)$
  Trobar "millor" solució $S_{best} \in \mathcal{N}(S)$ tal que $S_{best} \notin tabuList$
  Afegir moviment generador de S_{best} a $tabuList$
  Actualitzar $tabuList$
  $S \leftarrow S_{best}$
end while
return S

3.3 Algorisme final millorat

En aquest apartat s'exposen les millores que s'han observat en les primeres execucions de l'algorisme i com resultarà l'algorisme definitiu.

3.3.1 Propostes de millora

A partir de les execucions sobre instàncies de diferents mides i característiques, s'ha pogut observar un seguit de possibles millores per tal de potenciar l'eficiència de l'algorisme:

- L'operació **SBR**, la qual és de les més costoses, no aporta gaire en procés de cerca i no s'escolleix gairebé mai com a l'operador que minimitzi el cost al llarg de les iteracions. Es considerarà excloure'l de l'algorisme ja que millorarà significativament el temps d'execució d'aquest.
- Quan les instàncies són molt grans (100-300 peticions d'usuaris), els operadors **SPI** i **SBR** són molt costosos i minimitzen molt lentament la funció objectiu. Això crea la necessitat de començar amb una solució inicial més elaborada o trobar una estratègia que redueixi ràpidament el nombre de vehicles (secció 3.3.2).
- L'operador **DPI** sol ser el més utilitzat per a minimitzar en les primeres iteracions, per tant, es pot considerar utilitzar-lo al principi ja que és més ràpid que la resta d'operadors.
- Si a cada iteració s'ordenen les rutes de menys a més mida, els operadors troben veïns de menor cost més fàcilment i la funció objectiu convergeix de manera més ràpida. Això és degut a que si s'ordenen segons aquest criteri, s'està afavorint que les insercions dels operadors es facin de rutes que tenen poques peticions a rutes amb més peticions, intentant així buidar les més petites i reduir el nombre de rutes/vehicles.

3.3.2 Heurístiques inicials de construcció

Com ja s'ha vist a la secció 3.3.1, es necessita una heurística constructiva que generi una bona solució inicial. Una heurística senzilla seria un algorisme *greedy* que comenci amb una ruta buida i intenti encabir cada una de les peticions en la ruta (o en qualsevol de les rutes existents) i, en cas que no pugui ser encabida en cap de les rutes, se'n crea una de nova on poder-la inserir. Així successivament fins a col · locar totes les peticions a alguna ruta.

A continuació es presenta un pseudocodi:

Algorisme 4 Pseudocodi Heurística de construcció inicial

```
requests ← peticions ordenades segons la a_i del node de pickup
routes ← ∅
for all request ∈ requests do
    if request pot insertar-se en alguna ruta existent then
        request s'insereix en una ruta ja existent
    else
        es crea una nova ruta que conté request
        routes.append([request])
    end if
end for
return S
```

L'ordenació inicial es deu a que les peticions amb finestres de temps més properes són més propenses a poder compartir vehicles. L'heurística té un cost computacional de $\mathcal{O}(n^2k)$, ja que s'han de recórrer les n peticions i, per cada una, recórrer les k rutes existents i, alhora, per cada ruta, provar d'inserir la petició en el primer lloc factible d'aquesta (aquesta inserció es pot fer de manera lineal a la mida de la ruta, la qual és, en el pitjor dels casos, $\mathcal{O}(n)$).

Aquest procediment constructiu triga menys d'un segon per a les instàncies més grans (300 peticions) i obté una reducció de la flota significant per a començar a utilitzar els operadors de cerca més sofisticats.

Per a instàncies grans del problema (més de 60 usuaris), les operacions **SPI** i **SBR** són bastant costoses i les iteracions de la cerca són més lentes. Els operadors **SPI** i **SBR** funcionen bé quan el nombre de vehicles és ajustat i ja és difícil inserir les peticions en altres rutes. Per tant, d'alguna manera s'ha de poder decrementar ràpidament el nombre de vehicles. Una manera de fer-ho, és amb l'operador **DPI** que és computacionalment ràpid i utilitzar-lo com a fase prèvia de reducció de flota abans de començar amb la Cerca Tabú.

Aquest procés inicial a la cerca local (i posterior a la heurística constructiva) consistirà en executar només l'operador **DPI** fins que no trobi una millora, amb la diferència, però, que la doble inserció es realitzarà de diferent manera amb una estratègia voraç per a convergir ràpidament a una solució on quedin el màxim de rutes buides.

A cada iteració, s'ordenen les rutes no buides de menys a més mida, i es divideix la llista per la meitat: d'una banda es tindran les rutes que passen per menys nodes i a l'altre les que passen per més. L'heurística consisteix en agafar dues rutes de la primera meitat i inserir un *PD-pair* de cada una a una ruta de la segona meitat, és a dir, executar l'operador **DPI** entre dues rutes de la primera meitat cap a una de la segona meitat. D'aquesta manera, s'aniran concentrant les peticions (*PD-pairs*) a un petit conjunt de rutes. A aquest procés li direm **DPI_greedy**.

A continuació es mostra el pseudocodi corresponent:

Algorisme 5 Pseudocodi DPI greedy (fase prèvia a la Cerca Tabú)

$S \leftarrow S_0$ solució inicial o actual
$tabuList \leftarrow \emptyset$
while no superat el nombre d'iteracions **do**
    $rutes \leftarrow$ ordenar ascendentment les rutes de S segons la mida
    $meitat1 \leftarrow rutes[1 : k/2]$
    $meitat2 \leftarrow rutes[k/2 : end]$
    $S \leftarrow$ **DPI**(S, i, j, k), on $i, j \in meitat1$ i $k \in meitat2$
end while
return S

El que interessa d'aquesta heurística és reduir en poques iteracions el nombre de vehicles, és a dir, deixar rutes buides, perquè així la Cerca Tabú (que presenta un cost computacional més alt) serveixi com a procediment de millora d'una solució prou bona.

3.3.3 Proposta final

A continuació es presenta la metaheurística final que reuneix totes les idees i millores proposades a la seccions 3.3.1 i 3.3.2.

La metaheurística que s'implementa com a proposta definitiva comença generant una solució inicial amb una heurística de construcció, seguidament s'intenta reduir de forma voraç el nombre de vehicles únicament amb l'operador **DPI_greedy** fins a no obtenir millora i, finalment, s'executa la Cerca Tabú per a seguir millorant la solució obtinguda fins al moment. La Cerca Tabú s'atura si se supera el límit d'iteracions de millora o si s'arriba a un estat de convergència on ja no s'obté millora.

La configuració dels paràmetres de la Cerca Tabú limita el procés de cerca a 50 iteracions, el nombre d'iteracions tabú es fixa a 5 i la mida de la llista tabú a 20 moviments.

Tot seguit, es descriu el pseudocodi de l'algorisme definitiu que s'implementarà.

Algorisme 6 Pseudocodi Heurística constructiva inicial + Cerca Tabú

$S \leftarrow S_0$ solució inicial generada amb l'**Heurística de Construcció Inicial**
$tabuList \leftarrow \emptyset$
while s'obtingui millora **do** ▷ Greedy DPI
  $\mathcal{N}(S) =$ **DPI_greedy**(S)
  Trobar "millor" solució $S_{best} \in \mathcal{N}(S)$ tal que $S_{best} \notin tabuList$
  $S \leftarrow S_{best}$
end while
▷ Tabu Search
while no superat el nombre d'iteracions o s'obtingui millora **do**
  $S \leftarrow$ **WRI**(S)
  Generar veïnat $\mathcal{N}(S) =$ **SPI**$(S) \cup$ **DPI**(S)
  Trobar "millor" solució $S_{best} \in \mathcal{N}(S)$ tal que $S_{best} \notin tabuList$
  Afegir moviment generador de S_{best} a $tabuList$
  Actualitzar $tabuList$
  $S \leftarrow S_{best}$
end while
return S

3.4 Elecció llenguatge de programació

En aquesta secció es raona l'elecció del llenguatge de programació amb el que s'implementarà la metaheurística argumentant si convé més un llenguatge compilat i, per tant, més ràpid, o si bé considerar-ne un d'interpretat.

El llenguatge ideal per a aquest tipus de problemes on el temps d'execució és clau, seria un de baix nivell, compilat, el qual el cost d'executar una instrucció sigui el menor, com ara `C` o `C++`. Dit això, també s'ha de tenir en compte el context on s'executarà l'algorisme i quin tipus de problema aborda.

El sistema on s'executarà aquest algorisme és un *back-end* d'una aplicació web que està implementat en *Python*. Triar aquest llenguatge per a desenvolupar l'algorisme seria clarament l'opció més senzilla a l'hora d'integrar-lo al sistema, ja que es podria importar i utilitzar com una llibreria. A més, per a seguir amb el manteniment del mòdul, utilitzar un llenguatge que domini l'equip de desenvolupament, és un altre punt a favor per triar *Python*.

El problema a resoldre és de cerca local, que implementa una heurística que pretén ser un mètode ràpid sigui quina sigui la mida de l'entrada. El procés de cerca passa per la generació de solucions veïnes a cada iteració, les quals algunes s'han de guardar en memòria, però deixen de ser necessàries a la següent iteració. Si s'acumulen i no s'esborren, la memòria pot anar creixent i empitjorar el rendiment del computador on s'estigui executant. Així mateix, un llenguatge que ofereixi *garbage collectors*, els quals eliminen objectes de la memòria que hagin perdut la referència, són recomanables per a garantir que no hi hagi *memory leaks* i es faci un ús raonable de la memòria. Llenguatges com *Python* disposen de *garbage collectors*, en canvi,

`C++` no[4].

Tal i com està representat el nostre problema (amb llistes i arrays), s'ha de treballar amb estructures dinàmiques de dades. Un llenguatge com *Python* permet desenvolupar a més alt nivell i fer operacions complicades amb aquestes estructures mitjançant menys línies de codi, la qual cosa facilita la legibilitat i la manipulació d'aquest. També compta amb funcions d'ordre superior (com ara `map, filter, any`, etc.) que faciliten treballar amb llistes o arrays (`C++` també disposa d'aquestes funcions d'ordre superior).

En conclusió, per la gran conveniència del context i les facilitats i alguns dels aventatges que ofereix, s'ha triat *Python* com a llenguatge per a implementar l'algorisme.

[4]https://www.educba.com/python-vs-c-plus-plus/

Capítol 4

Avaluació de l'algorisme

En aquest capítol s'exposen els resultats obtinguts a partir d'instàncies artificials del problema i es valida quantitativament la qualitat de les solucions generades per l'heurística contrastant-les amb la solució òptima calculada per un *solver* de programació lineal entera.

4.1 Generació d'instàncies del problema

En les següents seccions es prova l'algorisme amb un conjunt d'instàncies que s'han hagut de generar. En aquesta secció s'explica quins criteris s'han utilitzat per a construir-les i com s'han generat.

La necessitat de generar instàncies artificials es deu a que no s'ha pogut obtenir les peticions reals dels serveis fixos del Servei Públic de Transport Especial. Les úniques dades que se'ns ha proporcionat són rutes amb una prèvia agrupació de nodes en petits clústers, amb la qual cosa no es tenen els orígens i destins reals de la petició. Per aquest motiu, s'ha decidit generar instàncies artificials de peticions uniformement distribuïdes que serviran, d'igual manera, per a avaluar la qualitat de l'algorisme. Cal remarcar que és més difícil optimitzar rutes i aconseguir compartició de vehicles amb peticions que estiguin uniformement distribuïdes en l'espai, que no pas si segueixen un patró o tenen destinacions comunes (com succeeix amb les peticions reals).

Les instàncies del problema amb les que posarem a prova l'algorisme han de ser de la mida semblant a les de la vida real. Des de `IMPD` i `AMB` s'ha proporcionat informació de la magnitud de serveis fixos que han d'assignar a taxis. Aquestes dades serviran per donar una idea de la càrrega amb la que s'haurà de provar l'algorisme.

A la taula 4.1 es mostren dades reals per a l'interval de 5 dies del 10 al 14 de desembre del 2018.

	Tipus de taxi assignat als serveis				
Dia	**T.Amic compartit**	**Taxi Amic**	**Taxi ecològic**	**Taxi mes**	**Total Peticions**
Dilluns	58	366	162	35	**621**
Dimarts	55	446	154	42	**697**
Dimecres	56	375	168	30	**629**
Dijous	57	395	160	39	**651**
Divendres	56	390	152	34	**632**

Taula 4.1: Serveis fixos de taxi fets entre 10/12/2018 i 14/12/2018.

Veiem que el volum de vehicles és molt gran, més de 600 serveis fixos cada dia que es realitzen amb taxi. S'haurien de realitzar sub-instàncies del problema (p.e. resoldre un `VRP` pe les peticions del matí i una altre per les de la tarda) per tal de no sobrecarregar l'algorisme i tenir una matriu de costos més reduïda. Això és similar a com es fa actualment, ja que hi ha una assignació de taxis de matí i una altre de tarda.

A la taula 4.2 a continuació, veiem quina quantitat de serveis fixos hi ha en els franges horàries més conflictives.

	Franges horàries conflictives					
Dia	**8:00 - 8:30**	**8:31 - 9:00**	**9:01 - 9:30**	**16:31 - 17:00**	**17:01 - 17:30**	**17:31 - 18:00**
Dilluns	**31**	**70**	25	**76**	**51**	22
Dimarts	**35**	**65**	29	**74**	**51**	24
Dimecres	**34**	**67**	29	**78**	**54**	17
Dijous	**37**	**65**	26	**73**	**46**	21
Divendres	**33**	**62**	26	**73**	**45**	16

Taula 4.2: Serveis fixos de taxi fets entre 10/12/2018 i 14/12/2018.

Veiem que en les franges més conflictives s'arriba a tenir, en el pitjor dels casos, una demanda d'uns **130 vehicles per hora**.

Segons els informes proporcionats, també se sap que el nombre de serveis adaptats ronda el 50% de mitjana entre els diferents dies. Així doncs, la meitat dels serveis fixos les instàncies artificials que es generin seran d'usuaris que necessitin un vehicle adaptat. També hi ha alguns usuaris que no poden compartir vehicle, dels quals no s'ha pogut saber la proporció. L'única informació que s'ha proporcionat és que són una minoria i, per això, en la generació d'aquestes instàncies s'ha suposat que un 10% dels usuaris no podran compartir vehicle.

Una vegada definides les característiques i magnituds dels serveis, falta definir l'àrea on es generaran i hora de les peticions.

Aquestes peticions han d'estar ubicades dins de l'àrea de Barcelona i municipis del voltant (Badalona, Esplugues de Llobregat, l'Hospitalet de Llobregat, Sant Adrià de Besòs i Santa Coloma de Gramenet). Per a fer-ho, s'ha generat un *GEOJson* que abasta l'àrea desitjada de manera aproximada (figura 4.1) i s'han generat a l'atzar coordenades uniformement distribuïdes dins d'aquesta zona, les quals han de satisfer una distància mínima de 500 metres entre els *PD-pairs* (això és per evitar desplaçaments massa curts).

Les hores de *pickup* també s'han generat de manera uniformement distribuïda al llarg de l'horitzó temporal que s'hagi definit en els experiments. Les hores de recollida de les peticions generades són a partir de les 8:00 hores del mati.

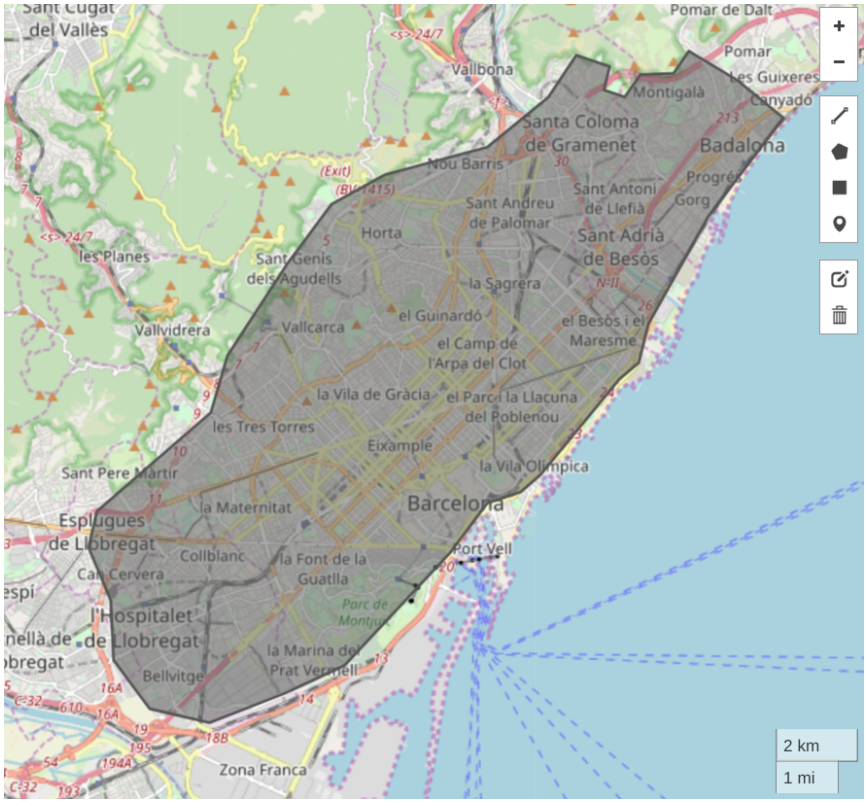

Figura 4.1: Visualització del GEOJson de l'àrea de peticions de les instàncies.

Visualització generada amb `http://geojson.io`.

4.2 Validació de l'algorisme

En aquesta secció s'estudia l'efectivitat de la metaheurística i s'analitzen les solucions obtingudes versus resultats òptims calculats pel *solver*. Aquesta part del projecte determina la qualitat de l'heurística i es mostra quant lluny es queda de l'òptim.

Primer es mostra el model que s'ha implementat per a calcular solucions òptimes, a continuació es realitza un disseny d'experiments i, finalment, una comparativa entre els resultats òptims i els de la heurística.

Per tal de validar l'algorisme, provem instàncies del problema petites (10-60 peticions) envers un model d'optimització en AMPL que es resol amb `CPLEX`, concretament la versió v12.0.8.0.0. El *solver* `CPLEX` que s'utilitza és una versió per a estudiants que s'ha concedit per a la realització d'aquest Treball de Fi de Grau, la qual admet un límit de restriccions superior al de la versió gratuïta. Cal remarcar que aquest *solver* no es pot utilitzar per amb propòsits més enllà dels acadèmics i per tant no es podria utilitzar en el sistema que entrarà en producció. Això reforça la decisió d'implementar una metaheurística per a resoldre el problema.

4.2.1 Model AMPL

AMPL és un acrònim de *A Mathematical Programming Language* i és un llenguatge de programació algebraica per a descriure i resoldre problemes d'optimització i de planificació. AMPL suporta desenes de *solvers* per a resoldre aquest tipus de problemes.

El model d'optimització del `VRPPDTW` descrit a la secció 2.1.3 s'ha traduït a un model escrit en AMPL (annex A). Amb aquest model es podran calcular valors òptims exactes i servirà per contrastar la qualitat de les solucions de l'algorisme desenvolupat i observar si el temps d'execució amb el que s'obtenen resultats és raonable comparat amb utilitzar un *solver*.

Tanmateix, s'ha hagut de modificar alguna restricció del model de programació lineal (secció 2.1.3) ja que `CPLEX` no admet restriccions no lineals, i en concret n'hi ha dues que ho són: la (2.7) i (2.10).

$$x_{ijk}(T_{ik} + s_i + t_{ij} - T_{jk}) \leq 0 \qquad \forall k \in K,\ (i,j) \in A_k \qquad (2.7)$$

$$x_{ijk}(L_{ik} + l_j - L_{jk}) = 0 \qquad \forall k \in K, (i,j) \in A_k \qquad (2.10)$$

La restricció (2.7) es pot linealitzar fàcilment utilitzant el mètode de la *big-M*, ja que és una restricció de desigualtat entre variables enteres. Donada una restricció del tipus $xy \leq 0$, es pot linealitzar prenent una valor M apropiadament gran i reescrivint-la de la forma $y \leq (1-x)M$, sent $x \in \{0,1\}$.

D'aquesta manera, la restricció (2.7) passarà a ser de la següent forma:

$$x_{ijk}(T_{ik} + s_i + t_{ij} - T_{jk}) \leq 0 \implies (T_{ik} + s_i + t_{ij} - T_{jk}) \leq (1 - x_{ijk})M \qquad (2.7)$$

i, en concret, M pot valer $max(b_i + s_i + t_{ij} - a_j, 0)$, donat que $a_i \leq T_{ik} \leq b_i$.

Pel que fa a la restricció (2.10), la linealització no és tan directa, ja que es tracta d'una igualtat i es necessita d'una variable auxiliar activadora. Donada una restricció $xy = 0$, on $x, y \in \mathbb{Z}$ qualssevol, es pot aplicar el mètode de la *big-M* si es generen les variables δ_1, $\delta_2 \in \{0,1\}$, llavors la restricció es pot reescriure com

$$-M\delta_1 \leq x \leq M\delta_1 \qquad -M\delta_2 \leq y \leq M\delta_2 \qquad \delta_1 + \delta_2 \leq 1$$

sent M un valor prou gran tal que $\forall x,\ |x| < M,\ \forall y,\ |y| < M$.

Tanmateix, com que en el nostre cas particular la variable x_{ijk} ja és binària, es pot realitzar la linealització utilitzant una única variable auxiliar: sigui $x \in \{0,1\}$, $y \in \mathbb{Z}$ qualsevol i una variable auxiliar $\delta \in \{0,1\}$ llavors, la restricció $xy = 0$ es pot reescriure tan sols amb les restriccions $-M\delta \leq y \leq M\delta,\ \ x + \delta \leq 1$.

Així doncs, sigui $\delta_{ijk} \in \{0,1\}$, la restricció (2.10) quedaria de la forma:

$$-M\delta_{ijk} \leq L_{ik} + l_j - L_{jk} \leq M\delta_{ijk}$$
$$x_{ijk} + \delta_{ijk} \leq 1 \qquad \forall k \in K,\ (i,j) \in A_k$$

Per tant, el model en AMPL té com a variables les inicialment definides pel problema x_{ijk}, T_{ik}, L_{ik} i, a més, la variable binària δ_{ijk} que ha sorgit per a linealitzar una de les restriccions.

4.2.2 Disseny d'experiments

Per a validar l'algorisme es realitzaran un seguit d'experiments que s'executaran tant amb el *solver* `CPLEX` com amb la metaheurística implementada per tal de contrastar els resultats. Dit això, els factors que poden ser modificats a cada escenari són el **nombre de peticions** i l'**amplada de la finestra temporal** de recollida i lliurament (amplada TW). Fixarem el temps de servei a 1 min i l'horitzó temporal a 1 hora.

Pel que fa al nombre de peticions, es proven els nivells $10, 20, 30, 40, 50, 60$ i 70, que es consideren valors petits i computacionalment viables executar-se amb el *solver*.

Per al factor de finestra temporal es proven nivells de 5 min i 10 min. Quan es diu que l'amplada de la finestra és de 5 minuts, significa que donada una hora h (d'arribada o lliurament) d'un node i, la finestra de temps associada a aquest node és $[a_i, b_i] = [h - 2.5, h + 2.5]$.

Instància	Nombre peticions (n)	Horitzó temporal	Temps de servei	Amplada TW *Pickup* (min)	Amplada TW *Delivery* (min)
`10-1H-5TW-VRP`	10	1 hora	1 min	5	5
`10-1H-10TW-VRP`	10	1 hora	1 min	10	10
`20-1H-5TW-VRP`	20	1 hora	1 min	5	5
`20-1H-10TW-VRP`	20	1 hora	1 min	10	10
`30-1H-5TW-VRP`	30	1 hora	1 min	5	5
`30-1H-10TW-VRP`	30	1 hora	1 min	10	10
`40-1H-5TW-VRP`	40	1 hora	1 min	5	5
`40-1H-10TW-VRP`	40	1 hora	1 min	10	10
`50-1H-5TW-VRP`	50	1 hora	1 min	5	5
`50-1H-10TW-VRP`	50	1 hora	1 min	10	10
`60-1H-5TW-VRP`	60	1 hora	1 min	5	5
`60-1H-10TW-VRP`	60	1 hora	1 min	10	10
`70-1H-5TW-VRP`	70	1 hora	1 min	5	5
`70-1H-10TW-VRP`	70	1 hora	1 min	10	10

Taula 4.3: Disseny d'experiments de per a la validació de l'algorisme.

4.2.3 Comparativa CPLEX versus Metaheurística

A la taula 4.4 es mostra una comparativa entre els resultats obtinguts per `CPLEX` i la metaheurística implementada. La columna *Cost Total* fa referència al cost de la funció objectiu (distància total recorreguda en metres), que inclou els cost de desplaçar-se des dels *depots* i cap a ells (que és un cost molt elevat). La columna *Cost Taxis* conté el cost de les rutes dels taxis entre nodes dels usuaris (sense considerar els *depots*). Cal tenir en compte que pot donar-se el cas en que el *Cost Taxi* obtingut per la metaheurística sigui inferior al de l'òptim, doncs potser amb menys vehicles es fa més recorregut entre petició i petició que si destinem un vehicle exclusivament per a cada petició.

Hi ha algunes execucions amb `CPLEX` que no tenen resultats. Això es deu a que no ha sigut possible executar-les per temps i per falta de memòria en els ordinadors que es disposaven. El fet de posar una finestra de temps més ampla (10 min) fa que hi hagi moltes més solucions factibles i que algunes variables, com ara T_{ik}, puguin

	AMPL + CPLEX				Heurística constructiva + Tabu Search				
Instància	**Cost Total Òptim (m)**	**Cost Taxis (m)**	**Vehicles utilitzats (k)**	**Temps d'execució**	**Cost Total Metaheurística (m)**	**Cost Taxis (m)**	**Vehicles utilitzats (k)**	**Temps d'execució**	**% diferència Cost Total**
10-1H-5TW-VRP	3 303 466.0	103 466.0	8	0.09	3 698 233.0	98 233.0	9	0.17s	11.95%
10-1H-10TW-VRP	2 921 083.0	121 083.0	7	0.11s	2 921 083.0	121 083.0	7	0.17s	0%
20-1H-5TW-VRP	3 862 561.0	262 561.0	9	0.77s	4 266 525.0	266 525.0	10	0.78s	10,45%
20-1H-10TW-VRP	4 636 164.0	236 164.0	11	4.57s	5 037 441.0	237 441.0	12	0.87s	8.65%
30-1H-5TW-VRP	7 137 462.0	337 462.0	17	6.39s	7 537 519.0	337 519.0	18	0.34s	5.61%
30-1H-10TW-VRP	4 649 826.0	221 194.0	13	5127.98s = 1h 28min (aturat per temps)	4 957 158.0	255 465.0	14	0.19s	6.61%
40-1H-5TW-VRP	11 220 170.0	420 170.0	27	33.21s	11 227 827.0	427 827.0	27	0.83s	0.07%
40-1H-10TW-VRP	-	-	-	-	9 591 043.0	391 043.0	23	1.36s	-
50-1H-5TW-VRP	11 749 359.0	549 359.0	28	140.26s	12 181 176.0	581 176.0	29	2.41s	3.67%
50-1H-10TW-VRP	-	-	-	-	9 698 352.0	498 352.0	23	3.07s	-
60-1H-5TW-VRP	11 892 461.0	692 461.0	28	312.18s	13 073 115.0	673 115.0	31	2.68s	9.93%
60-1H-10TW-VRP	-	-	-	-	11 067 898.0	667 898.0	26	1.21s	-
70-1H-5TW-VRP	14 337 175.0	737 175.0	34	5746.69s = 1h 36 min	15 193 573.0	793 573.0	36	3.01s	5.97%
70-1H-10TW-VRP	-	-	-	-	13 093 136.0	693 136.0	31	4.27s	-

Taula 4.4: Resultats obtinguts amb CPLEX (òptims) i amb la metaheurística implementada (aproximats) de les execucions de les instàncies.

prendre molts més valors. Això fa augmentar el temps d'execució del *solver* de manera significant.

La figura 4.2 compara el nombre de vehicles utilitzats resultants de cada un dels mètodes, exacte (`CPLEX`) vs. aproximat (metaheurística), per als escenaris amb finestra de temps fixada a 5 min, els quals s'ha pogut completar l'execució de tots ells amb el *solver* `CPLEX` i obtenir una solució.

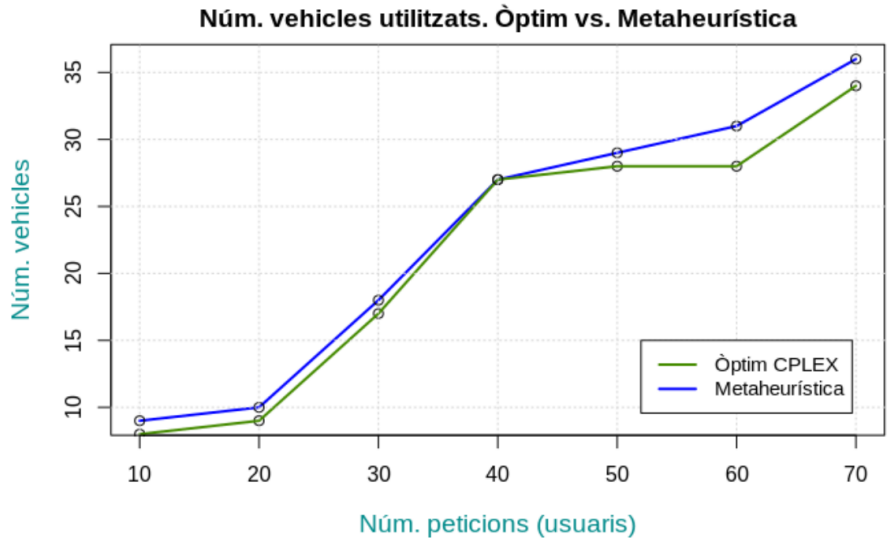

Figura 4.2: Nombre de vehicles utilitzats calculats per CPLEX i per la metaheurística en funció del nombre de peticions. Amplada de finestres de temps fixada a 5 min.

La figura 4.3 compara el temps de la metaheurística envers el temps de càlcul d'obtenir l'òptim del *solver* `CPLEX`. Cal remarcar que per al cas de 70 peticions, el temps d'execució de `CPLEX` supera la hora i mitja (més de 5700 segons) i s'ha limitat l'eix d'ordenades a un rang que permetés visualitzar la resta de valors correctament.

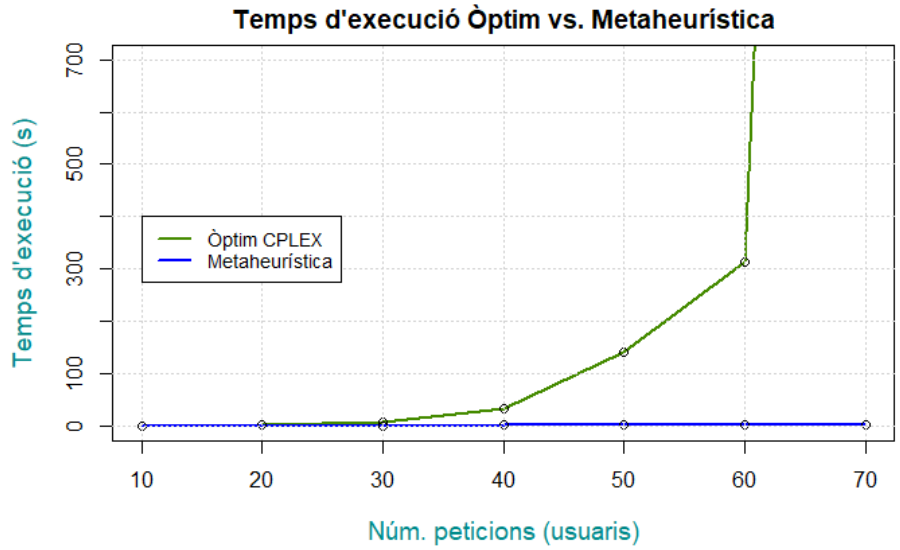

Figura 4.3: Temps d'execució del CPLEX i de la metaheurística en funció del nombre de peticions. Amplada de finestres de temps fixada a 5 min.

Vista la comparació de resultats en les taules i les gràfiques anteriors, s'observa que l'heurística obté bons resultats propers a l'òptim per als escenaris dissenyats (taula 4.2).

A més, el temps d'execució de la metaheurística és molt millor per al problema, donat que el temps de càlcul del *solver* incrementa de manera exponencial amb la mida de l'entrada (figura 4.3). La metaheurística és ideal per obtenir solucions prou bones en un temps raonable, tal i com es necessita amb els serveis fixos dels Servei Públic de Transport Especial.

4.3 Resultats de l'algorisme

Després de validar l'algorisme, en aquest apartat es mostren els resultats d'execucions d'escenaris amb diferents factors per veure com afecten a la solució calculada. També es presenten els resultats de manera visual per tal de facilitar la interpretació i la correctesa de les solucions.

4.3.1 Disseny d'experiments

Donat que la màxima càrrega que s'arriba a tenir és de 130 peticions de serveis fixos en una franja d'1 hora (secció 4.1), s'han provat instàncies de 100 i 150 peticions. Dit això, els factors que s'han modificat són l'**amplada de la finestra temporal** de recollida i lliurament (amplada TW) i els **temps de servei**.

El factor més interessant de veure quin impacte té sobre la solució és l'amplada de la finestra temporal. Per a aquest factor es proven nivells de 5 min (finestra ajustada), 10 min, i 20 min (finestra bastant ampla). D'igual manera a la secció 4.2 de la validació, quan es diu que l'amplada temporal és de 5 minuts, significa que donada una hora h (d'arribada o lliurament) d'un node i, la finestra de temps associada a aquest és $[a_i, b_i] = [h - 2.5, h + 2.5]$.

Per al temps de servei mig, s'han provat dos valors: 2 min (temps raonable per a la majoria d'usuaris) i 4 min (temps molt generós, raonable per a usuaris amb mobilitat molt reduïda). També s'han configurat diferents horitzons de temps per veure com es comporta l'algorisme amb un escenari amb més o menys concentració de peticions, amb nivells d'1 hora (per simular la càrrega real) i de 2 hores (un càrrega més repartida).

En resum, aquests són els factors i els diferents nivells que s'han configurat:

- **Horitzó temporal**: {1, 2} hores.
- **Temps de servei**: {2, 4} minuts.
- **Amplada *time-windows* (TW)**: {5, 10, 20} minuts.

A la taula 4.5 es pot veure la configuració dels escenaris segons els factors i nivells que s'acaben de comentar.

Nombre peticions (n)	Horitzó temporal	Temps de servei	Amplada TW (min)	Nom instància
100	1 hora	2 min	5	100-1H-2ST-5TW-VRP
			10	100-1H-2ST-10TW-VRP
			20	100-1H-2ST-20TW-VRP
		4 min	5	100-1H-4ST-5TW-VRP
			10	100-1H-4ST-10TW-VRP
			20	100-1H-4ST-20TW-VRP
	2 hora	2 min	5	100-2H-2ST-5TW-VRP
			10	100-2H-2ST-10TW-VRP
			20	100-2H-2ST-20TW-VRP
		4 min	5	100-2H-4ST-5TW-VRP
			10	100-2H-4ST-10TW-VRP
			20	100-2H-4ST-20TW-VRP
150	1 hora	2 min	5	150-1H-2ST-5TW-VRP
			10	150-1H-2ST-10TW-VRP
			20	150-1H-2ST-20TW-VRP
		4 min	5	150-1H-4ST-5TW-VRP
			10	150-1H-4ST-10TW-VRP
			20	150-1H-4ST-20TW-VRP
	2 hora	2 min	5	150-2H-2ST-5TW-VRP
			10	150-2H-2ST-10TW-VRP
			20	150-2H-2ST-20TW-VRP
		4 min	5	150-2H-4ST-5TW-VRP
			10	150-2H-4ST-10TW-VRP
			20	150-2H-4ST-20TW-VRP

Taula 4.5: Disseny d'experiments per a avaluar l'impacte dels diferents factors.

D'aquestes execucions, s'han recollit un seguit de KPIs (*Key Performance Indicator*) que serviran per a interpretar i analitzar el rendiment de l'algorisme i la qualitat de les solucions.

4.3.2 Anàlisi de KPIs

El KPI més important de solució obtinguda és el nombre de vehicles utilitzats, doncs és el que els *stakeholders* de negoci desitgen reduir per a poder destinar taxis a altres serveis. Degut a la preocupació d'aprofitar millor els recursos i de fomentar la compartició de vehicles, es presenten KPIs relacionats amb l'ocupació del vehicle. A més, com que els vehicles cobren per quilometratge recorregut, aquests indicadors estan relacionats amb la distància recorreguda per les rutes.

A continuació, es descriuen els KPIs que es mostren seguidament a la taula 4.6:

- **Vehicles utilitzats**: Nombre de vehicles que l'algorisme calcula que són necessaris per a cobrir les n peticions.
- **% de distància de vehicle compartit**: distància relativa en la que s'ha compartit vehicle respecte la distància total recorreguda per tots els vehicles.
- **Ocupació mitjana per unitat de distancia**: distància ponderada pel nombre d'usuaris que ocupen el vehicle dividit entre la distància total de tots els vehicles. El valor d'aquest indicador, seguint la notació de la formulació del

VRPPDTW de la secció 2.1.3, és el següent:

$$\frac{\substack{\textit{Distància recorreguda ponderada}\\ \textit{per l'ocupació de tots els vehicles}}}{\substack{\textit{Distància total recorreguda}\\ \textit{per tots els vehicles}}} = \frac{\sum_{\forall k \in K} \sum_{\substack{(i,j)\in A_k \\ x_{ijk}=1}} t_{ij} L_{ik}}{\sum_{\forall k \in K} \sum_{\substack{(i,j)\in A_k \\ x_{ijk}=1}} c_{ij}}$$

- **% de distància de vehicle buit**: distància relativa en la que els vehicles estan buits respecte la distància total recorreguda per tots els vehicles

Nom instància	Vehicles utilitzats (k)	% de distància de vehicle compartit	Ocupació mitjana per unitat de distància (usuaris)	% de distància de vehicle buit	Temps d'execució algorisme
100-1H-2ST-5TW-VRP	**62**	0.01 %	0.92	8.22 %	13.33s
100-1H-2ST-10TW-VRP	**51**	11.86 %	1.07	4.77 %	11.63s
100-1H-2ST-20TW-VRP	**40**	34.36 %	1.35	1.98 %	13.02s
100-1H-4ST-5TW-VRP	**70**	0.0 %	0.92	8.15 %	9.74s
100-1H-4ST-10TW-VRP	**59**	1.3 %	0.91	10.13 %	10.04s
100-1H-4ST-20TW-VRP	**46**	27.06 %	1.21	5.82 %	7.10s
100-2H-2ST-5TW-VRP	**35**	0.0 %	0.8	19.99 %	14.62s
100-2H-2ST-10TW-VRP	**33**	7.88 %	0.88	20.17 %	15.28s
100-2H-2ST-20TW-VRP	**25**	23.4 %	1.1	14.31 %	11.66s
100-2H-4ST-5TW-VRP	**40**	0.0 %	0.81	18.68 %	17.34s
100-2H-4ST-10TW-VRP	**37**	0.5 %	0.78	22.34 %	16.68s
100-2H-4ST-20TW-VRP	**30**	22.13 %	1.06	16.63 %	12.06s
150-1H-2ST-5TW-VRP	**78**	0.1 %	0.89	10.94 %	38.64s
150-1H-2ST-10TW-VRP	**66**	8.65 %	0.99	9.44 %	26.73s
150-1H-2ST-20TW-VRP	**55**	30.57 %	1.28	4.66 %	26.15s
150-1H-4ST-5TW-VRP	**90**	0.0 %	0.91	9.42 %	41.08s
150-1H-4ST-10TW-VRP	**78**	1.69 %	0.91	10.88 %	39.40s
150-1H-4ST-20TW-VRP	**63**	24.0 %	1.2	4.72 %	17.04s
150-2H-2ST-5TW-VRP	**48**	0.05 %	0.77	23.22 %	28.17s
150-2H-2ST-10TW-VRP	**42**	6.69 %	0.86	20.49 %	21.82s
150-2H-2ST-20TW-VRP	**37**	21.88 %	1.08	14.78 %	28.30s
150-2H-4ST-5TW-VRP	**53**	0.0 %	0.77	23.24 %	25.92s
150-2H-4ST-10TW-VRP	**46**	0.76 %	0.73	27.69 %	10.24s
150-2H-4ST-20TW-VRP	**41**	15.86 %	0.98	17.59 %	28.67s

Taula 4.6: Execucions del disseny d'experiments.

Començarem per comentar l'efecte que té el factor de l'**amplada de la finestra temporal** sobre els resultats obtinguts. Clarament, es veu que a mesura que es fa més ampla aquesta finestra, els vehicles poden compartir més serveis i es passa de pràcticament un 0% de compartició per unitat de distància a tenir-ne un 20-30% (comparant els extrems de 5 min i 20 min d'amplada). Aquesta compartició de vehicle afavoreix de manera significant la reducció de vehicles utilitzats (figura 4.4). Tenir una amplada de 5 min de les finestres temporals significa donar un servei molt bo als usuaris ja que a la pràctica es recullen i es lliuren a la hora que han demanat i sense demores. Tanmateix, això restringeix generar rutes que permetin compartició de vehicle i flexibilitat per a que un vehicle pugui desviar-se per a atendre altres serveis. Tot i així, una finestra de 20 min que permeti aprofitar molt més els recursos pot provocar un pitjor servei incrementant el temps de viatge per als usuaris. Queda

a criteri dels *stakeholders* de negoci triar una configuració que beneficii a totes les parts implicades.

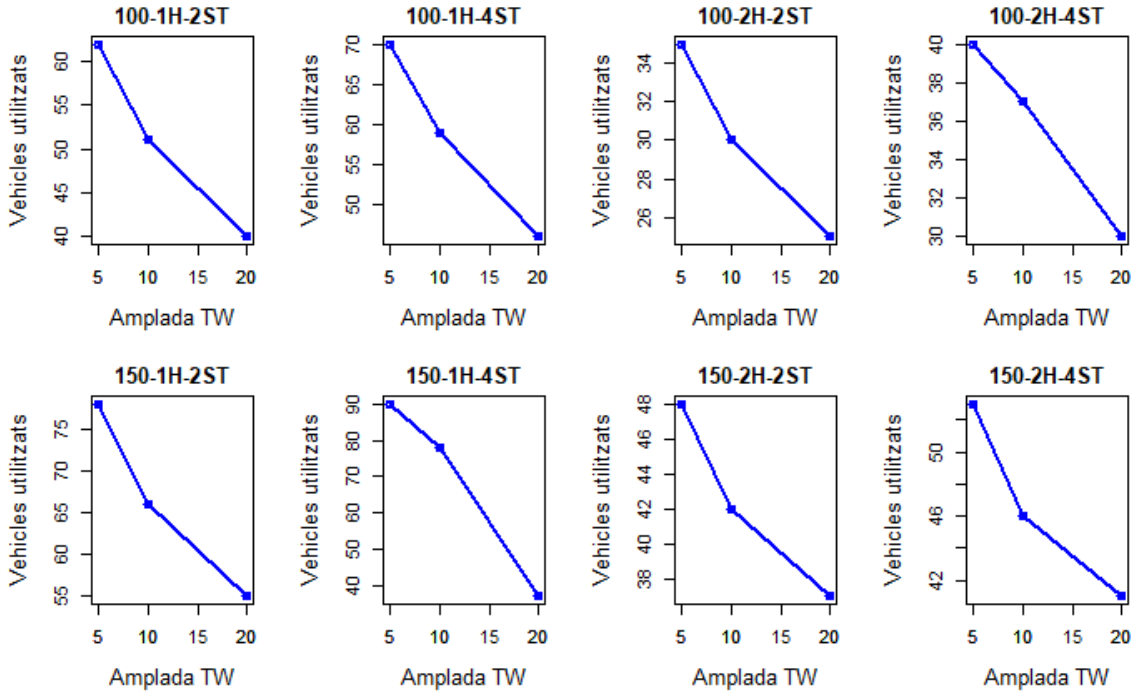

Figura 4.4: Evolució del nombre de vehicles utilitzats en funció de l'amplada de la finestra de temps per a les instàncies definides al disseny d'experiments.

Pel que fa al **temps de servei**, quan s'augmenta de 2 min a 4 min, s'observa que el resultats empitjoren. Aquest efecte és obvi, doncs és temps que l'algorisme considera que els vehicles estan quiets atenent a l'usuari i que no es troben circulant. Configurar un temps de servei elevat, pot resultar en donar millor servei a l'usuari ja que s'està procurant no subestimar el temps de recollida i lliurament (que en el cas d'alguns usuaris amb mobilitat molt reduïda pot ser força elevat), però, a canvi, això es veu lleugerament repercutit en utilitzar més recursos (vehicles) i menys aprofitats (menys compartició).

Modificar l'**horitzó temporal** ens serveix per veure com afecta als resultats que les peticions estiguin més o menys concentrades en un interval de temps. El patró que s'observa és que amb un horitzó temporal més ampli, és requereixen molts menys vehicles per a cobrir les peticions. Això és clarament degut a que si les peticions estan repartides en un interval de temps més ample, els vehicles poden encadenar més peticions i, en canvi, en un interval més estret, l'hora de les peticions són més concurrents i impossible de servir per un mateix vehicle.

4.3.3 Instància de 300 peticions

A part d'aquestes instàncies, també és vol provar un escenari similar al que s'executaria quan el sistema estigui en funcionament. Tal i com s'ha vist a la secció 4.1,

es realitzen uns 600 serveis fixos al dia. Per tant, una assignació de matí o de tarda consta, aproximadament, de 300 serveis cada una. Així doncs, les instàncies que es mostren a continuació consten de 300 peticions al llarg d'una mitja jornada laboral de taxi, amb un horitzó temporal de 4 hores. Els factors que es configuren són el temps de servei (amb nivells de 2 min i 4 min) i l'amplada de la finestra de temps (amb els nivells 5 min, 10 min i 20 min).

La taula 4.7 mostra els resultats d'aquests escenaris amb 300 peticions obtinguts amb l'algorisme.

Notem que el temps d'execució per a instàncies d'aquesta mida és al voltant d'un minut. L'impacte dels factors és similar a l'observada en les instàncies de 100 i 150 peticions.

Com que els 300 serveis es troben repartits en 4 hores, la càrrega mitjana de peticions és de 75 serveis/hora. Això pot haver causat que el % de distància de vehicle buit sigui més elevat degut a que les peticions estan més disperses en la línia temporal i els vehicles no poden compartir tants serveis ja que no succeeixen tan simultàniament. El mateix succeeix per als KPIs d'ocupació mitjana i de % de vehicle compartit, els quals també empitjoren per culpa d'aquest fenòmen.

En la realitat, les peticions es concentren més en unes franges que en altres (semblant a una distribució normal), però en la generació d'instàncies s'han distribuït uniformement en l'espai i en el temps amb la fi d'avaluar l'algorisme amb instàncies complicades que no segueixen ninguna mena de patró.

A la gràfica mostrada a la figura 4.5, es veu com l'algorisme millora la solució a cada iteració per a la instància de 300 peticions, 4 hores d'horitzó temporal, 2 minuts de temps de servei i 10 min d'amplada de finestra temporal.

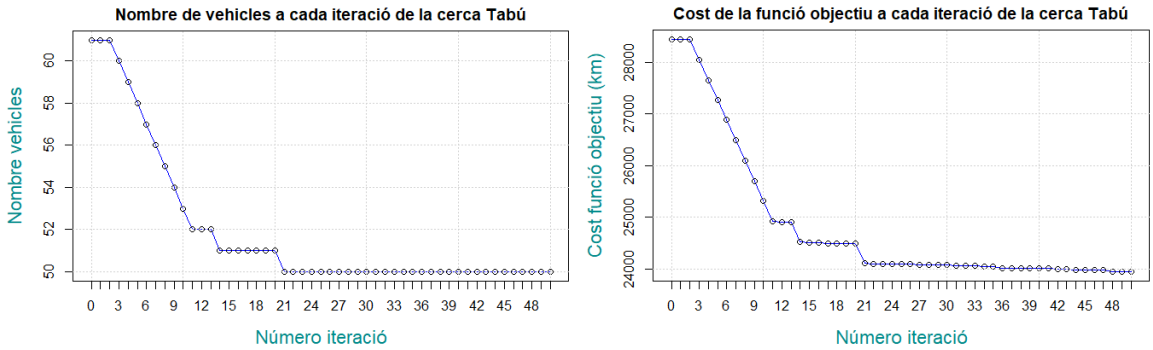

Figura 4.5: Gràfiques de l'evolució del nombre de vehicles (esquerra) i de la funció objectiu (dreta) en funció de nombre d'iteracions (instància 300-4H-2ST-10TW).

Observem que per a 300 peticions, l'heurística inicial constructiva aconsegueix encabir les peticions en 61 vehicles i després amb la Cerca Tabú s'aconsegueix millorar la solució a 50 vehicles.

A la figura 4.5 es veu que el fet de treure un vehicle (o el que és el mateix, buidar una ruta) de la solució fa disminuir significativament el cost de la funció objectiu.

Nombre peticions (n)	**Horitzó temporal**	**Temps de servei**	**Amplada TW (min)**	**Vehicles utilitzats (k)**	**% de distància de vehicle compartit**	**Ocupació mitjana per unitat de distància (usuaris)**	**% de distància de vehicle buit**	**Temps d'execució algorisme**
300 serveis	4 hores	2 min	5	**52**	0.0 %	0.69	30.74 %	64.06s
			10	**50**	4.41 %	0.75	29.01 %	74.03s
			20	**44**	21.17 %	1.0	21.35 %	54.14s
		4 min	5	**61**	0.0 %	0.69	31.13 %	52.77s
			10	**58**	0.05 %	0.71	29.5 %	62.34s
			20	**51**	12.49 %	0.89	23.31 %	65.94s

Taula 4.7: Resultats de la metaheurística de l'execució amb 300 peticions.

4.3.4 Visualització dels resultats

En aquesta secció es mostra els *outputs* de les rutes calculades per l'algorisme de la instància de 100 peticions `100-1H-2ST-10TW`. L'*output* complet de l'execució es pot trobar a l'annex B.

A continuació es mostra la planificació del vehicle número 44 (el qual és adaptat) i la seqüència de nodes que ha de visitar amb l'hora a la que hi ha d'arribar. Seguidament (figura 4.6) es mostra la ruta sobre la xarxa de transport.

```
VEHICLE # 44 (Adapted)
https://www.google.es/maps/dir/41.37845790717103,2.1063572531733166/41.37229630463679,2.
    0919588754533147/41.36705665887789,2.104060714791908/41.36394301138547,2.1052702014402063/41.
    36907243291208,2.169891238190284/41.372104617272015,2.1691091160374194/
Sequence of nodes:
(node 9) -> Pickup user 9 at 8:01
(node 109) -> Delivery user 9 at 8:12
(node 65) -> Pickup user 65 at 8:26
(node 17) -> Pickup user 17 at 8:35
(node 165) -> Delivery user 65 at 8:58
(node 117) -> Delivery user 17 at 9:06
```

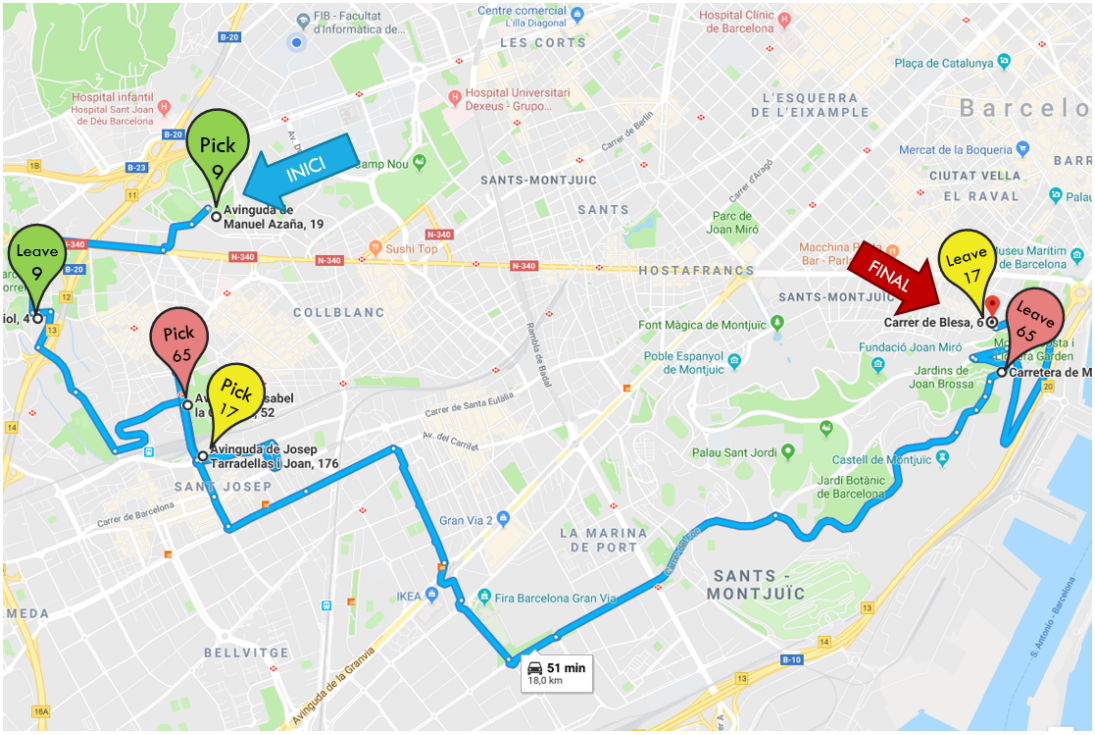

Figura 4.6: Ruta del vehicle núm. 44 per a la instància `100-1H-2ST-10TW`.

Convé destacar que a ruta de la figura 4.6 s'aconsegueix una compartició del vehicle significant.

Tot seguit, es mostra un altre exemple de planificació de ruta per a la mateixa instància del problema i la visualització respectiva sobre el mapa.

VEHICLE *# 47 (Adapted)*

https://www.google.es/maps/dir/41.382022701339054,2.1426726099068776/41.35578379623356,2.1361623252563455/41.35348463671869,2.134829036556043/41.41437675886846,2.219356777734983/41.453553976103414,2.1939977186499853/41.38587245766062,2.1225486267685705/

Sequence of nodes:

(node 43) -> Pickup user 43 at 7:57
(node 143) -> Delivery user 43 at 8:14
(node 32) -> Pickup user 32 at 8:18
(node 40) -> Pickup user 40 at 8:39
(node 132) -> Delivery user 32 at 8:53
(node 140) -> Delivery user 40 at 9:12

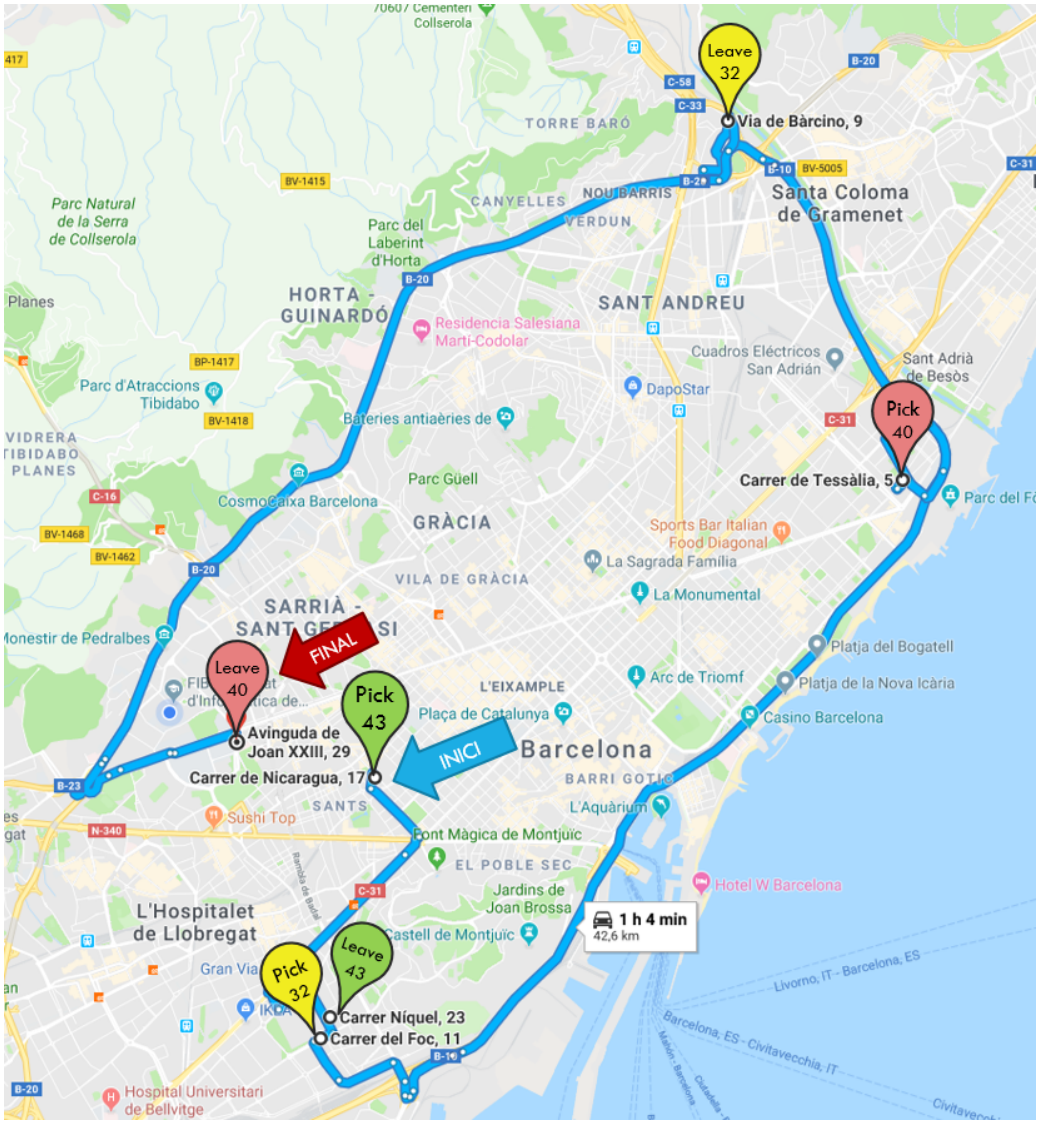

Figura 4.7: Ruta del vehicle núm. 47 per a la instància 100-1H-2ST-10TW.

Capítol 5

Proposta d'integració de l'algorisme

En aquest capítol es detalla la proposta d'integració del mòdul implementat en el sistema. Amb la intenció de poder reaprofitar aquest desenvolupament en altres projectes, s'ha anat un pas més enllà i s'ha realitzat una proposta d'arquitectura basada en microserveis. No obstant, la implementació de la proposta queda fora de l'abast del projecte.

5.1 Integració amb el sistema

Tant l'algorisme implementat com el sistema estan desenvolupats en el llenguatge de programació *Python*, això permet que el sistema faci ús directe de les funcions definides en el mòdul de l'algorisme. A més, el sistema ha sigut desenvolupat amb el *framework* web *Django*, el qual permet afegir i treure mòduls amb certa facilitat. Tot i així, es necessari definir la interfície d'ús de l'algorisme i l'adaptador necessari per a la transformació de dades del sistema.

Un exemple de com utilitzar i integrar el mòdul de manera directa al sistema és el que es mostra a continuació:

```
import VRP
import TabuSearch

# Es crea una instància del problema
VRP = VRPPDTW(requests=requests)
# o bé, si es té les matrius de distancia i temps en un fitxer extern
VRP = VRPPDTW(requests, duration='duration_matrix', distance='distance_matrix')

# S'introdueix la instància del problema a la metaheurística
TS = TabuSearch(VRP)

# Càlcul optimització
TS.solve()
result = TS.best_solution
```

Tal i com es mostra al codi anterior, en primer lloc es declara una instància del problema *VRPPDTW* amb les peticions com a paràmetre de construcció. El format de les peticions és una col · lecció de diccionaris amb les parelles de clau-valor mostrats a la secció 3.1.1. A continuació, s'utilitza la classe *TabuSearch* inicialitzant una instància amb el problema declarat anteriorment. En aquest moment, es pot executar la metaheurística mitjançant la crida *TabuSearch.solve()*. Un cop finalitzat aquest procediment, el resultat es guarda a l'atribut de *best_solution* de la classe *TabuSearch*. Per tal d'obtenir la solució en el format d'*output* especificat a la secció 3.1.2, cal cridar la funció *best_solution.to_dict()* per a transformar-la en format clau-valor.

També hi ha l'opció d'exportar la solució en format d'horari com es pot veure a continuació i de manera completa a l'annex B amb la funció *TabuSearch.schedule()* en un *output* més llegible.

```
SCHEDULE
-----------------------------------------------
VEHICLE # 1
https://www.google.es/maps/dir/41.37240192771526,2.1564798785901393/41.450726316296155,2.
    250007299151488/
(node 22) -> Pickup user 22 at 8:18
(node 122) -> Delivery user 22 at 8:48
-----------------------------------------------
VEHICLE # 2
https://www.google.es/maps/dir/41.38780519456423,2.1053433313478394/41.393302010237484,2.
    2029086725241056/
(node 14) -> Pickup user 14 at 8:34
(node 114) -> Delivery user 14 at 9:02
-----------------------------------------------
...
```

Com que el model de dades emprat en l'algorisme no correspon, ni en contingut ni en format, al del sistema, serà necessari desenvolupar un adaptador entre les dos peces, seguint un patró adaptador [24]. Aquest adaptador transformarà tant les dades d'entrada de l'algorisme com les de sortida. A més, utilitzar aquest patró de disseny facilita el desacoblament amb la interfície de l'algorisme.

La integració directa es pot realitzar afegint el codi en un fitxer dins del projecte i utilitzar-lo com una classe qualsevol: D'altra banda, també hi ha la opció de encapsular el codi com un *package* de *Python* per a ser gestionat amb PIP[1] i poder afegir aquest mòdul d'assignació i *routing* a la llista de dependències del sistema.

[1]sistema de gestió de paquets utilitzat per a instal · lar i administrar paquets de *software* escrits en *Python*.

5.2 Mòdul d'assignació i *routing* com un servei web

Una millora per a la portabilitat del mòdul de l'algorisme és la de canviar el model de distribució, per exemple, aplicar un disseny del tipus *Software as a Service* (SaaS). Aquest model de computació al núvol, implica abstraure el mòdul d'un llenguatge de programació concret i permetre el seu ús com a un servei de software a través de la xarxa.

Aquest disseny aporta diferents avantatges:

- Permet utilitzar una mateixa instància del servei en diferents projectes a l'hora.
- Es pot aplicar actualitzacions i modificacions sense afectar al funcionament dels clients que l'utilitzen.
- És accessible a través de la xarxa on estigui desplegat.
- Permet adaptar els requisits de hardware necessaris.
- És fàcilment escalable.

Per a transformar el mòdul en un servei web s'ha d'elevar la interficie d'ús, actualment en *Python*, a una interfície web. Aquesta pot seguir diferents estils com REST, SOAP o RPC. Serà necessari fer una documentació d'especificació per a que es pugui utilitzar el servei. A nivell de codi s'ha de fer un *wrapper* que transformi les sol · licituds i dades web en el format i la interfície de les funcions *Python*.

5.2.1 Tecnologies

Amb l'objectiu de transformar el mòdul en un servei web es proposa l'ús de les següents tecnologies:

- El *microframework* web *Flask* per al *wrapper* de l'algorisme.
- *Swagger* com a eina de documentació de la interfície web.
- *Docker* com a eina d'encapsulament del servei.

En la figura 5.1 es visualitza un esquema de les tecnologies implicades en el disseny.

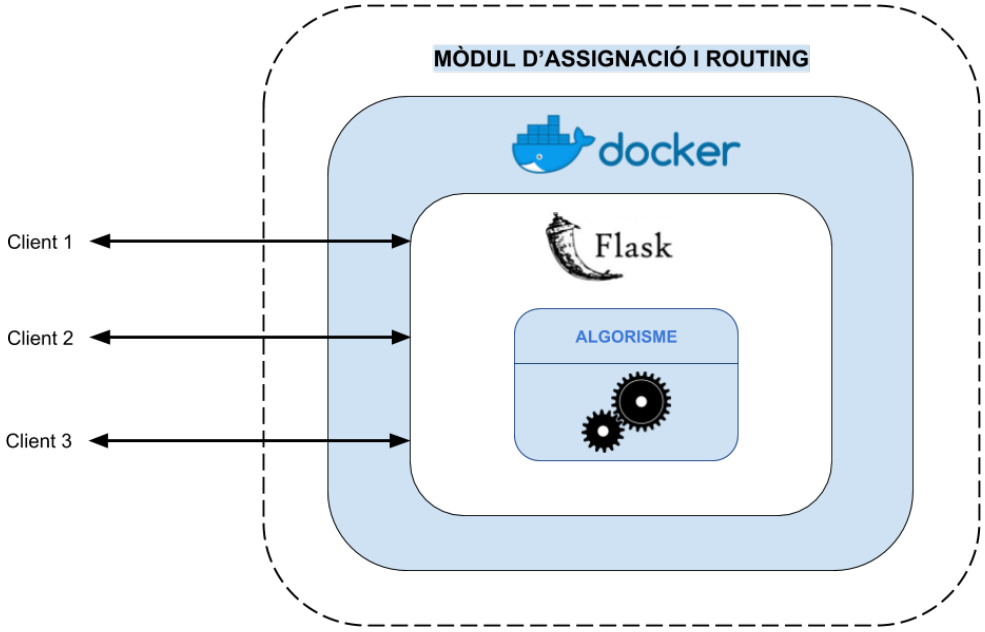

Figura 5.1: Esquema de la proposta d'arquitectura del mòdul com a servei web.

5.2.2 Escalabilitat

L'escalabilitat fa referència a l'habilitat d'incrementar la capacitat de processar peticions. En el cas dels serveis *web* es pot escalar afegint més instàncies del servei i repartint les tasques.

Una altre possibilitat és la d'incrementar el volum sota demanda, és a dir, aixecar i destruir instàncies del servei web segons es necessitin. Aquest procés és molt àgil si el servei s'encapsula en una imatge *Docker*.

Capítol 6

Gestió del projecte

6.1 Planificació temporal

6.1.1 Planificació del projecte global

La duració estimada del projecte sencer és de 10 mesos, des de principis de Febrer del 2018 fins a finals de Desembre del 2018, que és quan s'ha de finalitzar el producte. Durant el més d'Agost s'aturarà el projecte degut a les vacances d'estiu.

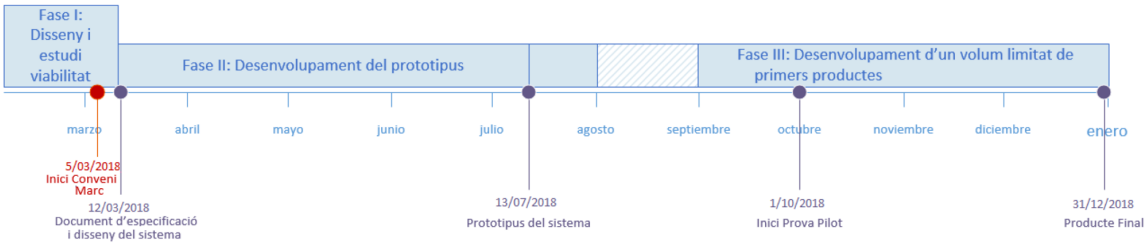

Figura 6.1: Planificació temporal del projecte global.

- **Fase I - Disseny i estudi viabilitat**: Reunions amb els clients, planificació temporal i formació de l'equip.
- **Fase II - Desenvolupament del prototipus**: Durant aquesta fase es realitzarà principalment el desenvolupament *software* del sistema que es presentarà a una prova pilot a principis d'Octubre. Comença un cop signat el conveni del projecte i lliurada l'especificació i disseny del sistema.
- **Fase III - Desenvolupament d'un volum limitat de primers productes**: En aquesta fase, la part principal del producte estarà desenvolupada i és aquí quan es realitzarà l'optimització de l'assignació del serveis fixos que suposa el tema troncal d'aquest TFG.

6.1.2 Planificació de la part del projecte tractada al TFG

Com ja s'ha explicat a la secció d'*Abast*, la part que es tracta en aquest TFG és la part d'assignació i optimització de rutes del servei fix, que es desenvoluparà al llarg de tota la **Fase III** de la planificació.

Tenint això en consideració, aquest TFG se centra principalment en la Fase III i a continuació es descriuran les tasques i la planificació pertinents a aquesta fase. La durada de la Fase III és de 4 mesos, des de principis de Setembre 2018 fins al 30 de Desembre de 2018 (pel que fa al projecte a nivell de negoci) però s'allargarà fins al 26 de Gener 2019 (abans de la lectura del treball) per a acabar de documentar i preparar la presentació final. Per tant, en total, el projecte del TFG durarà prop de 5 mesos.

Seguint les metodologies àgils basades en *Scrum* que s'utilitzen al projecte, la Fase III s'organitzarà en 5 - 6 *sprints* d'unes 3 setmanes de durada on s'encabiran les taques adaptant-se a l'evolució temporal del projecte.

6.1.3 Descripció de tasques

En aquesta secció es descriuran les tasques principals que s'han plantejat per tal de realitzar el projecte, que etiquetarem amb $T1, \ldots, Tn$.

Planificació i organització del projecte (T1)

Aquesta tasca és la que es cobreix amb l'assignatura de GEP, que consisteix en acotar i planificar el projecte, així com també analitzar els requisits i avaluar la seva viabilitat. Aquesta tasca durarà un mes aproximadament i es pot subdividir en les següents subtasques:

- **Reunions amb l'equip i clients**: Se'n realitzaran al llarg de tot el desenvolupament, però en especial a l'inici de la Fase III per al *kick-off* del projecte.
- **Definició de l'abast**.
- **Planificació temporal**.
- **Planificació econòmica i sostenibilitat**.

Adquirir coneixement sobre la literatura actual del VRP (T2)

Abans de començar a desenvolupar algorismes, es necessitarà una preparació prèvia per tal d'adquirir coneixement sobre `VRP` i com es tracta actualment. A partir d'aquesta recerca, podrem modelar el problema, seleccionar els algorismes més adients i decidir amb quines eines o llenguatges de programació treballar segons els mètodes que triem per resoldre'l.

Aquesta tasca consisteix en llegir bibliografia recomanada sobre el tema, i posteriorment indagar en articles més actuals sobre el problema. Aquesta fase pot prendre al voltant d'un mes, però pot fer-se paral · lelament a la *Planificació temporal*.

Formalització matemàtica del problema (T3)

En aquesta tasca hem de generar un model d'optimització del `VRP` adaptat al cas dels serveis fixos, amb el que, tal i com hem vist a la secció de formulació del problema, hem d'ajuntar les variants del `VRP` (`VRPPD`, `VRPTW` i `HVRP`) en una formulació única (`HVRPPDTW`). Aquesta és una de les tasques més complexes ja que els models i formalitzacions d'aquests problemes existeixen per separat i s'hauran de combinar correctament i de manera adequada.

Cal remarcar que és requisit per a la resta del projecte i, si no es realitza adequadament, s'haurà de retrocedir fins a finalitzar-la correctament.

Desenvolupament dels algorismes (T4)

Aquesta és una de les tasques principals del projecte, i consisteix en desenvolupar els algorismes que s'hagin seleccionat en les anteriors tasques. Posteriorment, els resultats d'aquests algorismes s'analitzaran i es contrastaran per determinar quins funcionen millor. A continuació enumerem les subtasques implicades en ella:

- **Programació algorismes**: desenvolupar els mètodes en codi font.
- **Anàlisi i contrast de mètodes**: comparar els mètodes a nivell de complexitat temporal i recursos.
- ***Testeig***: validar amb *tests* els algorismes per a garantir el correcte funcionament.

Integració amb el sistema (T5)

Consisteix en incorporar el mòdul desenvolupat al sistema global i adaptar-lo per a que un usuari del sistema pugui introduir les dades per a generar assignacions del servei fix. Podem distingir dues sub tasques:

- **Vista web**: desenvolupar una vista web on els usuaris administradors puguin introduir les dades de les sol · licituds dels serveis fixos.
- ***Back-end***: Dins el sistema, persistir la informació i els resultats de l'algorisme a la base de dades del sistema.

Documentació (T6)

La principal labor d'aquesta tasca serà documentar el projecte de cara a la memòria del TFG i preparar la presentació per a la seva lectura. Aquesta activitat es realitzarà transversalment a la resta de tasques i durant tota Fase III del projecte.

Recursos

Per a les taques que s'han descrit anteriorment, es necessiten una sèrie de recursos els quals podem dividir en recursos humans, de *hardware* i de *software*. Tot seguit els llistem i indiquem a quines tasques corresponen.

Recursos humans

- Product-Owner: necessari per a totes les tasques, però en especial a la *Planificació i organització del projecte* (**T1**).
- Equip de desenvolupament: format per 2 integrants: 1 desenvolupador sènior o cap tècnic expert en tecnologies i 1 desenvolupador júnior (autor del TFG).
  - Desenvolupador Sènior: necessari per a les tasques **T1** i **T5**.
  - Desenvolupador Júnior: necessari a totes les tasques.
- Expert en mobilitat i transport (*routing*): necessari a les tasques **T1**, **T2**, **T3** i **T4**.

Recursos *hardware*

- Ordinador personal (DELL Optiplex 7020, Intel Core i7, 8GB RAM, 512GB SSD): necessari a totes les tasques. Serà on s'implementin s'avaluïn els algorismes
- Servidor del sistema: necessari per a la *Integració amb el sistema* (**T5**).

Recursos *software*

- *Windows 10*: a totes les tasques.
- *Ubuntu*: sistema operatiu del servidor, per a la tasca **T5**.
- *Django Framework*: usat a la tasca **T5**.
- *Python*: usat a la tasca **T4** i **T5**.
- *Angular 5*: usat a la tasca **T5**.
- Editors de text (*PyCharm*, *WebStorm* i *Visual Studio Code*): tasques **T4** i **T5**.

- *Docker*: usat a la tasca **T5**.
- LaTeX: usat a totes les tasques.

Altres

- Bibliografia : utilitzada principalment per a les tasques **T2**, **T3** i **T4**.

6.1.4 Estimació temporal

La taula 6.1 resumeix el temps estimat de cada una de les tasques descrites en la secció anterior. Algunes compten amb la participació de 6 persones de l'equip de desenvolupament i del product-owner (en total 7 persones).

	Estimació de temps (hores)			
Tasca	**Desenvolupador Júnior**	**Product Owner**	**Expert en routing**	**Desenvolupador Sènior**
T1: Reunions amb l'equip	30	30	30	30
T1: Definició de l'abast	10	10	10	0
T1: Planificació temporal	10	2	0	0
T1: Planificació econòmica i sostenibilitat	10	2	0	0
T2: Adquirir coneixement sobre la literatura actual del `VRP`	60	0	10	0
T3: Formalització matemàtica del problema	80	0	10	0
T4: Programació algorismes	120	0	10	0
T4: Anàlisi i contrast de mètodes	30	0	10	0
T4: *Testeig*	20	0	0	0
T5: Vista Web	20	0	0	5
T5: *Back-end*	20	0	0	5
T6: Redactar memòria	100	0	0	0
T6: Preparar presentació	20	0	0	0
Total	**530 hores**	**44 hores**	**80 hores**	**40 hores**

Taula 6.1: Estimació de temps per a les tasques.

Les tasques T2, T3, T4, T5 i T6 les realitzarà només l'autor d'aquest TFG (Desenvolupador Júnior), que juntament amb la T1 consistiran en **530 hores** de feina. L'estimació temporal de l'equip sencer és de **694 hores** per a aquesta part del projecte.

6.1.5 Diagrama de Gantt

Per a generar el diagrama de *Gantt*, hem de detectar les dependències causals entre les diferents tasques que hem llistat anteriorment.

- T2<T3; T2<T4; T2<T5
- T3<T4; T3<T5
- T4<T5

A la Figura 6.2 podem veure el diagrama de *Gantt* resultant.

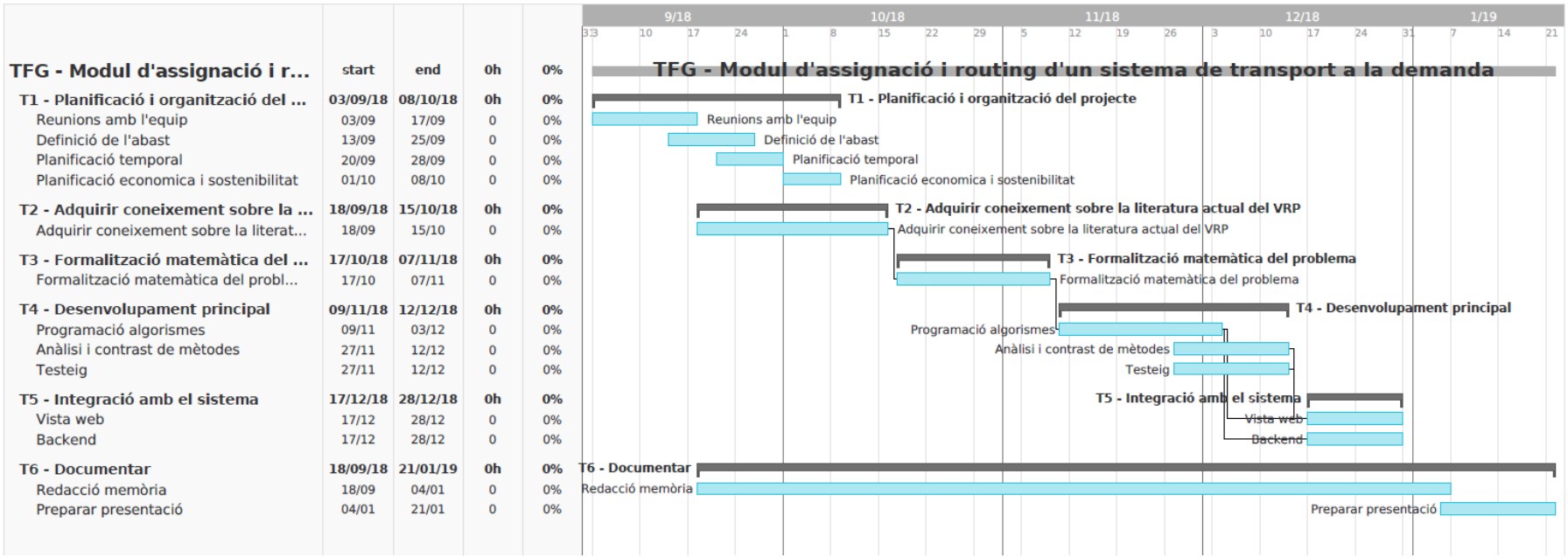

Figura 6.2: Diagrama de Gantt del projecte (generat amb TeamGantt).

6.1.6 Pla d'acció i alternatives

Com s'ha explicat a la secció de *Metodologies*, s'utilitzaran tècniques àgils basades en *Scrum* que ens permetran revisar i adaptar dinàmicament la planificació inicial, és a dir, que es modificarà la planificació acord amb la durada real de les tasques.

A continuació es mencionen motius i situacions que podrien causar endarreriments en la planificació inicial, i es proposen alternatives per a garantir arribar als objectius principals del projecte.

Complexitat de la formalització

Realitzar la formalització matemàtica del problema real de flotes implica haver-se documentat en profunditat sobre el tema i haver assimilat la notació matemàtica.

Estimar el temps d'aquesta tasca és difícil i ve donat pel nivell de comprensió que es tingui sobre el tema. Per a compensar possibles impediments temporals, podem donar més temps de sobreestimació a aquesta tasca i així assegurar no excedir el temps planificat inicialment.

Bugs algorisme

Degut a les moltes restriccions que tindrà el nostre problema que el fan relativament complex, és fàcil cometre algun error a mesura que es va modificant el codi font. Això pot causar retards si els *bugs* no es troben i solucionen ràpidament. Per això és important organitzar bé el codi i testejar-lo a cada canvi que s'introdueixi i evitar errors inesperats.

Selecció dels mètodes a desenvolupar

En cas de patir un endarreriment respecte la planificació establerta i no donés temps a comparar diversos mètodes, s'haurà d'escollir un únic mètode que segons l'estat de l'art sigui una aposta segura i estable per poder garantir la implementació d'almenys un algorisme que resolgui el problema.

Dependència amb la resta del sistema

Tal i com es va explicar al la secció d'*Obstacles*, existeix una dependència amb la resta del sistema global per a poder integrar-hi el mòdul algorísmic desenvolupat. Si el sistema no funcionés o estigués inacabat, una de les solucions podria ser oferir als usuaris administratius una interfície separada del sistema per a poder introduir i calcular les rutes per als serveis fixos, aprofitant que la vista *web* s'ha desenvolupat per a aquest TFG, o bé deixar preparat el mòdul dins un contenidor virtual que faciliti una posterior integració amb el sistema.

D'altra banda, per tal de no arribar a aquesta situació, com a integrant de l'equip de desenvolupament podria assumir encara més hores del rol de desenvolupador *software*.

D'aquesta manera s'avançaria més feina per a arribar al sistema final, però, per contrapartida, em restaria temps de dedicació al mòdul d'algorísmic.

6.1.7 Desviacions temporals

Durant la Fase II del projecte van quedar deutes tecnològics que van fer endarrerir la posada en marxa de la Fase III. Degut a això, l'especificació dels serveis fixos amb el client no es va acabar de definir fins més tard, cosa que va dificultar la posada en marxa de les tasques relacionades amb els serveis fixos d'`IMPD`. Això ha causat que la tasca T3 (*Formalització matemàtica del problema*) i, per conseqüència, les que depenen d'ella, T4 i T5, es veiessin postposades respecte la planificació inicial.

Per a que el mòdul d'assignació i *routing* estigués llest de cara a l'entrega del TFG, s'ha decidit començar amb la informació justa que el client a pogut proporcionar sobre com operen amb els serveis fixos i implementar-lo tenint en compte de facilitar la modificació i extensió de les seves funcionalitats.

D'altra banda, si l'equip de desenvolupament no va alineat amb la planificació temporal del TFG, s'haurà de deixar el mòdul preparat per tal de facilitar a posteriori la integració amb el sistema global.

6.2 Planificació econòmica

6.2.1 Identificació de costos

En aquesta secció es realitzarà una estimació del pressupost necessari per a implementar la part del projecte tractada en aquest TFG. Dividirem el pressupost en tres seccions, depenent del tipus de recursos implicats:

- Recursos humans
- Recursos *hardware*
- Recursos *software*

Com que es tracta d'un projecte en el que no es requereix cap *hardware* ni *software* especial, el gruix dels costos estarà localitzat en els recursos humans.

Al final, es mostrarà el pressupost total sumant tots els recursos, les contingències i les amortitzacions.

6.2.2 Estimació de costos

Recursos Humans

A la part dels recursos humans només ens centrarem al personal contractat per a desenvolupar la Fase III (Desenvolupament d'un primer producte o servei en forma

de sèrie de prova). El desenvolupament de la Fase III tindrà una durada de 4 mesos, des del 01/09/2018 fins el 30/12/2018 i els recursos assignats seran:

- 1 Cap de projecte o *Product-owner*
- 1 Expert en mobilitat i transport (*routing*)
- 1 Programador Sènior o *Scrum* màster
- 1 Programador Júnior o Desenvolupador (l'autor d'aquest TFG)

Les tarifes de cada rol que es mostren a la taula 6.2 són exactament les que s'aplicaran ja que provenen de la planificació econòmica que ens ha proporcionat la cap projectes de l'`inLab FIB` per a aquest projecte.

Personal/Rol	**Quantitat**	**Dedicació (hores)**	**€/hora**	**Cost**
Programador Júnior	1	530	12,5€	6625€
Expert en routing	1	44	54,13€	2381,72€
Product Owner	1	80	43,24€	3459,20€
Programador sènior	1	40	37,83€	1513,20€
Total				**13979,12€**

Taula 6.2: Estimació dels costos de recursos humans

Recursos *Hardware*

A continuació es mostren els dispositius *hardware* que s'han utilitzat a llarg de la Fase III del projecte. Cal remarcar que aquest *hardware* ha estat i serà aprofitat per a altres projectes dins l'`l'inLab FIB`, per tant s'indicarà el cost durant el temps d'us i les respectives amortitzacions. Pel que fa al temps d'ús, tindrem en compte la durada les projecte del TFG, que és de 5 mesos.

Hardware	**Quantitat**	**Preu**	**Temps d'ús**	**Vida útil**	**Amortització**
PC *Dell Optiplex* (512GB SSD, 8GB RAM)	1	1000€	5 mesos	5 anys	83,33 €
Servidors del sistema	1	1596€	5 mesos	1 any	665 €
Total		**2596€**			**748,33€**

Taula 6.3: Estimació dels costos de recursos hardware.

Recursos *Software*

La següent taula mostra el *software* de pagament que s'ha utilitzat per al desenvolupament. El programari gratuït (com ara *Python*, *Angular 5*, *Django*, LaTeX, *Ubuntu*, *Docker*) queda omès de la taula 6.4.

Software	Quantitat	Preu	Temps d'ús	Vida útil	Amortització
Llicències de *PyCharm Professional*	1	199€	5 mesos	1 any	82,92 €
Llicències de *WebStorm*	1	129€	5 mesos	1 any	53,75 €
Llicències d'eina de gestió *YouTrack*	1	750€	5 mesos	1 any	312,50 €
Windows 10 *Enterprise*	1	200€	5 mesos	1 any	83,33 €
Total		**1278€**			**532,50€**

Taula 6.4: Estimació dels costos de recursos software

Imprevistos

A la taula 6.5 es mostren els possibles imprevistos i amb la probabilitat de que succeeixin. Això permet tenir un pressupost extra de marge per a enfrontar problemes inesperats que puguin sorgir durant el projecte.

Tipus	Cost	Probabilitat	Cost eventual
Averia PC *Dell Optiplex*	1000€	5,0%	50€
Averia Servidor del Sistema	1596€	0,5%	7,98€
Total			**57,98€**

Taula 6.5: Costos causats per imprevistos

Costos indirectes

Com a costos indirectes, s'han tingut en compte el cost de contractar un servei d'accés a Internet i despeses en transport per a desplaçar a part de l'equip a reunions amb el client. Pel que fa a al cost d'Internet (40,00€mensuals) tan sols es comptarà la part proporcional a les 4 persones que participen en aquesta part del projecte respecte dels 70 treballadors de l'`inLab FIB`, que representen el 5.71% del personal. Respecte als costos de transport, es tindra compte que hi ha uns 5 *sprints* que poden comportar reunions, de les quals es pressuposten 30€ per a l'anada i tornada a una reunió. El còmput total queda: 5 *sprints* × 30€/viatge = 150€.

Tipus	Cost
Internet	11,43€
Transport	150€
Total	**161,43€**

Taula 6.6: Costos indirectes

6.2.3 Control de gestió

Hi haurà un control constant per a preveure desviacions en el projecte. Per a l'estimació total, degut a que es tracta l'un projecte de llarga durada, s'ha decidit guardar un percentatge extra del subtotal obtingut dels costos directes i indirectes, en concret un 10%, per a possibles contingències.

Tot plegat, el cost total d'aquesta fase serà el que es mostra a la següent taula.

Tipus	**Cost**
Personal	13.979,12€
Hardware	748,33€
Software	532,50€
Indirectes	161,43€
Imprevistos	57,98€
Subtotal	**15.479,36€**
Contingències (10%)	1.547,94€
Total	**17.027,30€**

Taula 6.7: Costos totals del projecte

6.3 Sostenibilitat

En aquesta secció avaluarem la sostenibilitat del projecte en tres àrees diferents: l'econòmica, la social i l'ambiental. A cada apartat es comentarà com afecta a cada un dels àmbits durant el desenvolupament i a llarg del temps de vida del sistema.

6.3.1 Domini de la competència de sostenibilitat

Després de realitzar l'enquesta sobre Sostenibilitat s'ha arribat a la conclusió de que conèixer els tres aspectes; social, econòmic i ambiental és clau en el desenvolupament de projectes TIC. També és molt important l'apartat de creativitat i innovació per tal d'aportar solucions que funcionin en aquestes tres dimensions. Crec que és necessari aprofundir en aquests temes per tal de garantir una solució sostenible, no només per aquest projecte, si no per a tota la resta. Cal remarcar que nosaltres que tenim un perfil més tècnic d'enginyers no dominem tant aquest tipus de coneixements. A més a més, és necessari saber valorar l'impacte tant positiu com negatiu que els productes i els serveis TIC tenen sobre la societat i saber utilitzar els indicadors adients per a mesurar els efectes que generen de forma holística.

En quant a la part de planificació econòmica és necessari conèixer les eines per realitzar una bona gestió. Tinc familiaritat amb els termes d'amortitzacions, costos fixos i variables i eines de treball però, per poder treballar de forma eficient amb aquestes conceptes cal aprofundir en el seu coneixement.

Respecte a les eines de gestió col · laborativa i organització, tinc experiència prèvia d'altres projectes. He fet ús anteriorment d'eines de treball en equip com ara *Trello*, *Taiga*, o *YouTrack*.

En aquest projecte els usuaris finals són persones amb mobilitat reduïda, que poden tenir algun tipus de discapacitat que els dificulti la interacció amb el sistema, per tant, s'ha de tenir en compte els aspectes d'accessibilitat, ergonomia i seguretat de les solucions tecnològiques.

6.3.2 Dimensió econòmica

Per al desenvolupament s'han utilitzat tecnologies i software *open-source* que no suposen cap cost. L'únic programari que no es és el Sistema Operatiu utilitzat a l'ordinador personal. Es podria utilitzar un SO diferent a *Windows 10* com ara alguna distribució de *Linux*, però és el que s'utilitza als ordinadors de l'organització on s'ha desenvolupat el projecte.

Un dels objectius d'aquest projecte era aprofitar millor la flota dedicada als serveis fixos, amb la qual cosa es pretén oferir una millora en la dimensió econòmica del servei actual. Per altra banda, les operadores poden perdre benefici ja que deixaran de planificar les seves rutes que que fins ara no eren tan optimitzades i per tant més cares.

6.3.3 Dimensió ambiental

Els recursos usats en el projecte són els que s'han detallat a la secció de *Gestió econòmica* i a la de *Planificació temporal*.

Per al desenvolupament del projecte s'utilitzarà l'ordinador personal i el servidor al final per integrar el modul al sistema global. Els únics recursos que s'utilitzaran seran aquests dispositius, l'energia que consumeixes i el paper utilitzat per a imprimir documentació. També s'hauran de tenir en compte alguns possibles desplaçaments en transport per a les reunions amb el client.

Amb les prestacions del l'ordinador personal utilitzat, que és el que ha estat pràcticament cada dia encès (500 hores), consumeix uns 286kWh.

Durant el temps de vida del sistema, l'optimització de rutes pretén aprofitar millor els recursos dels serveis per a encabir més viatges a la flota que opera el servei fix, els quals són viatges que no es realitzaran en altres vehicles. D'aquesta manera s'està reduint el consum d'altres medis de transport.

6.3.4 Dimensió social

A nivell social, la part que es desenvolupa en aquest TFG té un impacte molt significatiu, ja que ajuda a millorar el funcionament d'un servei públic que té com a usuaris un col · lectiu en risc d'exclusió social.

Amb el nou sistema els *stakeholders* que impulsen el projecte podran oferir més serveis dedicant el mateix nombre de recursos pressupostats. Així, hi haurà capacitat per servir a un major nombre de persones amb mobilitat reduïda.

A més, aquest treball contribueix al tema del transport a la demanda, el qual és tòpic molt estudiat actualment com a una nova forma de transport per a millorar els problemes de trànsit de les ciutats d'avui dia.

Durant el desenvolupament, s'ha treballat en un entorn col · laboratiu i multidisciplinari. Això ha afavorit el creixement personal dins d'un àmbit laboral i social.

6.3.5 Matriu de sostenibilitat

	Econòmica	Ambiental	Social
Valor	8	8	9

Taula 6.8: Matriu de sostenibilitat

Capítol 7

Integració del coneixement

7.1 Coneixements de les assignatures

A continuació es llisten les assignatures de l'especialitat de computació i els coneixements que s'han aplicat per a realitzar el projecte de cada una d'elles.

Algorísmia (A)

Aquesta assignatura ha proporcionat habilitat per a realitzar l'anàlisi de complexitat (temporal i de memòria) sobre algorisme. També sobre nocions de programació lineal.

Lògica a la Informàtica (LI)

Aporta coneixement sobre problemes `P, NP, NP-Complet, NP-Hard` i del problema de satisfabilitat booleana `SAT`, que és el problema `NP-Complet` més bàsic el qual es pot reduir al `VRP` (problema troncal d'aquest projecte). Servirà per a conèixer la *naturalesa* del problema al que ens enfrontem, quinesformes hi ha de resoldre'l.

Intel · ligència Artificial (IA)

A l'assignatura d'Intel · ligència Artificial s'hi treballen els mètodes de cerca local i heurístiques com ara *Simulated Annealing*, Algorismes genètics o *Tabu Search*. Aquests mètodes serviran per a obtenir resultats propers a l'òptim d'un problema `NP-Hard` (com el `VRP` que hem de resoldre) amb un temps i recursos raonables.

Ampliació d'Algorísmia (AA)

En aquesta assignatura obtenim una visió més àmplia sobre les diferents classes de complexitat i reduccions entre problemes `NP-Complet`, `NP-Hard`. També proporciona una maduresa en l'anàlisi de complexitat algorísmica. S'hi treballa la formalització de problemes d'optimització en forma de Programació Lineal (entera i no entera),

que ens serà útil per a la formalització del problema de serveis fixos. Un nou tema que s'introdueix en aquesta assignatura i d'utilitat per al nostre problema és el dels algorismes d'aproximació (entre ells, la relaxació de la Programació Lineal Entera).

7.2 Justificació especialitat de computació

El projecte tracta de formalitzar un cas real de servei de flotes com una variant del problema del `VRP`. A més, s'ha de realitzar un estudi de quin mètode utilitzar per a resoldre'l, implementar l'algorisme i integrar-lo en un sistema. Tot aquest procés tracta de manera directa els següents temes:

- **Classes de complexitat** (`P, NP, NP-Complet, NP-Hard`) i reduccions entre problemes.
- ***Vehicle Routing Problem* (`VRP`) i les seves variants**. Tal i com hem vist, no hi ha cap problema estandard del `VRP` que encaixi amb el cas les serveis fixos, per tant s'ha de formalitar el problema com a una variant juntant la HVRP, VRPPD i VRPTW en una sola: *Heterogeneous Vehicle Routing Problem with Pick Up and Delivery and Time Windows* (`HVRPPDTW`).
- **Optimització i formalització de problemes**: Programació Lineal (`LP`) i Programació Lineal Entera (`ILP`). La formalització es representarà com un problema de programació lineal entera de minimització.
- **Algorismes d'aproximació** (heurístiques, relaxació lineal). Es realitzarà un estudi de les heurístiques actuals que utilitzen per a resoldre el `VRP`, com ara algorísmes genétics, *simulated annealing* o cerca *Tabu*.

Tots aquests temes que es tracten al projecte estan lligats a l'àmbit de les Ciències de la Computació amb la qual cosa es justifica que el projecte s'adeqüi a l'especialitat de Computació.

7.3 Competències tècniques

A continuació es mostren les competències tècniques associades a aquest projecte.

CCO1.1: Avaluar la complexitat computacional d'un problema, conèixer estratègies algorísmiques que puguin dur a la seva resolució, i recomanar, desenvolupar i implementar la que garanteixi el millor rendiment d'acord amb els requisits establerts. [Nivell: En profunditat]

CCO1.2: Demostrar coneixement dels fonaments teòrics dels llenguatges de programació i les tècniques de processament lèxic, sintàctic i semàntic associades, i saber aplicar-les per a la creació, el disseny i el processament de llenguatges. [Nivell: En profunditat]

CCO2.2: Capacitat per a adquirir, obtenir, formalitzar i representar el coneixement humà d'una forma computable per a la resolució de problemes mitjançant un sistema informàtic en qualsevol àmbit d'aplicació, particularment en els que estan relacionats amb aspectes de computació, percepció i actuació en ambients o entorns intel · ligents. [Nivell: En profunditat]

CCO3.1: Implementar codi crític seguint criteris de temps d'execució, eficiència i seguretat. [Nivell: En profunditat]

Justificació i nivell d'assoliment

En aquest apartat justificarem el per què s'han escollit cada una de les competències tècniques i com es preveu assolir el nivell d'assoliment.

CCO1.1

Aquesta competència s'ha escollit ja que per a realitzar el projecte s'haurà d'analitzar la complexitat del problema abans de triar quin mètode utilitzar per a resoldre'l. A partir de conèixer a quin problema ens estem enfrontant, es realitzarà un estat de l'art sobre les metodologies actuals que s'utilitzen i implementar-ne les més adients.

El nivell d'assoliment serà en profunditat ja que és la part principal d'aquest treball i s'haurà d'implementar un o més mètodes que garanteixin el millor rendiment segons els requisits establerts.

CCO1.2

En funció dels mètodes triats per a resoldre el problema, s'haurà de valorar si és necessari utilitzar llenguatges de programació de més o menys alt nivell per motius de rendiment.

Com que aquest mòdul estarà desacoblat del sistema global i pot funcionar a mode de servei, podem valorar quin llenguatge de programació és més adient en funció del seu funcionament i processament intern. Per aquest motiu, el nivell d'assoliment serà en profunditat.

CCO2.2

Aquesta competència s'ha triat ja que és la part fonamental del projecte, doncs s'ha de formalitzar un problema real a una forma computable. En aquest cas s'ha de representar el servei fix del Servei Public de Transport Especial com una variant del `VRP` per a poder obtenir rutes òptimes.

Donat que és important realitzar correctament aquesta part per a que la resta del projecte sigui factible, el nivell d'assoliment esperat serà en profunditat.

CCO3.1

El codi a implementar ha de poder obtenir solucions de manera eficient per a poder tenir una planificació del servei fix en un temps raonable. A més s'ha de desenvolupar un codi íntegre que no doni lloc a errors.

Per aquests motius, com que l'eficiència i seguretat del codi són fonamentals de cara al producte final, aquesta competència s'assolirà en profunditat.

Nivell d'assoliment

En resum, el nivell d'assoliment de les competències tècniques és el que es mostra a la següent taula.

	CCO1.1	**CCO1.2**	**CCO2.2**	**CCO3.1**
En profunditat	×	×	×	×
Suficientment				
Una mica				

Taula 7.1: Nivell d'assoliment de les competències tècniques d'especialitat

Capítol 8

Conclusions

En aquest capítol se recullen les conclusions del treball realitzat. En primer lloc, s'avalua l'assoliment dels objectius proposats a l'inici del projecte. A continuació, s'expliquen les contribucions que aporta la feina realitzada en aquest Treball de Fi de Grau i les conclusions personals que s'han extret. Finalment, s'indiquen alguns punts de treball futur que podrien que podrien millorar o estendre el mòdul que s'ha desenvolupat.

8.1 Assoliment d'objectius

En aquesta secció, es fa una recapitulació de quin són els objectius d'aquest treball i s'avalua l'assoliment de cada un d'ells.

A continuació, recordem els objectius que es van proposar a l'inici del projecte.

- ✓ Descriure la formulació matemàtica del problema per al cas específic dels serveis fixos que ofereix el Servei Públic de Transport Especial.
    - Aquest objectiu s'ha assolit correctament. A la secció 2.1.4 s'explica com s'ha adaptat el model d'optimització matemàtic general, que resol el PDPTW, al cas particular dels serveis fixos del Servei Públic de Transport Especial. A més, a l'annex A es pot consultar la formalització traduida a AMPL per a poder ser resolt amb un *solver* de programació lineal entera.
- ✓ Realitzar un estudi de la complexitat algorísmica del problema per a decidir quin mètode és més adient utilitzar per a resoldre'l.
    - A la secció 2.2.3 s'ha vist que utilitzar mètodes exactes per a resoldre el problema requereix d'un *solver* de pagament i que la complexitat computacional creix exponencialment amb aquests procediments. Tot seguit, a la secció 2.3, s'ha realitzat un estat de l'art per a veure com es resolen aquests problemes mitjançant mètodes aproximats i s'ha conclòs en optar per una metaheurística basada en una Cerca Tabú, donada la amplia literatura que hi havia al respecte i els bons resultats reportats.

✓ Desenvolupar un algorisme que resolgui el problema.

- També s'ha assolit el desenvolupament de l'algorisme tal i com s'ha detallat a la secció 3.2. Per a comprovar-ho, s'han pogut llençar execucions de diferents configuracions d'escenaris i s'han obtingut solucions que compleixen la factibilitat del problema.

✓ Validar i avaluar els resultats de l'algorisme.

- A la secció 4.2 s'ha realitzat una comparació entre els resultats de la metaheurística que s'ha implementat i els resultats òptims calculats per un *solver* de programació lineal entera. Els resultats obtinguts amb l'algorisme no s'allunyen del que l'òptim (taula 4.2) i els temps d'execució són molt millors que els del *solver* (figura 4.3)

✓ Realitzar una proposta d'integració de l'algorisme al sistema complet.

- Al capítol 5 s'ha vist que, pel fet de desenvolupar l'algorisme en al mateix llenguatge de programació que el del sistema, la integració és molt més directa i senzilla. Encapsulant el mòdul en un *package* de *Python*, i aplicant un patró de disseny adaptador, aquest ja pot ser utilitzat directament pel sistema. També s'ha realitzat una proposta d'integració per a utilitzar l'algorisme com a servei amb propòsits general.

En resum, s'ha vist que tots els objectius definits al capítol 1.4 proposats s'han assolit correctament.

8.2 Contribucions

D'una banda, l'algorisme implementat servirà per a assignar i generar rutes per un servei real de transport a la demanda com ho és el Servei Públic de Transport Especial. El propòsit d'aquest mòdul, a més, és aprofitar millor els recursos destinats a aquest servei, amb la qual cosa contribueix en millorar l'àmbit econòmic d'un servei públic destinat a un determinat col · lectiu.

D'altra banda, s'ha implementat un mòdul desacoblat del sistema, que es pot reaprofitar i adaptar fàcilment per a resoldre instàncies generals del problema amb una configuració flexible de paràmetres. Veient que en els darrers anys l'interès en nous sistemes de transport a la demanda que fomentin la compartició de vehicle (*carsharing*) ha anat a l'alça, així com també la importància en optimitzar la logística de transport d'alguns serveis, el mòdul dissenyat podria servir per a trobar solucions per a aquests altres àmbits de la mobilitat.

8.3 Treball futur

Tot i que els objectius s'hagin assolit correctament, el projecte presenta alguns punts a partir dels quals es pot estendre o millorar el mòdul desenvolupat.

Un dels punts de treball futur és millorar la metaheurística provant altres mètodes o pensar en maneres de millorar el temps de convergència del mètode. Una idea proposada que ha quedat fora de la implementació seria provar algun operador d'algorisme genètic que creués dues solucions veïnes (intercanviar les rutes) i comprovar el rendiment d'aquest nou operador

Un altre punt interessant, es afegir un procés de *clusterització* un cop obtingudes les rutes. Als usuaris se'ls pot exigir que el punt de recollida o lliurament sigui a una distància de com a molt uns 200 metres d'on han sol · licitat. Dit això, es podria detectar a posteriori si hi ha nodes propers que es puguin agrupar per facilitar el trajecte del vehicle i reduir el temps de viatge. Un exemple d'això es mostra a la figura 8.1.

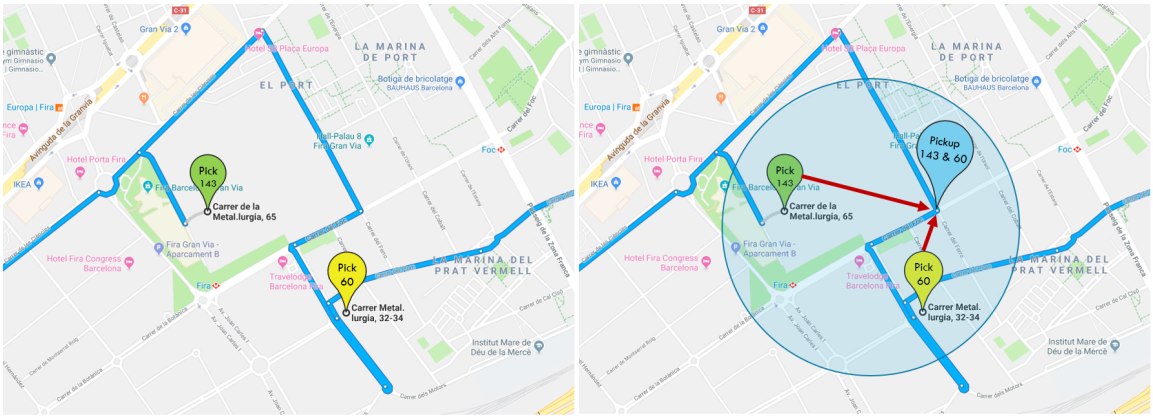

Figura 8.1: Exemple de dos punts de pickup que es poden ajuntar en un de sol per facilitar la circulació del vehicle

Aquest post-procés es pot realitzar fàcilment a simple vista, però seria interessant facilitar la feina als gestors de servei. Tot i així, no és un procés trivial, doncs per a fer aquests clústers s'ha de saber si l'usuari pot moure's fins al punt calculat i saber on situar aquest punt intermedi (s'ha de tenir en compte el sentit dels carrers i que el nou punt de recollida no perjudiqui el recorregut del vehicle). De totes maneres, és millor que la decisió de formar aquests clústers entre nodes quedi a càrrec del conductor del vehicle i dels gestors del servei, ja que disposen de la informació de la xarxa de transport i del grau de mobilitat dels usuaris.

Les execucions amb les que s'ha provat l'algorisme són escenaris on les peticions estan distribuïdes uniformement en l'espai i en l'horitzó temporal, i no seguien una distribució realista ni patrons de mobilitat. Seria interessant obtenir les peticions reals del servei actual per provar l'algorisme i contrastar com es millora l'ús de recursos respecte l'assignació actual. Malauradament, no s'ha pogut obtenir aquesta informació durant el desenvolupament d'aquest treball, ja que la informació ja venia pre-processada amb agrupacions d'alguns dels nodes.

El darrer punt, i un dels més importants, és que, per a posar a prova l'algorisme, s'ha utilitzat matrius de distància i temps de viatge sobre la xarxa de transport proporcionades per la API de *Google Maps*. Aquesta API és de pagament si se supera un límit mensual de prova, i un alt volum de peticions pot acabar sortint

car (1000 elements = 5$). Una alternativa clara seria utilitzar *OpenStreetMap*, que serviria d'igual manera per a obtenir distància de viatge, però el temps de viatge el proporciona en *free-flow* (sense considerar trànsit). És important tenir una bona estimació del temps de viatge (o almenys una sobreestimació no exagerada) ja que a partir d'aquesta es calculen les finestres temporals dels nodes de *delivery*. Si s'utilitza *OpenStreetMap*, s'hauria de multiplicar temps de viatge en *free-flow* per un valor que ens donés una estimació de temps realista. Es podria generar un model que calculés el temps de viatge segons l'hora del dia analitzant patrons de mobilitat amb dades històriques, però això ja seria treball per a realitzar en un altre projecte.

8.4 Conclusions personals

Primer de tot, el fet de poder participar en un projecte real treballant dins d'un equip de desenvolupament amb metodologies àgils i amb experts sobre la temàtica, ha sigut una experiència molt satisfactòria. Més encara, si l'objectiu del projecte és millorar un servei públic de la teva pròpia ciutat del que en depèn un col · lectiu de la societat.

Per altra part, el procés d'endinsar-se en un problema matemàtic complex, estudiar les seves variants i analitzar on es troben els límits de la seva tractabilitat per a resoldre'l, ha sigut una experiència molt enriquidora acadèmicament. Tot aquest coneixement adquirit em serà de molta utilitat en un futur ja que actualment l'interès en resoldre problemes d'optimització similars al `VRP` està incrementant per tal de minimitzar costos en planificació i logística.

Seguint aquesta línia, he pogut treballar amb problemes del camp de la Investigació Operativa, utilitzant models matemàtics d'optimització amb els que no havia pogut aprofundir durant el grau i que ara considero que és un coneixement crucial en la formació d'un enginyer en informàtica.

Bibliografia

[1] Ajuntament de Barcelona. Servei públic de transport especial. http://ajuntament.barcelona.cat/accessible/ca/impd/transport-servei-public-transport-especial, 2018. [Online; últim accés 24-Setembre-2018].

[2] Ajuntament de Barcelona. Institut municipal de persones amb discapacitat. http://ajuntament.barcelona.cat/accessible/ca/impd/institut-municipal-persones-amb-discapacitat, 2018. [Online; últim accés 24-Setembre-2018].

[3] Wikipedia. Scrum (software development). https://en.wikipedia.org/wiki/Scrum_(software_development), 2018. [Online; últim accés 24-Setembre-2018].

[4] Kent Beck. *Test Driven Development: By Example*. Addison-Wesley, 2002.

[5] Paolo Toth and Daniele Vigo. *The Vehicle Routing Problem (Monographs on Discrete Mathematics and Applications)*. SIAM, 2002.

[6] Christos H. Papadimitriou. *Computational Complexity*. Pearson, 1994.

[7] G. Clarke and J. Wright. Scheduling of vehicles from a central depot to a number of delivery points. Technical report, Operations Research, 1964.

[8] M. Solomon. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. 1993.

[9] J. Potvin and J. Rousseau. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. Technical report, European Journal of Operational Research, 1993.

[10] I. Osman. Algorithms for the vehicle routing ans scheduling problem with time window constraints. Technical report, Operations Research, 1987.

[11] Stuart Russell and Peter Norvig. *Artificial Intelligence - A Modern Approach*. Prentice Hall, 2003.

[12] Imen Harbaoui Dridi, Ryan Kammarti, Mekki Ksouri, and Pierre Borne. A genetic algorithm for the multi-pickup and delivery problem with time windows. *CoRR*, abs/1009.5031, 2010.

[13] F. Glover and M. Laguna. *Tabu Search*. Kluwer, 1997.

[14] P. Toth and D. Vigo. *Fast local search algorithms for the handicapped persons transportation problem. (In book "Metaheuristics: Theory and Applications").* Kluwer Academic Publishers, 1996.

[15] Paolo Toth and Daniele Vigo. Heuristic algorithms for the handicapped persons transportation problem. *Transportation Science*, 31:60–71, 1997.

[16] Haibing Li and Andrew Lim. A metaheuristic for the pickup and delivery problem with time windows. *International Journal on Artificial Intelligence Tools*, 2001.

[17] Ronald L. Rivest Thomas H. Cormen, Charles E. Leiserson and Clifford Stein. *Introduction to Algorithms, Third Edition.* The MIT Press, 2009.

[18] Jon Kleinberg and Eva Tardos. *Algorithm Design.* Pearson, 2005.

[19] Hammadi S. Borne P. Kammarti, R. and M. Ksouri. Improved tabu search in an hybrid evolutionary approach for the pickup and delivery problem with time windows. *[IEEE 2005 IEEE Intelligent Transportation Systems, 2005. - Vienna, Austria (13-16 Sept. 2005)] Proceedings. 2005 IEEE Intelligent Transportation Systems, 2005.*, 2005.

[20] Wan-Rong Jih, Jane Yung, and Jane Hsu. A family competition genetic algorithm for the pickup and delivery problems with time window. 12 2018.

[21] R M Jorgensen, Jesper Larsen, and K B Bergvinsdottir. Solving the dial-a-ride problem using genetic algorithms. *Journal of the Operational Research Society*, 58, 10 2007.

[22] William P Nanry and J Wesley Barnes. Solving the pickup and delivery problem with time windows using reactive tabu search. *Transportation Research Part B: Methodological*, 34, 2000.

[23] Hoong Chuin Lau and Zhe Liang. Pickup and delivery problem with time windows: Algorithms and test case generation. *International Journal on Artificial Intelligence*, 11, 2002.

[24] Erich Gamma, Richard Helm, Ralph Johnson, and John M. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley Professional, 1 edition, 1994.

Annex

Annex A

Model d'optimització VRPPDTW en AMPL

```
# Model d'optimitzacio per a Heterogeneous VRPPDTW
#Numero de peticiones
param n;
#Numero de vehiculos
param K := n;

set DEPOT_O := {0};
set DEPOT_D := {(2*n)+1};
set DEPOTS := DEPOT_O union DEPOT_D;
set ALL_NODES := 0..2*n+1;
set PNODES := 1..n;
set PNODES_NA within PNODES;
set DNODES := n+1..2*n;
set DNODES_NA within DNODES;
set NODES := 1..2*n;
set NODES_NA within NODES;
set VEHICLES:=1..K;
set ADAPTED within VEHICLES;
set V := ALL_NODES;
set V_NA := NODES_NA union (DEPOT_O union DEPOT_D);

param load{ALL_NODES};
param cost{ALL_NODES,ALL_NODES}>=0, integer;
param ttime{ALL_NODES,ALL_NODES}>=0, integer;
param capacity{VEHICLES}>=0;
param stime{ALL_NODES};
param a{ALL_NODES};
param b{ALL_NODES};

var x {ALL_NODES,ALL_NODES,VEHICLES} binary;
```

```
var delta {ALL_NODES,ALL_NODES,VEHICLES} binary;
var T {ALL_NODES,VEHICLES}>=0, integer;
var L {ALL_NODES,VEHICLES}>=0, integer;

# Funcion objectivo
minimize total_cost:
(sum{k in ADAPTED, i in V, j in V}(cost[i,j]*x[i,j,k])) +
(sum{k in VEHICLES diff ADAPTED, i in V_NA, j in V_NA}
   (cost[i,j]*x[i,j,k]));

# Restricciones
# Initial
subject to NA_vehicles_cant_serve_A{k in VEHICLES diff ADAPTED, i in
   NODES diff NODES_NA}:
sum{j in ALL_NODES} x[i,j,k] = 0;
subject to vehicles_cant_go_to_a_DNODE_first{k in VEHICLES}: #, i in
   NODES}:
sum{j in DNODES} x[0,j,k] = 0;
subject to set_diagonal_to_infinite{k in VEHICLES, i in V}: #, i in
   NODES}:
x[i,i,k] = 0;

# 9.2 All served once
subject to all_served_once_A{i in NODES}:
(sum{k in ADAPTED, j in NODES union DEPOT_D} x[i,j,k]) + (sum{k in
   VEHICLES diff ADAPTED, j in NODES_NA union DEPOT_D} x[i,j,k])=1;

# 9.3 Same vehicle
subject to served_by_same_vehicle_A{k in ADAPTED, i in PNODES}:
(sum{j in NODES} x[i,j,k] - sum{j in NODES} x[j,n+i,k]) = 0;
subject to served_by_same_vehicle_NA{k in VEHICLES diff ADAPTED, i
   in PNODES_NA}:
(sum{j in NODES_NA} x[i,j,k] - sum{j in NODES_NA} x[j,n+i,k]) = 0;

# 9.4 Depot o(k)
subject to starts_from_origin_A{k in ADAPTED}:
sum{j in (PNODES union DEPOT_D)} x[0,j,k] = 1;
subject to starts_from_origin_NA{k in VEHICLES diff ADAPTED}:
sum{j in PNODES_NA union DEPOT_D} x[0,j,k] = 1;

# 9.5 Flow conservation
subject to flow_conservation_A{k in ADAPTED, j in NODES}:
(sum{i in NODES union DEPOT_O}x[i,j,k]) - (sum{i in NODES union
   DEPOT_D}x[j,i,k]) = 0;
subject to flow_conservation_NA{k in VEHICLES diff ADAPTED, j in
   NODES_NA}:
```

```
(sum{i in NODES_NA union DEPOT_O}x[i,j,k]) - (sum{i in NODES_NA
   union DEPOT_D}x[j,i,k]) = 0;

# 9.6 Depot d(k)
subject to finish_at_destin_A{k in ADAPTED}:
sum{i in DNODES union DEPOT_O}x[i,(2*n)+1,k] = 1;
subject to finish_at_destin_NA{k in VEHICLES diff ADAPTED}:
sum{i in DNODES_NA union DEPOT_O}x[i,(2*n)+1,k] = 1;

# 9.7 TW
subject to tw1_A{k in ADAPTED, i in V, j in V}:
(T[i,k] + stime[i] + ttime[i,j] - T[j,k]) <= (1 - x[i,j,k]) *
   max(b[i]+stime[i]+ttime[i,j]-a[j], 0);
subject to tw1_NA{k in VEHICLES diff ADAPTED, i in V_NA, j in V_NA}:
(T[i,k] + stime[i] + ttime[i,j] - T[j,k]) <= (1 - x[i,j,k]) *
   max(b[i]+stime[i]+ttime[i,j]-a[j], 0);

# 9.8
subject to tw2_A{k in ADAPTED, i in V}:
a[i] <= T[i,k] <= b[i];
subject to tw2_NA{k in VEHICLES diff ADAPTED, i in V_NA}:
a[i] <= T[i,k] <= b[i];

# 9.9
subject to tw3_A{k in ADAPTED, i in PNODES}:
T[i,k] + ttime[i,n+i] <= T[n+i,k];
subject to tw3_NA{k in VEHICLES diff ADAPTED, i in PNODES_NA}:
T[i,k] + ttime[i,n+i] <= T[n+i,k];

# 9.10
subject to linearization1_A{k in ADAPTED, i in V, j in V}:
(L[i,k] + load[j] - L[j,k]) + 10000000*delta[i,j,k]>= 0;
subject to linearization1_A_2{k in ADAPTED, i in V, j in V}:
(L[i,k] + load[j] - L[j,k]) - 10000000*delta[i,j,k]<= 0;
subject to linearization2_A{k in ADAPTED, i in V, j in V}:
x[i,j,k] + delta[i,j,k] <= 1;

subject to linearization1_NA{k in VEHICLES diff ADAPTED, i in V_NA,
   j in V_NA}:
(L[i,k] + load[j] - L[j,k]) + 10000000*delta[i,j,k]>= 0;
subject to linearization1_NA_2{k in VEHICLES diff ADAPTED, i in
   V_NA, j in V_NA}:
(L[i,k] + load[j] - L[j,k]) - 10000000*delta[i,j,k] <= 0;
subject to linearization2_NA{k in VEHICLES diff ADAPTED, i in V_NA,
   j in V_NA}:
x[i,j,k] + delta[i,j,k] <= 1;
```

```
# 9.11
subject to capacity1_A{k in ADAPTED, i in PNODES}:
load[i] <= L[i,k] <=capacity[k];
subject to capacity1_NA{k in VEHICLES diff ADAPTED, i in PNODES_NA}:
load[i] <= L[i,k] <=capacity[k];

# 9.12 Repassar amb maripaz
subject to capacity2_A{k in ADAPTED, i in DNODES}:
0 <= L[i,k] <= capacity[k] + load[i-n];
subject to capacity2_NA{k in VEHICLES diff ADAPTED, i in DNODES_NA}:
0 <= L[i,k] <= capacity[k] + load[i-n];

# 9.13 Initial Load
subject to initial_load{k in VEHICLES}:
L[0,k] = 0;
subject to initial_load_D{k in VEHICLES}:
L[(2*n)+1,k] = 0;

# 9.14
#subject to initial_load{i in ALL_NODES, j in ALL_NODES, k in
   VEHICLES}:
#x[i,j,k] >= 0;
```

Annex B

Output instància 100-1H-2ST-10TW

A continuació es mostren les solucions obtingudes a cada fase de la metaheurística per a la instancia `100-1H-2ST-10TW`. Les rutes es representen com una seqüència de nodes que ha de visitar cada vehicles ('A' és vehicle adaptat i 'NA' vehicle no adaptat). El node 0 fa referència al depot.

Solució heurística inicial

Es comença amb una solució que requereix 56 vehicles obtinguda amb la heurística de construcció inicial.

```
A  [0, 48, 75, 148, 175, 18, 118, 0]
A  [0, 60, 82, 160, 182, 67, 167, 0]
NA [0, 53, 153, 99, 199, 11, 111, 0]
A  [0, 43, 20, 143, 120, 0]
NA [0, 71, 171, 4, 104, 0]
A  [0, 76, 176, 41, 141, 0]
A  [0, 87, 187, 84, 184, 0]
NA [0, 77, 177, 58, 158, 0]
A  [0, 38, 16, 138, 116, 0]
A  [0, 31, 131, 59, 159, 0]
A  [0, 9, 109, 44, 144, 0]
NA [0, 36, 136, 62, 162, 0]
NA [0, 13, 5, 113, 105, 89, 189, 0]
A  [0, 50, 47, 150, 147, 0]
A  [0, 49, 149, 81, 181, 93, 193, 0]
NA [0, 24, 85, 124, 185, 2, 102, 0]
A  [0, 35, 135, 21, 121, 0]
A  [0, 25, 46, 125, 146, 0]
NA [0, 64, 78, 164, 178, 92, 192, 0]
A  [0, 68, 74, 168, 19, 174, 119, 0]
A  [0, 26, 126, 29, 129, 0]
```

```
A  [0, 15, 115, 80, 180, 0]
NA [0, 51, 10, 151, 110, 0]
NA [0, 23, 123, 54, 154, 0]
NA [0, 61, 161, 73, 173, 0]
NA [0, 97, 197, 7, 107, 0]
A  [0, 1, 101, 83, 183, 0]
NA [0, 22, 122, 0]
A  [0, 32, 132, 0]
NA [0, 6, 106, 37, 137, 0]
NA [0, 27, 127, 0]
NA [0, 42, 86, 142, 186, 0]
NA [0, 45, 69, 169, 145, 0]
A  [0, 8, 108, 0]
A  [0, 33, 133, 0]
NA [0, 100, 200, 0]
A  [0, 12, 112, 0]
A  [0, 55, 155, 95, 195, 0]
NA [0, 57, 91, 157, 191, 0]
A  [0, 56, 72, 156, 172, 0]
A  [0, 66, 166, 0]
A  [0, 70, 170, 0]
A  [0, 65, 17, 165, 117, 0]
NA [0, 94, 194, 0]
A  [0, 39, 139, 0]
A  [0, 79, 52, 179, 152, 0]
NA [0, 90, 190, 0]
A  [0, 98, 198, 0]
NA [0, 14, 114, 0]
A  [0, 30, 88, 130, 188, 0]
A  [0, 3, 103, 0]
NA [0, 28, 128, 0]
A  [0, 40, 140, 0]
A  [0, 63, 163, 0]
A  [0, 96, 196, 0]
A  [0, 34, 134, 0]
Real Cost: 1130470.0
Total Cost: 23530470.0
56 vehicles
initial\_solution {} 0.046874284744262695 sec
No improvement for pre-heuristic
```

Solució després de la fase de millora

Veiem que s'ha pogut alliberar 5 vehicles a la fase de millora i reduix el cost de recorregut dels taxis.

```
ASSIGNATION RESULT:
A  [0, 0]
A  [0, 0]
A  [0, 0]
NA [0, 0]
NA [0, 0]
NA [0, 22, 122, 0]
NA [0, 14, 114, 0]
A  [0, 33, 133, 0]
NA [0, 94, 194, 0]
NA [0, 61, 161, 0]
NA [0, 7, 107, 0]
NA [0, 6, 106, 0]
A  [0, 8, 108, 0]
NA [0, 27, 127, 0]
A  [0, 66, 166, 0]
A  [0, 20, 120, 0]
A  [0, 98, 198, 0]
A  [0, 13, 113, 3, 103, 0]
A  [0, 55, 89, 155, 189, 0]
A  [0, 5, 105, 59, 159, 0]
NA [0, 91, 191, 11, 111, 0]
NA [0, 100, 200, 67, 167, 0]
A  [0, 57, 21, 157, 121, 0]
NA [0, 90, 95, 190, 195, 0]
NA [0, 28, 128, 83, 183, 0]
A  [0, 1, 62, 101, 162, 0]
A  [0, 97, 46, 197, 146, 0]
NA [0, 4, 73, 104, 173, 0]
A  [0, 12, 29, 112, 129, 0]
A  [0, 26, 39, 126, 139, 0]
A  [0, 25, 44, 125, 144, 0]
A  [0, 35, 135, 63, 163, 0]
A  [0, 36, 136, 96, 196, 0]
A  [0, 76, 176, 41, 141, 2, 102, 0]
A  [0, 87, 187, 84, 184, 0]
NA [0, 77, 177, 58, 158, 0]
A  [0, 38, 16, 138, 116, 0]
A  [0, 50, 47, 150, 147, 0]
A  [0, 15, 115, 80, 180, 0]
NA [0, 51, 10, 151, 110, 0]
NA [0, 23, 123, 54, 154, 0]
```

```
NA [0, 42, 86, 142, 186, 0]
A  [0, 56, 72, 156, 172, 0]
A  [0, 79, 52, 179, 152, 0]
A  [0, 60, 82, 160, 182, 0]
NA [0, 53, 153, 99, 199, 0]
NA [0, 24, 85, 124, 185, 0]
NA [0, 45, 69, 169, 145, 37, 137, 0]
A  [0, 9, 109, 65, 17, 165, 117, 0]
A  [0, 31, 131, 30, 88, 130, 188, 0]
A  [0, 71, 171, 70, 34, 170, 134, 0]
A  [0, 43, 143, 32, 40, 132, 140, 0]
A  [0, 48, 75, 148, 175, 18, 118, 0]
A  [0, 49, 149, 81, 181, 93, 193, 0]
NA [0, 64, 78, 164, 178, 92, 192, 0]
A  [0, 68, 74, 168, 19, 174, 119, 0]
Real Cost: 1043602.0
Total Cost: 21443602.0
Feasible result
Number of iterations: 38
Used 51 vehicles for 100 requests
Stopped due to no improvement, local minimum reached
```

Assignació resultant

```
SCHEDULE
-------------------------------------------------
VEHICLE # 1
https://www.google.es/maps/dir/41.37240192771526,2.1564798785901393/41.450726316296155,2.
    250007299151488/
(node 22) -> Pickup user 22 at 8:18
(node 122) -> Delivery user 22 at 8:48
-------------------------------------------------
VEHICLE # 2
https://www.google.es/maps/dir/41.38780519456423,2.1053433313478394/41.393302010237484,2.
    2029086725241056/
(node 14) -> Pickup user 14 at 8:34
(node 114) -> Delivery user 14 at 9:02
-------------------------------------------------
VEHICLE # 3
https://www.google.es/maps/dir/41.420051100668154,2.2227462096158037/41.40306030009588,2.
    1569763500896775/
(node 33) -> Pickup user 33 at 8:21
(node 133) -> Delivery user 33 at 8:50
-------------------------------------------------
VEHICLE # 4
https://www.google.es/maps/dir/41.41744357446403,2.1794699152876897/41.42554129030501,2.
    2230583503503887/
(node 94) -> Pickup user 94 at 8:26
```

(node 194) -> Delivery user 94 at 8:50

VEHICLE # *5*

https://www.google.es/maps/dir/41.34424355776647,2.1058336430526485/41.417355908696415,2.171267756346324/

(node 61) -> Pickup user 61 at 8:14

(node 161) -> Delivery user 61 at 8:45

VEHICLE # *6*

https://www.google.es/maps/dir/41.44429731259926,2.2292081183672954/41.4037499922858,2.215352856909565/

(node 7) -> Pickup user 7 at 8:37

(node 107) -> Delivery user 7 at 8:54

VEHICLE # *7*

https://www.google.es/maps/dir/41.387413992045445,2.153073850255448/41.36525430829251,2.1549432246225004/

(node 6) -> Pickup user 6 at 8:19

(node 106) -> Delivery user 6 at 8:36

VEHICLE # *8*

https://www.google.es/maps/dir/41.34297266703129,2.1132786934895904/41.38056627088029,2.174873125504756/

(node 8) -> Pickup user 8 at 8:21

(node 108) -> Delivery user 8 at 8:45

VEHICLE # *9*

https://www.google.es/maps/dir/41.3766352006628,2.096458797295751/41.39050511407172,2.160669472609889/

(node 27) -> Pickup user 27 at 8:20

(node 127) -> Delivery user 27 at 8:43

VEHICLE # *10*

https://www.google.es/maps/dir/41.38416593288398,2.1707062375861903/41.39523308348684,2.17698498224184/

(node 66) -> Pickup user 66 at 8:25

(node 166) -> Delivery user 66 at 8:39

VEHICLE # *11*

https://www.google.es/maps/dir/41.39755042250464,2.135818872617668/41.4044449640226,2.1991018679081797/

(node 20) -> Pickup user 20 at 8:13

(node 120) -> Delivery user 20 at 8:40

VEHICLE # *12*

https://www.google.es/maps/dir/41.41172617091007,2.132841563712378/41.417907391104855,2.1929084362746525/

(node 98) -> Pickup user 98 at 8:28

(node 198) -> Delivery user 98 at 8:51

VEHICLE # *13*

https://www.google.es/maps/dir/41.393806047642954,2.15352307030391/41.399265053772524,2.1577066569162415/41.41034989298338,2.170450111889562/41.40567206947544,2.187370052243322/

(**node** 13) -> Pickup user 13 at 8:02

(**node** 113) -> Delivery user 13 at 8:13

(**node** 3) -> Pickup user 3 at 8:35

(**node** 103) -> Delivery user 3 at 8:54

VEHICLE # *14*

https://www.google.es/maps/dir/41.4088381163352,2.185697795895723/41.406129469848196,2.1827150419301384/41.397667112668906,2.162008235929492/41.42692904598016,2.15436996941117/

(**node** 55) -> Pickup user 55 at 8:23

(**node** 89) -> Pickup user 89 at 8:37

(**node** 155) -> Delivery user 55 at 8:46

(**node** 189) -> Delivery user 89 at 9:03

VEHICLE # *15*

https://www.google.es/maps/dir/41.39833697571543,2.1488458637525527/41.407981739405024,2.1504032971282463/41.408437022719184,2.1548674056137567/41.379138965843836,2.178133891343553/

(**node** 5) -> Pickup user 5 at 8:13

(**node** 105) -> Delivery user 5 at 8:26

(**node** 59) -> Pickup user 59 at 8:31

(**node** 159) -> Delivery user 59 at 8:58

VEHICLE # *16*

https://www.google.es/maps/dir/41.38955364814662,2.1482155529705045/41.377023261491445,2.157164333748227/41.3788783619956,2.153945962373683/41.37401536965582,2.1165293279572124/

(**node** 91) -> Pickup user 91 at 8:32

(**node** 191) -> Delivery user 91 at 8:46

(**node** 11) -> Pickup user 11 at 8:54

(**node** 111) -> Delivery user 11 at 9:14

VEHICLE # *17*

https://www.google.es/maps/dir/41.40385705042207,2.1457271602347734/41.40385705042207,2.1457271602347734/41.39505786824027,2.115786802131641/41.456968325842524,2.2138406587884623/

(**node** 100) -> Pickup user 100 at 8:21

(**node** 200) -> Delivery user 100 at 8:40

(**node** 67) -> Pickup user 67 at 8:56

(**node** 167) -> Delivery user 67 at 9:15

VEHICLE # *18*

https://www.google.es/maps/dir/41.40290368241243,2.1241347213329114/41.39695321093745,2.142553012832942/41.38763894119782,2.1561333453929654/41.384646287507714,2.0949491487054983/

(**node** 57) -> Pickup user 57 at 8:23

(**node** 21) -> Pickup user 21 at 8:34

(**node** 157) -> Delivery user 57 at 8:45

(**node** 121) -> Delivery user 21 at 9:00

VEHICLE # *19*

https://www.google.es/maps/dir/41.37139691677315,2.1668818256513145/41.39027848591838,2.1216784194504488/41.385880519410684,2.10994150663752/41.440015812803274,2.1743902691367802/

(**node** 90) -> Pickup user 90 at 8:28
(**node** 95) -> Pickup user 95 at 8:48
(**node** 190) -> Delivery user 90 at 8:54
(**node** 195) -> Delivery user 95 at 9:12

VEHICLE # *20*

https://www.google.es/maps/dir/41.41111639066616,2.1219368268912757/41.375733343048616,2.100376366997139/41.37451762858559,2.121587197523643/41.36396264451981,2.1538592092652142/
(**node** 28) -> Pickup user 28 at 8:35
(**node** 128) -> Delivery user 28 at 8:49
(**node** 83) -> Pickup user 83 at 8:59
(**node** 183) -> Delivery user 83 at 9:22

VEHICLE # *21*

https://www.google.es/maps/dir/41.372386460535736,2.1677772474966575/41.38953795691845,2.1927491999951276/41.39251049201075,2.192396112316732/41.42036063026296,2.1499980214590435/
(**node** 1) -> Pickup user 1 at 8:16
(**node** 62) -> Pickup user 62 at 8:36
(**node** 101) -> Delivery user 1 at 8:42
(**node** 162) -> Delivery user 62 at 9:07

VEHICLE # *22*

https://www.google.es/maps/dir/41.43209805549199,2.1895905066445343/41.43943730539795,2.1992816036401672/41.43675478347506,2.2194381568882164/41.381098419960075,2.1809349639776316/
(**node** 97) -> Pickup user 97 at 8:14
(**node** 46) -> Pickup user 46 at 8:29
(**node** 197) -> Delivery user 97 at 8:42
(**node** 146) -> Delivery user 46 at 9:02

VEHICLE # *23*

https://www.google.es/maps/dir/41.35247217770629,2.113010301995855/41.42924308127392,2.1700634586882277/41.43291787526791,2.165341717384142/41.39705135679765,2.161110887443284/
(**node** 4) -> Pickup user 4 at 8:26
(**node** 73) -> Pickup user 73 at 8:53
(**node** 104) -> Delivery user 4 at 9:01
(**node** 173) -> Delivery user 73 at 9:24

VEHICLE # *24*

https://www.google.es/maps/dir/41.44663629204065,2.1876711303187837/41.40110603544174,2.139678276376054/41.40449956963676,2.1380905265998424/41.40893469748005,2.1344856197110054/
(**node** 12) -> Pickup user 12 at 8:23
(**node** 29) -> Pickup user 29 at 8:42
(**node** 112) -> Delivery user 12 at 8:53
(**node** 129) -> Delivery user 29 at 8:58

VEHICLE # *25*

https://www.google.es/maps/dir/41.41054669189377,2.199304938835019/41.42992148929914,2.1425980089028553/41.396854270088035,2.1430330354768765/41.37393693570697,2.1446488622710747/
(**node** 26) -> Pickup user 26 at 8:11
(**node** 39) -> Pickup user 39 at 8:27

(node 126) -> Delivery user 26 at 8:45
(node 139) -> Delivery user 39 at 9:03

VEHICLE # *26*

https://www.google.es/maps/dir/41.36491680928197,2.1020193534542906/41.37843774021407,2.1127846621751245/41.39690621973505,2.207512949159215/41.438941759311035,2.2365751061831385/
(node 25) -> Pickup user 25 at 8:06
(node 44) -> Pickup user 44 at 8:17
(node 125) -> Delivery user 25 at 8:41
(node 144) -> Delivery user 44 at 8:53

VEHICLE # *27*

https://www.google.es/maps/dir/41.42737363417305,2.140870008919561/41.36422474712989,2.0948881142877225/41.35888262881336,2.099928012079781/41.418754318359504,2.195471709708778/
(node 35) -> Pickup user 35 at 8:05
(node 135) -> Delivery user 35 at 8:23
(node 63) -> Pickup user 63 at 8:35
(node 163) -> Delivery user 63 at 9:08

VEHICLE # *28*

https://www.google.es/maps/dir/41.41452557246937,2.128593784360822/41.38113429215801,2.17589916852294/41.3799030302449,2.176565243756661/41.356677987927746,2.131483245554454/
(node 36) -> Pickup user 36 at 8:01
(node 136) -> Delivery user 36 at 8:28
(node 96) -> Pickup user 96 at 8:41
(node 196) -> Delivery user 96 at 9:09

VEHICLE # *29*

https://www.google.es/maps/dir/41.40400480460379,2.2117863573879086/41.368363523803325,2.130037179237436/41.35472607493583,2.1383761433436295/41.427074098469284,2.2073312909811795/41.426625011488504,2.2110626220201386/41.38071916718298,2.1237432106296166/
(node 76) -> Pickup user 76 at 7:57
(node 176) -> Delivery user 76 at 8:21
(node 41) -> Pickup user 41 at 8:33
(node 141) -> Delivery user 41 at 8:55
(node 2) -> Pickup user 2 at 8:59
(node 102) -> Delivery user 2 at 9:25

VEHICLE # *30*

https://www.google.es/maps/dir/41.36531594638897,2.103435175727658/41.45017056302025,2.2174706741898356/41.4488020334902,2.2274124558479205/41.42714025512682,2.153176241978267/
(node 87) -> Pickup user 87 at 7:57
(node 187) -> Delivery user 87 at 8:27
(node 84) -> Pickup user 84 at 8:40
(node 184) -> Delivery user 84 at 9:01

VEHICLE # *31*

https://www.google.es/maps/dir/41.458477294854596,2.2438136418904597/41.38298985532876,2.13064912037335/41.39781204432657,2.1405599824298145/41.444469360288416,2.186513302071651/
(node 77) -> Pickup user 77 at 7:58

(**node** 177) -> Delivery user 77 at 8:31
(**node** 58) -> Pickup user 58 at 8:39
(**node** 158) -> Delivery user 58 at 8:55

VEHICLE # *32*

https://www.google.es/maps/dir/41.35323798027913,2.1125429235179256/41.43431500191171,2.2293302234698276/41.43884576126499,2.219623101710822/41.360042491393884,2.143967774864733/
(**node** 38) -> Pickup user 38 at 7:59
(**node** 16) -> Pickup user 16 at 8:25
(**node** 138) -> Delivery user 38 at 8:34
(**node** 116) -> Delivery user 16 at 8:56

VEHICLE # *33*

https://www.google.es/maps/dir/41.35026153274539,2.1051894215901257/41.432986435620215,2.2260011703601004/41.40680282907835,2.1882902463413956/41.3979796864766,2.170868576891352/
(**node** 50) -> Pickup user 50 at 8:03
(**node** 47) -> Pickup user 47 at 8:30
(**node** 150) -> Delivery user 50 at 8:43
(**node** 147) -> Delivery user 47 at 8:55

VEHICLE # *34*

https://www.google.es/maps/dir/41.38789507613582,2.18769810218779/41.39831359726639,2.1990040543817306/41.40560942433947,2.200974591297531/41.38379714377036,2.1510407971509875/
(**node** 15) -> Pickup user 15 at 8:12
(**node** 115) -> Delivery user 15 at 8:29
(**node** 80) -> Pickup user 80 at 8:36
(**node** 180) -> Delivery user 80 at 8:59

VEHICLE # *35*

https://www.google.es/maps/dir/41.452359652385034,2.2318020180706175/41.36142902639562,2.130895754074072/41.34649548385862,2.11995319652673/41.42383620697996,2.14152828058758/
(**node** 51) -> Pickup user 51 at 8:12
(**node** 10) -> Pickup user 10 at 8:39
(**node** 151) -> Delivery user 51 at 8:46
(**node** 110) -> Delivery user 10 at 9:06

VEHICLE # *36*

https://www.google.es/maps/dir/41.42503287824611,2.1716954225202123/41.42665994551101,2.2000645728015584/41.44346167418366,2.217481818841234/41.43632146740857,2.183125547345098/
(**node** 23) -> Pickup user 23 at 8:13
(**node** 123) -> Delivery user 23 at 8:38
(**node** 54) -> Pickup user 54 at 8:48
(**node** 154) -> Delivery user 54 at 9:08

VEHICLE # *37*

https://www.google.es/maps/dir/41.38676220716886,2.1131608612891917/41.42593610158074,2.209663550900323/41.42039681199551,2.1914582286160913/41.42964417809163,2.1647955639608574/
(**node** 42) -> Pickup user 42 at 8:20
(**node** 86) -> Pickup user 86 at 8:48
(**node** 142) -> Delivery user 42 at 8:58

(**node** 186) -> Delivery user 86 at 9:15

VEHICLE # *38*

https://www.google.es/maps/dir/41.39076498689057,2.1231631004027585/41.440763947475226,2.2233910403491306/41.45413014888029,2.2322343365166755/41.359760797237556,2.138901560105724/

(**node** 56) -> Pickup user 56 at 8:24

(**node** 72) -> Pickup user 72 at 8:49

(**node** 156) -> Delivery user 56 at 8:59

(**node** 172) -> Delivery user 72 at 9:21

VEHICLE # *39*

https://www.google.es/maps/dir/41.38029969697425,2.1288684958342423/41.3866754985233,2.1114850208152838/41.420556389476616,2.2230853503428767/41.44713974607092,2.215403138319911/

(**node** 79) -> Pickup user 79 at 8:27

(**node** 52) -> Pickup user 52 at 8:42

(**node** 179) -> Delivery user 79 at 9:00

(**node** 152) -> Delivery user 52 at 9:10

VEHICLE # *40*

https://www.google.es/maps/dir/41.450150896214076,2.238917399126563/41.4552520910444,2.1983004725341586/41.43407858584143,2.200463367551738/41.37057717301755,2.094904643014082/

(**node** 60) -> Pickup user 60 at 7:55

(**node** 82) -> Pickup user 82 at 8:12

(**node** 160) -> Delivery user 60 at 8:23

(**node** 182) -> Delivery user 82 at 8:45

VEHICLE # *41*

https://www.google.es/maps/dir/41.44519216511467,2.180668710381282/41.37460942404317,2.1211345922785023/41.383090483923645,2.133861204326072/41.39629323800302,2.1522350111321087/

(**node** 53) -> Pickup user 53 at 7:56

(**node** 153) -> Delivery user 53 at 8:19

(**node** 99) -> Pickup user 99 at 8:31

(**node** 199) -> Delivery user 99 at 8:45

VEHICLE # *42*

https://www.google.es/maps/dir/41.42102492158094,2.1543294031110083/41.427124689840525,2.1511752910255804/41.3921438574669,2.2014647531746485/41.40477024784978,2.212123761960778/

(**node** 24) -> Pickup user 24 at 8:05

(**node** 85) -> Pickup user 85 at 8:18

(**node** 124) -> Delivery user 24 at 8:36

(**node** 185) -> Delivery user 85 at 8:45

VEHICLE # *43*

https://www.google.es/maps/dir/41.348788738579564,2.1374660389923776/41.349183695614805,2.1193627871380 46/41.354401309227235,2.122619394861179/41.37267618319784,2.098537962199482/41.37854826484176,2.1287916066711716/41.39370968865407,2.1876434459767196/

(**node** 45) -> Pickup user 45 at 8:20

(**node** 69) -> Pickup user 69 at 8:26

(**node** 169) -> Delivery user 69 at 8:30

(**node** 145) -> Delivery user 45 at 8:47

(node 37) -> Pickup user 37 at 9:02
(node 137) -> Delivery user 37 at 9:27

VEHICLE # *44*

https://www.google.es/maps/dir/41.37845790717103,2.1063572531733166/41.37229630463679,2.0919588754533147/41.36705665887789,2.104060714791908/41.36394301138547,2.1052702014402063/41.36907243291208,2.169891238190284/41.372104617272015,2.1691091160374194/

(node 9) -> Pickup user 9 at 8:01
(node 109) -> Delivery user 9 at 8:12
(node 65) -> Pickup user 65 at 8:26
(node 17) -> Pickup user 17 at 8:35
(node 165) -> Delivery user 65 at 8:58
(node 117) -> Delivery user 17 at 9:06

VEHICLE # *45*

https://www.google.es/maps/dir/41.383134916016374,2.11290461849233/41.43357911443101,2.1498524233222063/41.43391626289473,2.143585782264866/41.41223776811001,2.126425686214522/41.423963218930936,2.1422678853887469/41.437290064808344,2.18791792341772/

(node 31) -> Pickup user 31 at 8:00
(node 131) -> Delivery user 31 at 8:18
(node 30) -> Pickup user 30 at 8:34
(node 88) -> Pickup user 88 at 8:43
(node 130) -> Delivery user 30 at 8:54
(node 188) -> Delivery user 88 at 9:10

VEHICLE # *46*

https://www.google.es/maps/dir/41.388825068540264,2.1764610839946936/41.40472210081946,2.2017315205332704/41.41100480337783,2.2048907493439813/41.442495017680734,2.2443844739352365/41.442156836419144,2.2432547154075357/41.417257657808264,2.132541947187826/

(node 71) -> Pickup user 71 at 7:57
(node 171) -> Delivery user 71 at 8:15
(node 70) -> Pickup user 70 at 8:25
(node 34) -> Pickup user 34 at 8:43
(node 170) -> Delivery user 70 at 8:45
(node 134) -> Delivery user 34 at 9:09

VEHICLE # *47*

https://www.google.es/maps/dir/41.382022701339054,2.1426726099068776/41.35578379623356,2.1361623252563455/41.35348463671869,2.134829036556043/41.41437675886846,2.219356777734983/41.453553976103414,2.1939977186499853/41.38587245766062,2.1225486267685705/

(node 43) -> Pickup user 43 at 7:57
(node 143) -> Delivery user 43 at 8:14
(node 32) -> Pickup user 32 at 8:18
(node 40) -> Pickup user 40 at 8:39
(node 132) -> Delivery user 32 at 8:53
(node 140) -> Delivery user 40 at 9:12

VEHICLE # *48*

https://www.google.es/maps/dir/41.35831009390188,2.144620859036082/41.393892054597444,2.114034190503707/41.42135421085965,2.1428740217355604/41.41787147219376,2.1467302046758694/41.42699952365187,2.1550637521945806/41.40680213056007,2.131796070816028/

(node 48) -> Pickup user 48 at 7:55

(**node** 75) -> Pickup user 75 at 8:12
(**node** 148) -> Delivery user 48 at 8:26
(**node** 175) -> Delivery user 75 at 8:33
(**node** 18) -> Pickup user 18 at 8:40
(**node** 118) -> Delivery user 18 at 8:54

VEHICLE # *49*

https://www.google.es/maps/dir/41.41612179282738,2.2050504756592564/41.456752168369675,2.250789056359562/41.41961831588011,2.2215375900288636/41.437518824420415,2.1902832857415007/41.44282367760574,2.172065842922952/41.40111355537748,2.1256263197207446/
(**node** 49) -> Pickup user 49 at 8:04
(**node** 149) -> Delivery user 49 at 8:15
(**node** 81) -> Pickup user 81 at 8:27
(**node** 181) -> Delivery user 81 at 8:42
(**node** 93) -> Pickup user 93 at 8:52
(**node** 193) -> Delivery user 93 at 9:07

VEHICLE # *50*

https://www.google.es/maps/dir/41.37109628041763,2.132485435662145/41.373237054314174,2.1417215625362047/41.40977852387572,2.139912153537806/41.441184813820634,2.1748425242556517/41.43284774126024,2.173638911836216/41.380330487162055,2.1213654303463 27/
(**node** 64) -> Pickup user 64 at 8:06
(**node** 78) -> Pickup user 78 at 8:14
(**node** 164) -> Delivery user 64 at 8:33
(**node** 178) -> Delivery user 78 at 8:47
(**node** 92) -> Pickup user 92 at 8:57
(**node** 192) -> Delivery user 92 at 9:23

VEHICLE # *51*

https://www.google.es/maps/dir/41.357887765051665,2.147701566782449/41.4241140151375,2.2188603579622708/41.42637139542616,2.2133194311660853/41.460836547287826,2.222863336300979/41.45883422721205,2.2204702333471276/41.36936552378374,2.165412012999483/
(**node** 68) -> Pickup user 68 at 8:08
(**node** 74) -> Pickup user 74 at 8:34
(**node** 168) -> Delivery user 68 at 8:41
(**node** 19) -> Pickup user 19 at 8:55
(**node** 174) -> Delivery user 74 at 8:58
(**node** 119) -> Delivery user 19 at 9:23
