UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Laboratori de IA

Pràctica 1

Pràctica de Cerca Local

Alex Herrero
Lluc Clavera
Pol Forner
Walter J.Troiani

Prof: Albert Calvo

2022/23 Q2

Contents

1 Introducció

Aquest escrit té el propòsit de resoldre un problema de l'optimització d'un sistema de transport privat compartit de cotxes per transportar-se fins al treball, tal que es maximitzi el nombre de cotxes plens i per conseqüència es redueixi el nombre de cotxes en la carretera i el consum de combustible. Per aconseguir aquesta fita, implementarem un algorisme d'intel·ligència artificial amb l'estratègia de cerca local, ja que es tracta d'un problema d'optimització molt complex que seria difícil de resoldre en temps raonable amb altres apropaments.

Aplicarem aquesta cerca local en l'espai de solucions en el qual ens desplaçarem usant una sèrie d'operadors i funcions heurístiques que nosaltres mateixos dissenyarem.

Per assegurar-nos de crear la millor solució empírica al problema debatrem en l'apartat d'experimentació diferents operadors, funcions heurístiques, solucions inicials i altres paràmetres.

2 Descripció del problema

2.1 Descripció formal del problema

2.1.1 Entrada del problema

El primer pas de la resolució de problemes és establir els elements que actuen. En aquest cas:

1. Una ciutat de mida $10x10Km^2$
2. Un nombre N de persones que volen anar a treballar
3. Un número M de persones que estan disposades a conduir per la ciutat per portar a tota la gent a treballar (inclosos ells mateixos). Cada un d'ells té un cotxe.
4. Cada persona té assignada la posició on és i la posició on ha d'anar a treballar
5. També considerem que $M < N$

Els conductors tenen capacitat per a 2 passatgers en el seu cotxe així que en la recollida de persones no poden haver-hi més de 2 persones en el mateix vehicle.

Els cotxes van a una velocitat fixa de 30 Km/h i comencen a conduir a les 7:00. Tots els treballadors han de ser-hi, com a molt, a les 8:00 al treball. El que fa que cada cotxe no pot fer més de 30Km (en el nostre cas com cada poma fa una mida de $1x1Km^2$ cada cotxe no pot fer més de 300 distàncies de Manhattan). Totes les nostres solucions estan basades en la distància màxima que pot recórrer un cotxe (ja que parlar sobre no arribar més tard de les 8:00 i parlar sobre una distància màxima 30Km són coses isomorfes). La *distància de Manhattan* es pot calcular com:

$$d(i,j) = |i_x - j_x| + |i_y - j_y|$$

El càlcul total de la distància del problema es mesura com la suma de les distàncies que fa cada cotxe (totes sumades com a distància de Manhattan). La distància total que fa cada cotxe es mesura com la suma de totes les distàncies de Manhattan que fa cada cotxe en deixar o agafar passatgers.

2.1.2 Objectiu de l'algorisme i descripció de la solució

L'objectiu d'aquest algorisme és donar, per a cada conductor, una ruta per la qual ha de recollir i deixar a passatgers. Tindrem en compte dos criteris per optimitzar:

- **Minimització de distàncies:** Cada cotxe intentarà recórrer la mínima distància possible
- **Minimització de distàncies i conductors:** Cada cotxe intentarà recórrer la mínima distància possible i a part també s'intentarà minimitzar la quantitat de cotxes utilitzats.

Considerarem solució:

- Una solució en la qual tots els habitants de la ciutat han entrat al treball a una hora de no més de les 8:00
- Cada cotxe ha de recollir i deixar al passatger. No poden haver-hi passatgers que mai baixen del cotxe.
- Les distàncies que recorre cada cotxe no excedeixen dels 30000m (ja que és la màxima distància que es pot recórrer amb una velocitat fixa de 30Km/h)
- Els conductors han d'acabar el seu trajecte al seu lloc de treball

Els càlculs de distàncies es fan amb distàncies de *Manhattan*. Considerem una ciutat amb *pomes* així que no podem fer distàncies euclidianes.

2.2 Justificació de l'ús de cerca local

El primer que hauríem de considerar quan ens encarem amb un problema complex és si es pot resoldre sense recórrer a la cerca en l'espai d'estats, és a dir si es pot aplicar algun esquema més simple com programàcio dinàmica, algorismes golafres... que ens resolguin el problema.

Si intentéssim plantejar una estratègia coneguda per intentar resoldre el problema en temps polinòmic ràpidament veuríem que no seria possible. Utilitzar un mètode avariciós (escollir a cada pas la millor opció segons un criteri)

no sempre ens portaria cap a la solució òptima, ja que una solució parcial òptima no sempre arriba a una solució òptima globalment. A més, partint d'una solució òptima tampoc podem dividir-la en subestructures òptimes per aprofitar-nos de la programació dinàmica.

Addicionalment, una observació del problema és que, en el cas de disposar només d'un conductor, és extremadament similar al problema del TSP, ja que ens trobem davant un problema en el qual un agent (cotxe) ha de passar per una sèrie de punts (cases i llocs de feina) minimitzant la distància total recorreguda (i tenint en compte que només ha de passar 1 cop per cada punt i tots els punts es poden considerar com a connectats).

Com que no estem en l'assignatura d'Algorísmia no profunditzarem en fer reduccions entre el TSP i el nostre problema, però la qüestió és que si només tinguéssim un sol cotxe ja es presentaria com un problema que possiblement no es pot resoldre en temps polinòmic. De fet, en el cas de tenir un sol cotxe i no poder portar més d'una persona a la vegada (o sigui només hi cap el conductor i una altra persona en el cotxe) es pot reduir el TSP a aquest problema de forma bastant directa (encara que deixem aquest detall com a curiositat i no profunditzarem en la demostració).

Tenint en compte que una versió simplificada del problema és tant o més difícil que el TSP, és justificable utilitzar un mètode de cerca en l'espai d'estats per resoldre el problema. Tot i això, encara no hem justificat del tot l'ús de cerca local, ja que la cerca heurística podria ser una opció viable, tenint en compte que ens garanteix trobar l'òptim, cosa que la cerca local no.

Tenint en compte la informació mencionada anteriorment, aquest problema és similar al TSP, i el TSP no és factible resoldre'l amb cerca heurística, per culpa de la gran dificultat de trobar un bon heurístic. Si ens parem a pensar en possibles heurístics per a una cerca heurística, ràpidament veuríem que els heurístics que ens podem plantejar serien similars als que podríem trobar per al TSP, és a dir, heurístics no massa bons.

Finalment, veiem que en aquest problema no és massa complicat trobar una solució inicial i que, amb un bon conjunt d'operadors, podem assegurar que es podria accedir a tot l'espai de solucions sense que sigui necessari sortir-se d'aquest espai. A més, cal destacar que la distància total és una

bona funció de qualitat continua i, a priori, tindrà poques mesetes.

Per tots els motius exposats anteriorment concloem que aquest problema és massa difícil per intentar trobar l'òptim global i, per tant, és una decisió perfectament justificable renunciar a trobar l'òptim global i buscar una bona aproximació. A més dintre dels mètodes que ens permeten arribar a solucions aproximades, l'ús de cerca local està perfectament justificat tenint en compte l'especificació del problema.

3 Disseny dels elements clau de la cerca

3.1 Representació de l'estat

3.1.1 Espai de solucions

Considerem una cota superior al nombre d'espais de solució:

Sigui N el nombre de persones que volen anar a treballar i M el nombre de persones que estan disposades a conduir en un mes. Una solució a la cota superior de l'espai de solucions del problema es dona quan per a cada conductor s'assignen les persones que s'han de recollir (òbviament un conductor sempre s'agafa a si mateix). Llavors una cota superior del problema es dona com la següent pregunta: De quantes formes M conductors poden agafar a N persones? Aquesta pregunta la podem plantejar en un altre forma que ens pot ajudar més: De quantes formes pots agafar M nombres naturals de tal forma que la seva suma sigui N important l'ordre i sense repeticions? Fer-se aquesta pregunta i fer-se la pregunta "de quantes formes s'expandeix el polinomi

$$(x_1 + x_2 + ... + x_M)^N$$

és exactament la mateixa pregunta (ja que quan s'expandeix un polinomi, la suma dels exponents de cada monomi és igual a N). Aquesta resposta es pot calcular a través del *teorema multinomial* i la seva resposta és:

$$\binom{N+M-1}{M-1}$$

.

Per què fer-se aquesta pregunta? Cada cotxe agafa un cert nombre de passatgers i al final la suma de tots els passatgers de cada cotxe ha de ser

igual al nombre de passatgers totals de la ciutat. És cert que hem d'afegir un terme factorial al nombre binomial per ajustar-ho més al problema (ja que, per cada combinació de nombres, l'ordre en què agafem aquests números també importa així que ho tenim en compte amb el terme factorial). Així que la resposta final al problema és que la cota superior a l'espai de solucions és:

$$(\frac{N}{M})! \cdot \binom{N+M-1}{M-1}$$

.

Destacar, per finalitzar, que aquí tenim en compte estats que no seran possibles, però al final volem donar una cota superior.

3.1.2 Tipus de representació considerats

A l'hora de fer una cerca per l'espai de solucions és molt important tenir en compte la representació de l'estat, ja que el programa generarà una quantitat molt gran d'ells. Si guardem massa informació innecessària dins de l'estat podríem arribar al límit de memòria, d'altra banda, si intentem estalviar massa informació podríem acabar gastant molt de temps en computar dades bàsiques de l'estat.

Precisament per estalviar memòria, teníem clar des del principi que la informació dels usuaris (Casa, lloc de treball i informació sobre si és conductor o no), no hauria d'estar inclosa dins de l'estat. L'estat només haurà de guardar el trajecte que realitza cada cotxe, és a dir, les persones que agafa i deixa (i en quin ordre ho fa).

Per representar el trajecte de cada cotxe, el més natural és presentar-ho com una llista de trajectes. Ara bé, per representar un trajecte vam considerar 3 opcions diferents.

La primera opció, i la més naïf, seria representar el trajecte com una llista de les posicions que recorre cada cotxe (quins carrers recorre). Aquesta representació implicaria que cada cotxe es guardés una llista de, com a molt, 300 posicions, ja que un cotxe pot recórrer 300 centenars de metres abans d'arribar a la seva feina, és a dir, pot recórrer 300 posicions diferents.

Realment no és necessària tanta informació, ja que sabem quina és la

solució òptima per anar d'un punt a un altre (distància Manhattan) i escollir qualsevol altre camí que no sigui l'òptim només empitjoraria la solució. Per aquest motiu vam plantejar una segona solució on cada trajecte indiqués la llista de posicions *clau* per on passava, és a dir les posicions de les cases i els llocs de treball de cada usuari recollit pel cotxe. La distància total del trajecte seria, per tant, la suma de les distàncies entre un punt i el següent de la llista.

L'anterior representació sembla que serveix, però vam decidir simplificar-la una mica perquè fos més comprensible i més fàcil de treballar amb ella, la representació que considerarem d'ara en avant serà, per tant, la següent:

Un trajecte es representa com una llista d'enters. Cada enter correspon a un identificador d'un usuari o a un identificador negatiu d'un usuari. Un nombre positiu en la posició j de la llista significa que en aquella posició de la llista s'ha recollit a l'usuari amb identificador trajecte[j]. Per altra banda, un identificador negatiu representa que s'ha deixat a l'usuari amb identificador -trajecte[j].

D'aquesta manera podem portar un registre no només de les posicions que recorre un cotxe, sinó també de les persones que recull i deixa. Com que es guarda un conjunt d'usuaris dintre de la classe estat de forma estàtica (només una còpia per a tots els estats), podem obtenir immediatament la posició on es troba cada persona per poder calcular fàcilment les distàncies.

Finalment, hem decidit guardar, per cada trajecte, la distància total que recorre, per estalviar-nos fer el càlcul cada cop que volem calcular l'heurístic i també per facilitar la comprovació de la solució.

En total la memòria que gastem en la representació serà $2N + O(M)$ enters, ja que per cada usuari (els hem d'agafar tots) hem de guardar 2 identificadors (2*N enters en total) en l'estat: Un per indicar que l'agafem i un altre per indicar que el deixem. A més, hem de guardar un enter per cada conductor actiu indicant la distància del seu recorregut, encara que en aquest cas és possible tenir menys de M conductors actius (per això considerem O(M) enters).

3.2 Solució Inicial

Hem decidit crear dues estratègies similars per a la solució inicial. Una estratègia que normalment donarà solució en poc temps i en molts casos serà correcta (complirà amb totes les restriccions) és la següent: Utilitzem tots els cotxes i repartim els passatgers de forma equitativa a cada cotxe, d'aquesta manera tots els cotxes tindran aproximadament la mateixa càrrega i probablement podrem trobar una solució inicial ràpidament.

L'assignació d'usuaris a cotxes es fa de forma seqüencial, és a dir, un cotxe va agafant (i deixant) usuaris fins que hagi agafat a tots els usuaris que li toquen. Això sí, per escollir quins usuaris porta cada cotxe hem considerat dues estratègies diferents:

Estratègia golafre Cada conductor considera la millor opció que pot fer: Si no té cap passatger i encara ha d'agafar passatgers, llavors va pel que té més a prop. Si té dos passatgers llavors deixa primer el que té més a prop i, si només té un passatger, llavors decideix si deixar al passatger que té o bé agafar a un de nou (aquesta decisió es basa en fer l'acció que menys distància hagi de recorrer).

Estratègia Random Si el conductor té dos passatgers llavors deixa a un de forma aleatòria, si no té cap passatger llavors en recull un de forma aleatòria i sí, finalment, només té un passatger, llavors decideix de forma aleatòria si deixar a aquest passatger o recollir a un altre (escollit òbviament de forma *random*). El conductor pararà de recollir usuaris un cop hagi recollit tots els que havia de recollir.

L'estratègia golafre, per la seva natura, donarà millor solució inicial que l'aleatòria. En la golafre en tot moment es vol fer una minimització de la distància que ha de recórrer un cotxe, cosa que en la golafre no es fa i és "lliure elecció" escollir sempre que s'arribi a una solució correcta. Per tant, en tots els casos, la golafre generarà una solució millor.

Estratègia golafre Cada cotxe mira tots els possibles usuaris que pot agafar i decideix quin és el millor. Donat M cotxes es miren $O(N)$ usuaris. Pe tant aquí tenim un cost de $O(NM)$. Però com repartim equitativament cada conductor només mirarà la llista $O(\frac{N}{M})$ vegades. El que dona com a resultat un cost de $O(NM\frac{N}{M})$ o cosa que és el mateix: $O(N^2)$.

Estratègia Random Aquesta estratègia pot ser que no acabi, ja que genera una solució qualsevol i si no és llavors gènera un altre. Podem tenir la mala sort que totes les configuracions no siguin mai solució per tant pot ser que no acabi. Però generalment quasi totes les reparticions equitatives són solució.

3.3 Conjunt i subconjunt d'operadors

Pel nostre algorisme de Hill Climbing hem plantejat l'ús de 3 operadors diferents, amb una possible petita variació en el tercer operador. Els operadors plantejats són els següents:

3.3.1 Permutació de ruta

El primer operador que ens hem plantejat és el de permutar un sol trajecte de la solució. La permutació de ruta rep 3 paràmetres: idRuta, id1 i id2. El paràmetre idRuta indica quin trajecte es desitja permutar, id1 indica la primera posició a permutar i id2 indica la segona. Amb aquests paràmetres s'intercanvia l'acció realitzada a la posició id1 amb l'acció realitzada en la posició id2 del trajecte.

Cal destacar que després de la permutació no pot passar el següent:

- No es pot deixar un usuari abans de recollir-lo (-Usuari no pot aparèixer abans que Usuari)
- No es pot superar la capacitat màxima del cotxe
- La distància total recorreguda no pot superar els 300 centenars de metres
- El conductor és inamovible, és a dir id1 i id2 no poden ser ni la primera ni l'última posició del trajecte.

Si en aplicar-se l'operador es compleixen les tres condicions anteriors, llavors l'operador és aplicable.

A més, és important considerar que l'aplicació de l'operador és simètrica, és a dir permutació(idCar, id1, id2) és equivalent a permutació(idCar, id2, id1). Això implica que només s'han de provar les $\frac{n(n-1)}{2}$ parelles possibles de 2 identificadors diferents, en lloc de les n^2 combinacions que obtindríem sense considerar simetries.

Tenint en compte la informació anterior, el factor de ramificació seria: $O(M * \frac{u_i(u_i-1)}{2})$, on M és el nombre de trajectes i u_i és la mida d'un trajecte individual. La mitja de u_i és $\frac{2N}{M} - 2$, ja que cada usuari apareix 2 vegades (per agafar-lo i deixar-lo) i el conductor no és seleccionable. Tenint aquesta informació en ment podríem considerar que el factor de ramificació mig seria $M * \frac{(\frac{2N}{M}-2)(\frac{2N}{M}-3)}{2}$ que, asimptòticament equivaldria a $O(M * (\frac{2N}{M})^2) = O(\frac{4N^2}{M}) = O(\frac{N^2}{M})$

3.3.2 Intercanvi de passatjers

El segon operador plantejat és el d'intercanviar 2 usuaris entre 2 cotxes. Lògicament, si intercanvies dues persones hauràs d'intercanviar tant el punt de recollida com el punt de deixada.

L'operador d'intercanvi rep 4 paràmetres: idCar1, idCar2, id1, id2. idCar1 i idCar2 representen els 2 cotxes (o els 2 trajectes) dels quals es desitja fer un intercanvi, mentre que id1 representa la posició del primer trajecte i id2 la posició del segon trajecte.

Amb els anteriors paràmetres, l'operador seleccionarà els usuaris que es recullen en les posicions id1 i id2 de les seves respectives rutes i intercanviarà els 2 usuaris. Cal destacar que intercanviar els 2 usuaris implica intercanviar el contingut d'id1 amb el de id2, però també implica intercanviar l'acció de deixar a la feina l'usuari 1 amb l'acció de deixar a la feina l'usuari 2.

Les condicions d'aplicabilitat d'aquest operador són les següents:

- La distància total recorreguda de cap dels 2 trajectes no pot superar els 300 centenars de metres després de l'intercanvi
- El conductor és inamovible, és a dir id1 i id2 no poden ser ni la primera ni l'última posició del trajecte.

- Hem decidit que les accions que es guarden en trajecte1[id1] i trajecte2[id2] sempre han de ser accions de recollir un usuari, ja que com s'han de moure tant l'acció de recollida com la de deixada seria simètric tenir en compte que id1 i id2 poden representar accions de deixar usuari i, a més, considerar que tot són accions de recollir simplifica la implementació de l'operador.

Si prenem com la mitja d'usuaris elegibles de cada cotxe com a $\frac{N}{M} - 1$ (En aquest cas només hem de tenir en compte el punt en el qual es recull cada persona, exceptuant el conductor) el factor de ramificació seria $M^2 * (\frac{N}{M} - 1)^2$ que equivaldria, asimptòticament, a $O(M^2 * (\frac{N}{M})^2) = O(N^2)$. Addicionalment, caldria destacar que aquest intercanvi també és simètric, és a dir seria necessari comprovar només $\frac{M*(M-1)}{2}$ parelles de cotxes, encara que asimptòticament és equivalent.

3.3.3 Canvi de cotxe

L'últim operador que ens vam plantejar és el de canviar un usuari de cotxe.

Cal ressaltar que amb la combinació dels dos operadors anteriors era impossible explorar tot l'espai de cerca, ja que al fer només intercanvis, el nombre de passatgers d'un cotxe en concret es mantindrà constant. Per aquest mateix motiu se'ns escaparien moltes possibles solucions en les quals els usuaris no estan distribuïts equitativament i també se'ns fa impossible minimitzar el nombre de cotxes utilitzats.

Per tots aquests motius hem inclòs un operador que ens permeti moure una sola persona de cotxe, d'aquesta manera podem augmentar el nombre d'usuaris que viatgen en un sol cotxe movent els usuaris d'un en un, fent que la distribució pugui deixar de ser equitativa i també fent que es puguin eliminar cotxes (explicarem com més endavant).

L'operador rep 3 paràmetres: idCarOrigen, idCarDestí, idUsuari. Aquests paràmetres (per ordre) representen l'identificador del cotxe on es troba l'usuari que volem moure, l'identificador del cotxe destí i l'identificador de l'usuari dins del trajecte (Bàsicament, l'usuari que es deixa o recull a la posició idUsuari del trajecte del cotxe idCarOrigen es mourà al trajecte idCarDestí).

El funcionament de l'operador és similar al d'intercanvi, ho podríem veure com fer un intercanvi amb un *espai buit* del cotxe destí, de tal manera que el cotxe guanya un nou *espai buit* i el cotxe destí perd una part del seu potencial per portar passatgers.

Per executar l'operador haurem de col·locar a l'usuari (tant la recollida com la deixada) en algun punt del trajecte del cotxe destí. Per a efectuar aquesta operació ens vam plantejar 2 formes de fer-ho:

- El nou cotxe destí recollirà i deixarà al nou usuari al final del seu trajecte, just abans d'anar a la seva feina (la feina del conductor).
- El conductor del cotxe destí recollirà i deixarà al nou usuari just al principi del seu trajecte, és a dir, recollirà al nou usuari just després de "recollir-se" a ell mateix i el deixarà just abans d'agafar l'anterior primer usuari del trajecte.

Decidirem mitjançant experimentació quina de les 2 opcions és la més convenient per la nostra modelització del problema.

Si en aplicar l'operador, el cotxe destí es queda amb un sol passatger (el conductor), llavors es reassignarà el conductor a un altre cotxe (al final o al principi d'un altre trajecte, de la mateixa manera que amb la resta de passatgers) i s'eliminarà aquell cotxe de la solució. Cal destacar que en eliminar un conductor mai es podrà tornar a afegir, encara que com comencem amb tots els conductors ocupats sempre podrem explorar un conjunt prou gran de l'espai de cerca amb els nostres operadors.

Les condicions d'aplicabilitat, molt similars a les de l'anterior operador, concretament són les següents:

- La distància total recorreguda del trajecte destí no pot superar els 300 centenars de metres després de l'intercanvi
- En aquest cas el conductor es pot moure, però idCarOrigen només podrà representar el conductor en cas que sigui l'únic passatger restant.
- Com en el cas de l'operador anterior idCarOrigen ha de representar l'acció de recollida.

Respecte al factor de ramificació, tenint en compte la mateixa mitjana d'ocupació que en el cas anterior, seria $M^2 * \frac{N}{M}$ que equivaldria a $O(MN)$.

3.3.4 Observacions sobre el conjunt d'operadors

És important tenir en compte que cap dels 3 operadors plantejats, per si sol, pot aconseguir explorar tot l'espai de solucions.

L'operador de canvi de cotxe és imprescindible per reduir el nombre de cotxes totals, mentre que l'operador de permutació és necessari per canviar l'ordre d'un trajecte concret i així modificar a distància.

Un detall a tenir en compte és que fer un intercanvi entre 2 persones del mateix cotxe implicaria la generació d'una permutació, però com que l'intercanvi mou tant les posicions de recollida com les de deixada, no es poden assolir totes les permutacions d'aquesta manera. D'altra banda, intercanviar dos usuaris de cotxes diferents es pot veure com fer un canvi de cotxe i després fer una permutació.

Malgrat això, com els nostres operadors només són aplicables en cas que resultat no superi el límit de 300, en alguns casos seria impossible fer un intercanvi utilitzant la combinació de permutació i canvi de cotxe. Aquesta impossibilitat es pot veure observant que l'operador de canvi de cotxe sempre augmenta la distància total d'un trajecte mentre que redueix la distància d'un altre (la distància total pot reduir, però una distància individual sempre augmentarà), si es vol fer l'intercanvi entre 2 cotxes amb un recorregut proper al màxim permès, pot resultar impossible fer l'intercanvi, ja que la distància d'un dels 2 trajectes hauria d'augmentar. Si, en canvi, utilitzéssim l'operador d'intercanvi podríem reduir la distància individual dels 2 trajectes.

Com a últim detall, és important considerar que no tenim operador d'afegir cotxe. Aquest operador podria semblar necessari per recórrer tot l'espai de solucions, però com, amb la nostra solució inicial, sempre utilitzem el màxim nombre de cotxes permès, hem considerat que no ens feia falta, ja que augmentar el nombre de cotxes seria equivalent a no reduir-lo en un principi (i fer permutacions i intercanvis per arribar a la mateixa solució).

3.4 Funció de qualitat

Donat l'enunciat, es presenten 2 criteris a minimitzar: la suma de les distàncies dels recorreguts de tots els cotxes o bé aquesta suma i el nombre de cotxes emprats pel servei de compartició.

Inicialment, vam pensar forces heurístics senzills i ràpids, però que per la seva simplicitat, no oferien els millors resultats ni tampoc els que esperàvem, llavors vam triar els que teníem més certesa que funcionarien bé i dels quals esperaríem bons resultats amb poca complexitat temporal (Per tal de reduir el temps d'execució final). D'entre aquests heurístics vam triar:

1. **Sumatori de distàncies recorregudes:**
   $h_1(n) = \sum_{C \in cotxes} distancia(C)$
   És l'heurístic més simple en quant a distàncies recorregudes, contra menor sigui millor i contra major sigui pitjor (La versió quadràtica d'aquest va ser descartat ja que proporcionava, contraintuïtivament, pitjors solucions finals)

2. **Entropia de Shannon per distàncies recorregudes:**
   $h_2(n) = \sum_{C \in cotxes} distancia(C) \cdot log(distancia(C))$
   Aquesta funció, per la seva distribució probabilística recompensarà estats on aquesta suma no és ni màxima ni mínima, sinó una repartició més equitativa. Llavors, cadascun d'aquests cotxes recorreran una distància més mitjana i més semblant als altres i no hi haurà tanta disparitat. Inicialment, vam pensar que a pesar que no fos probable l'òptim global, optimitzaria en certs casos i en algunes situacions en concret podria ser útil, però evidentment no és ni serà el millor heurístic.

3. **Sumatori de distàncies ponderant la màxima distància**
   $h_3(n) = \sum_{C \in cotxes} distancia(C) + \lambda \cdot max(distancia(C))$
   Semblant al primer, afegeix una ponderació λ al cotxe amb major distància, per sancionar estats on el cotxe amb major distància recorreguda és massa gran. Podria aconseguir bons resultats a causa de l'optimització de la distància màxima.

4. **Sumatori de distàncies per nombre de cotxes**
   $h_4(n) = M \cdot \sum_{C \in cotxes} distancia(C)$

Semblant al primer heurístic però tenint en compte també el nombre de cotxes, minimitzant ambdues unitats simultàniament.

5. **Sumatori de distàncies per nombre de cotxes al quadrat**
   $h_5(n) = M^2 \cdot \sum_{C \in cotxes} distancia(C)$
   Igual que l'anterior, però donant-li molta més rellevància al factor nombre de cotxes, ja que està al quadrat, llavors penalitza fortament solucions on el nombre de cotxes sigui alt. Creiem que pot ser millor que l'anterior.

El paràmetre λ s'usarà de manera experimental per ponderar la importància del cotxe amb recorregut més llarg, el qual creiem que serà un factor de pes en obtenir solucions amb una suma de distàncies petita i possiblement superior als altres heurístics.

- $\lambda < 1$: Atribueix menys importància a la distància màxima, per valors de lambda molt petits seria el mateix que l'heurístic anterior.
- $\lambda = 1$: Afegeix un cert pes a l'heurístic (duplica el valor sumat màxim).
- $\lambda > 1$: Atribueix més importància a la distància màxima, podent minimitzar efectivament aquesta per valors grans de lambda.

En l'apartat d'experimentació es posaran a prova i es compararan de manera estadística per saber quins donen millors resultats a la pràctica.

4 Experimentació

4.1 Entorn d'Experimentació i metodologia

Tota l'experimentació ha estat realitzada en un subsistema d'ubuntu dins de Windows amb WSL 2. WSL 2 seria com una màquina virtual lleugera d'aquesta distribució

Les especificacions de l'ordinador sobre el qual s'ha executat el sistema virtual són les següents:

- **Processador:** Intel Core i7-12700K 3.60 GHz
- **Memòria RAM:** 32,0 GB

- **S.O:** WSL 2: Ubuntu 20.04 LTS on Windows 10 x86_64
- **Versió del Kernel de Linux:** 5.10.60.1-microsoft-standard-WSL2

Les especificacions de la versió de Java que hem utilitzat són les següents:

- **Versió:** 17.0.6 2023-01-17 LTS
- **Java(TM) SE Runtime Environment:** build 17.0.6+9-LTS-190
- **Memòria assignada:** 3.88 GB, excepete en un cas on s'han utilitzat 16 (especificat més endavant).
- **Càlcul del temps d'execució:** Llibreria System de Java (utilitzant currentTimeMillis)

Un detall important a destacar sobre la metodologia de tots els experiments, és que no hem utilitzat el generador de nombres aleatoris de Java per a generar les llavors, sinó que les hem generat a part i les hem inclòs dins del codi. Hem decidit fer-ho així per assegurar-nos que qualsevol execució d'un mateix experiment donarà els mateixos resultats (o com a mínim similars), i d'aquesta manera podem repetir i estendre els mateixos experiments més fàcilment.

A més, alguns experiments comparteixen llavors, aquesta compartició de llavors ens serveix per, potencialment, poder comparar els resultats de 2 experiments diferents (ja que s'hauran fet amb les mateixes llavors).

En resum, hem reduït lleugerament l'aleatorietat dels experiments per afavorir la facilitat per replicar-los, estendre'ls i comparar-los entre ells.

A més, hem utilitzat MATLAB per al tractament de les dades. Deixarem a la carpeta del projecte l'script de MATLAB utilitzat.

Abans de començar a explicar els experiments, és important destacar que hem recollit una quantitat elevada de dades, però les dades realment importants s'obtenen després d'un cert tractament (mitjanes, desviacions, gràfiques...).

Per aquest motiu hem decidit no incloure les dades directament en aquest document i treballar directament amb gràfiques i mitjanes. Si a algun lector l'interessen les dades específiques obtingudes, poden consultar-les en l'Excel que trobaran en la carpeta del projecte.

Finalment, hem afegit les imatges de totes les gràfiques (inclús algunes que no hem acabat utilitzant) en la carpeta del projecte.

4.2 Experiment 1: Experimentació d'operadors

Per a la cerca local, els operadors seleccionats afecten molt tant en el temps de cerca com en la qualitat de la solució final. Operadors molt senzills amb factors de ramificació no molt elevats poden fer que la cerca sigui molt ràpida, però també corren el risc de quedar-se encallats en mínims locals amb facilitat.

En el nostre cas, el conjunt d'operadors el tenim bastant ben definit, però hi ha hagut una decisió, per al tercer operador, que no teníem massa clara, és per això que volem provar experimentalment quina de les dues alternatives funciona millor.

El nostre operador de canviar de cotxe mou un usuari d'un trajecte a un altre, el dubte està que no tenim del tot clar a quin punt del nou trajecte s'hauria de col·locar l'usuari. Hem pensat que moure l'usuari tant al principi del nou trajecte (just després que es pugi el conductor) com al final del trajecte (just abans que el conductor vagi a la seva feina) serien opcions viables.

Cal destacar que combinant l'operador de canviar de cotxe amb el de permutar la ruta aconseguim col·locar al nou usuari en qualsevol punt de la nova ruta, però és possible que una opció optimitzi més el cost i, per tant, hi haurà més probabilitat que s'executi, provocant, potencialment, que s'explori una secció més amplia de l'espai de solucions.

Per aquest experiment fixarem el mètode d'inicialització a la solució golafre (ja que, a diferència de l'altre mètode, és determinista) i utilitzarem l'heurístic més simple, el de suma de distàncies de tots els trajectes.

El plantejament de l'experiment és el següent:

Observació	El punt on es mou el nou usuari en l'operador de canvi de trajecte pot influir tant en el temps d'execució com en la qualitat de la solució final.
Plantejament	Compararem l'opció de moure al principi i l'opció de moure al final
Hipòtesi	· Moure al principi i moure al final són equivalents en qualitat · Moure al principi i al final són equivalents en temps.
Mètode	· Escollim 10 llavors aleatòries · Per a cada llavor executem Hill Climbing amb els 2 operadors i mesurem temps i qualitat · Considerem sempre 100 conductors i 200 usuaris

Table 1: Plantejament de l'experiment 1

Després d'executar l'experiment hem pogut obtenir els següents resultats:

	Moure principi	Moure final
Suma de distàncies (Centenars de m)	10914.6	10695.3
Temps (segons)	8.0877	9.0076

Table 2: Mitjanes de l'experiment 1

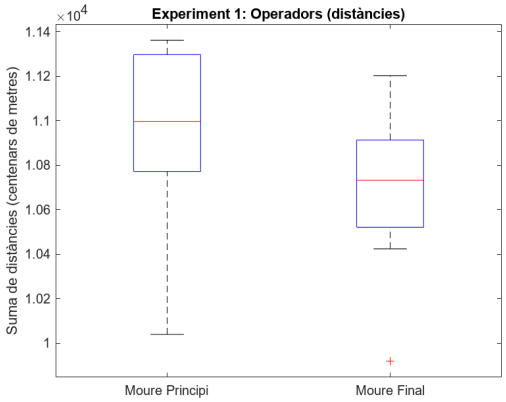


Figure 1: Boxplot de la qualitat de la solució

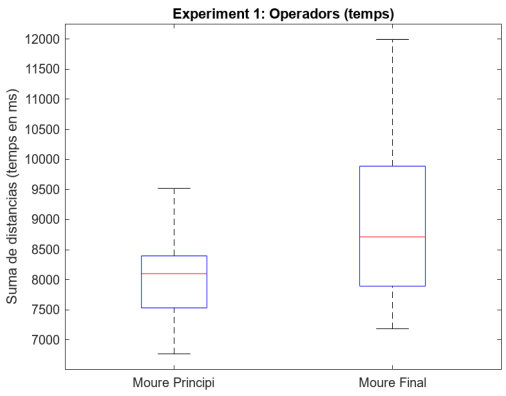


Figure 2: Boxplot del temps d'execució

En els 2 boxplots següents podem apreciar una diferència entre les 2 variants de l'operador. A priori sembla que l'opció de moure el nou usuari al

final minimitza la distància total, mentre que triga més que l'alternativa.

Igualment, per ser rigorosos, hem realitzat un test per a variables aparellades utilitzant la distribució T-student, per comprovar si realment podem descartar la hipòtesi nul·la de que són mètodes equivalents.

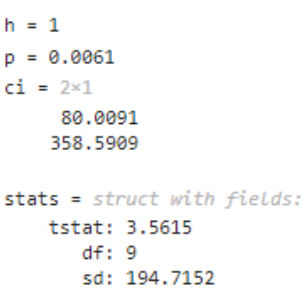


Figure 3: t-test sobre la diferència de distàncies

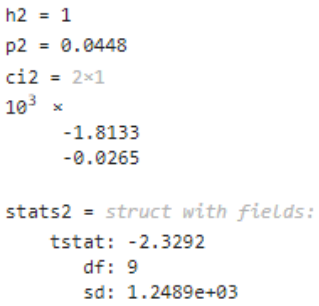


Figure 4: t-test sobre la diferència de temps

En aquest cas, MATALB (script adjuntat en la carpeta del projecte) ens diu que el p-valor per acceptar la hipòtesi nul·la és de 0.0028 per al cas de la suma de distàncies i de 0.0277 per al temps d'execució. Això ens porta a haver de descartar la hipòtesi nul·la ($h = 1$) si utilitzem un risc d'error del 5%.

A més, MATLAB ens dona intervals per a la diferència mitjana de la suma de distàncies i per a la diferència de temps d'execució amb confiança 95%, aquests intervals ens diuen que l'opció de moure al principi obté, de mitjana, suma de distàncies més elevades però temps més ràpids.

Com que el nostre interès principal és obtenir bones solucions, i estem disposats a assumir l'augment del cost temporal, ens quedem amb la segona opció de l'operador: moure l'usuari al final del nou trajecte.

Al principi, nosaltres teníem la hipòtesi que no hi hauria diferencia alguna entre els 2 operadors, però després de fer els experiments hem pogut comprovar que, efectivament, moure una persona al final és més efectiu.

4.3 Experiment 2: Les dues solucions

Com ja hem explicat anteriorment, la nostra solució inicial reparteix equitativament a totes les persones entre el nombre de cotxes disponibles, omplint-los tots. Tot i això, ens hem plantejat dos mètodes diferents per distribuir als usuaris en els cotxes: Un mètode completament aleatori i un mètode voraç.

L'objectiu d'aquest segon experiment és veure quina de les 2 estratègies d'inicialització ens dona millors resultats. El nostre interès principal és comprovar quina de les dues solucions ens aporta una qualitat de solució superior, però també creiem que hi haurà una diferència de temps notable que també comprovarem.

A més, està clar que l'estratègia aleatòria donarà un major rang de solucions possibles. Per aquest motiu també ens interessa veure si el seu rendiment en mitjana pot ser superior o igual al de la solució golafre, encara que hi hagi casos en els quals funcioni pitjor, en el que a optimització de distàncies es refereix.

En aquest experiment hem fixat l'heurístic a suma de distàncies i els operadors als resultats de l'anterior experiment (canvi de cotxe mou a l'usuari al final del trajecte destí). Tenint això en compte hem plantejat l'experiment de la següent manera:

Observació	El mètode d'assignació de persones a cotxes pot influir tant en el temps d'execució com en la qualitat de la solució final.
Plantejament	Compararem el mètode *golafre* i el mètode *random*
Hipòtesi	· La solució *golafre* i la *random* són diferents en quant a qualitat · La solució *golafre* i la *random* són diferents quant a temps.
Mètode	· Escollim 10 llavors aleatòries · Per a cada llavor executem Hill Climbing 1 cop amb la solució golafre i mesurem temps i qualitat · Per a cada llavor executem Hill Climbing 3 cops amb la solució random i mesurem la mitjana de temps i qualitat · Considerem sempre 100 conductors i 200 usuaris

Table 3: Plantejament de l'experiment 2

Després d'executar els experiments i recopilar les dades, hem obtingut els següents resultats:

	golafre	*Random*
Suma de distàncies (Centenars de m)	10669,1	10813,2
Temps (segons)	9'6475	10'7257

Table 4: Mitjanes de l'experiment 2

En els 2 boxlpots següents podem observar com la solució golafre sembla ser força millor en temps d'execució (encara que amb major variància) i una mica superior en distàncies. Igualment, farem t-tests per veure si podem afirmar de veritat que la solució golafre és millor que la random.

Els t-tests direccionals ens mostren que, amb p-valor de 0.98, podem acceptar les dues hipòtesis nul·les, és a dir, sembla que l'opció golafre sí que produeix millors resultats que la random, tant en temps com en bondat de la solució.

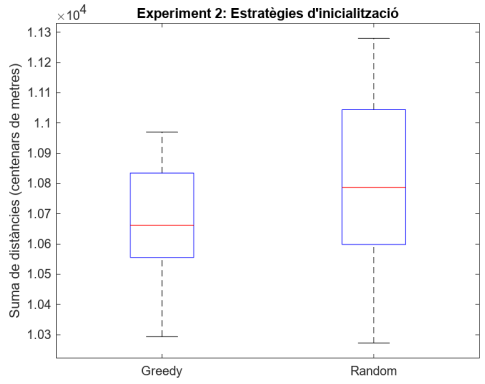


Figure 5: Boxplot de la distància recorreguda en solució final

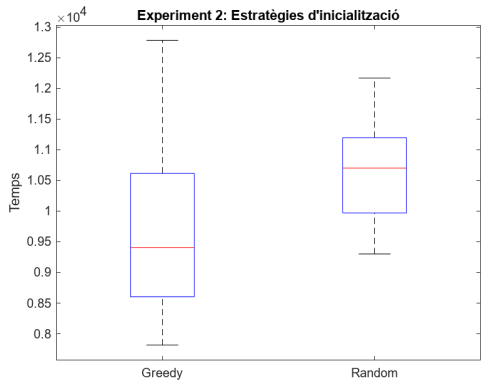


Figure 6: Boxplot del temps trigat en solució final

Nosaltres ja ens esperàvem que la solució golafre fos més ràpida, ja que comença des d'una solució millor i, per tant, més propera a l'òptim. Igualment, teníem la idea que, en ser una solució inicial més propera a un òptim, la golafre potser es quedaria en solucions finals pitjors.

```
h = 0
p = 0.9839
interval = 2×1
   -248.5386
         Inf

stats = struct with fields:
    tstat: -2.5293
       df: 9
       sd: 180.1656
```

Figure 7: t-test sobre les distàncies

```
ht = 0
pt = 0.9847
intervalT = 2×1
10^3 ×
     -1.8498
         Inf

statsT = struct with fields:
    tstat: -2.5615
       df: 9
       sd: 1.3311e+03
```

Figure 8: t-test sobre el temps trigat

Al final hem pogut comprovar que la nostra idea era falsa i la solució golafre no acostuma a acabar en mínims locals pitjors que en el cas de la solució aleatòria.

4.4 Experiment 3: Heurístic λ

La funció heurística del sumatori de distàncies ponderant la màxima distància conte un paràmetre λ el qual és un factor multiplicatiu a la distància màxima.

L'objectiu d'aquest experiment és observar si existeix una millora de la qualitat de la solució en variar el valor de λ. Només ens fixarem en la qualitat, no té sentit observar el temps d'execució, ja aquest no perquè variar massa.

El plantejament de l'experiment és el següent:

Després d'executar l'experiment hem pogut obtenir els següents resultats:

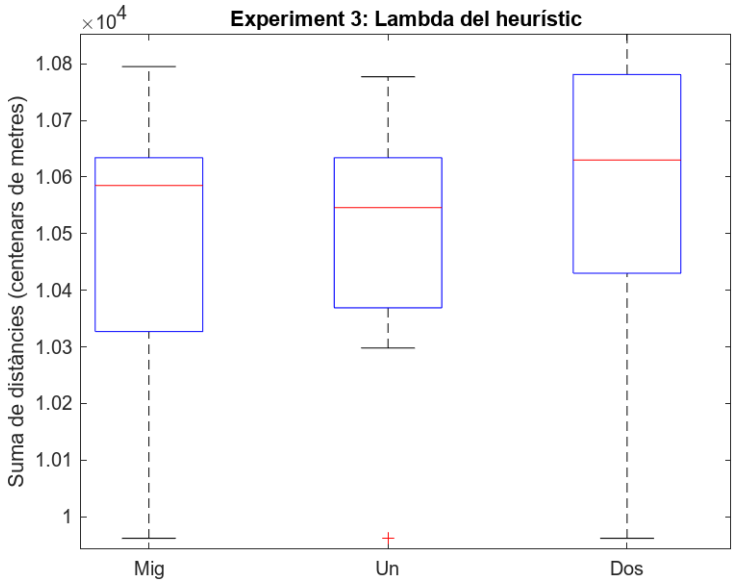
Com podem observar amb el gràfic, amb els tres valors de λ els resultats són molt similars. Podríem concloure que amb $\lambda = 1$ la variància i la mitjana són millors, però tampoc és res substancialment significatiu. Després d'aquest experiment ja podem començar a pensar que potser l'heurístic de sumatori de distàncies ponderant la màxima distància no és massa interessant.

Observació	La variable λ pot arribar a influir en la qualitat de la solució final si es modifica el seu valor.
Plantejament	Compararem diferents valors de λ per veure si veiem una relació a l'hora d'augmentar o disminuir el valor de λ
Hipòtesi	· Hi ha una relació entre el valor de λ i la qualitat de la solució final
Mètode	· Escollim 10 llavors aleatòries · Per a cada llavor executem Hill Climbing per a cada heurístic i mesurem la qualitat · Considerem sempre 100 conductors i 200 usuaris

Table 5: Plantejament de l'experiment 3

	$\lambda = 0.5$	$\lambda = 1$	$\lambda = 2$
Suma de distàncies (Centenars de m)	10489.5	10484.1	10592.6

Table 6: Mitjanes de l'experiment 3



Els t-tests que hem realitzat per comprovar les diferències tampoc ens aporten massa informació, ja que amb els diversos paràmetres de lambda s'obtenen valors molt similars. Si algun lector vol veure els t-tests els pot

trobar en l'script de MATLAB, però no els hem inclòs perquè aquest experiment no tindrà massa rellevància en un futur.

4.5 Experiment 4: Heurístics per minimitzar la distància

Ara després d'haver experimentat amb diferents elements, hem d'experimentar per decidir si hi ha un heurístic millor per optimitzar el primer criteri, que és que la suma de distàncies sigui mínima. Com hem mencionat anteriorment, tenim 3 heurístics que tenen aquest objectiu: sumatori de distàncies recorregudes, un heurístic basat en l'entropia de Shannon per distàncies recorregudes i el sumatori de distàncies ponderant la màxima distància.

El plantejament de l'experiment és el següent:

Observació	Hi ha heurístics que milloren la distància final del problema millors que altres
Plantejament	Compararem els diferents heurístics per veure la qualitat d'aquests
Hipòtesi	· Suma de distàncies és el millor · Entropia serà el segon millor. · $\lambda = 1$ serà el pitjor.
Mètode	· Escollim 10 llavors aleatòries · Per a cada llavor executem Hill Climbing i mesurem la qualitat · Considerem sempre 100 conductors i 200 usuaris

Table 7: Plantejament de l'experiment 4

Després d'executar l'experiment hem pogut obtenir els següents resultats:

	Suma de distàncies	Entropia	$\lambda = 1$
Suma de distàncies (Centenars de m)	10612,3	10921,2	10638,1

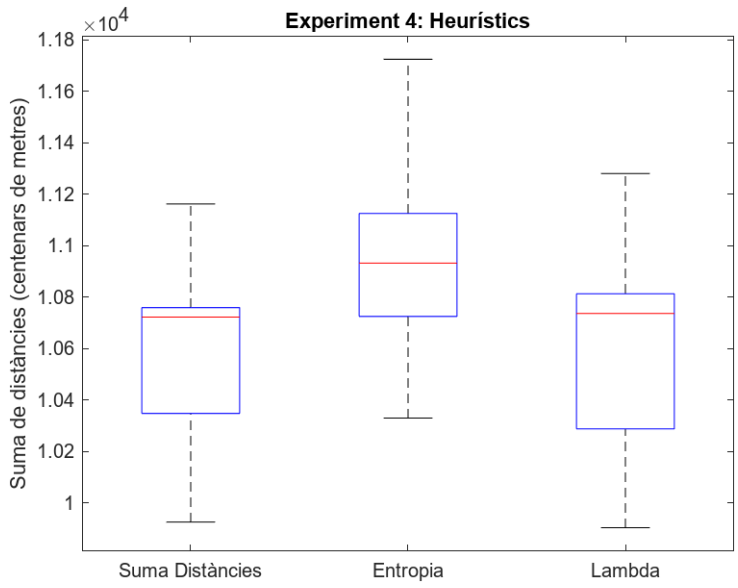
Table 8: Mitjanes de l'experiment 4

Aquí ja podem observar una clara distinció entre la suma de distàncies i lambda amb l'entropia. Al contrari al que pensàvem a la hipòtesi (que el

pitjor era la lambda), veiem clarament que el pitjor és l'entropia.

D'aquest experiment i veien les comparatives, ens hem de quedar amb la suma de distàncies o la lambda. No podem rebutjar la hipòtesi que són diferents, però sí que es decanta una mica més cap a la suma de distàncies. Per tant, concloem en què la suma de distàncies és el millor heurístic que optimitza el primer criteri.

Els t-tests que es mostren a continuació mostren com podem assegurar amb força seguretat que la suma de distàncies produeix millors resultats que l'entropia, però no podem confirmar ni descartar que sigui millor que l'heurístic amb lambda.



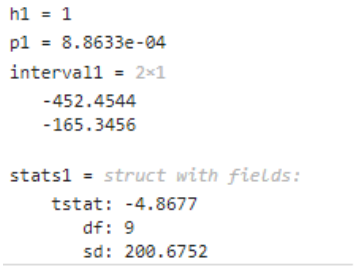


Figure 9: t-test Suma de Distàncies vs Entropia

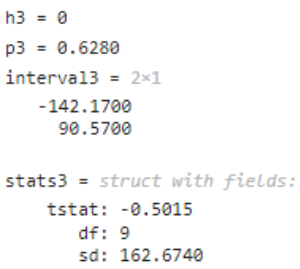


Figure 10: t-test Suma de Distàncies vs $\lambda = 1$

D'aquest experiment i veient les comparatives, ens hem de quedar amb la suma de distàncies o la lambda. No podem rebutjar la hipòtesi que són diferents, però sí que es decanta una mica més cap a la suma de distàncies. Per tant, concloem en què la suma de distàncies és el millor heurístic que optimitza el primer criteri.

4.6 Experiment 5: Simmulated Annealing

Fins ara hem experimentat utilitzant l'algorisme de Hill Climbing, però utilitzar l'algorisme de Simmuated Annealing podria donar-nos millors resultats en alguns casos.

L'algorisme de Simmulated Annealing inclou 4 paràmetres diferents. Aquests paràmetres influeixen notablement en el resultat de la solució definitiva. L'objectiu d'aquest experiment és trobar els millors valors per als paràmetres steps, k, λ i stiter.

Com a recordatori, el paràmetre steps indica el nombre d'iteracions de l'algorisme, els paràmetres k i λ controlen la temperatura (que indica la probabilitat d'acceptar solucions pitjors) i el paràmetre stiter indica cada quantes iteracions baixem la temperatura.

El plantejament de l'experiment és el següent:

Observació	· Poden existir paràmetres que poden obtenir una millor suma de distàncies · Pot influir tant en el temps d'execució com en la qualitat de la solució final.
Plantejament	Comparem en els següents experiments possibles paràmetres
Hipòtesi	· Els paràmetres poden influir en la solució

Table 9: Plantejament de l'experiment 5

Mètode:

- Seguirem una metodologia similar a la proposada en l'enunciat.
- Els primers paràmetres que ajustarem són la k i la λ, ja que els steps i el stiter dependran en gran part d'aquests dos paràmetres.
- Per provar els paràmetres k i lambda hauríem, primer, assegurar que el nombre d'iteracions és suficientment gran per assegurar-nos que l'algorisme convergeix a una solució. Per aquest motiu primer experimentarem amb diversos valors d'steps per veure a partir de quin punt el Simmulated Annealing aconsegueix resultats millors al Hill Climbing (encara que de moment no estem buscant l'òptim).

  Per la prova d'steps fixem $k = 20$, $\lambda = 0.005$ i $stiter = 100$, els paràmetres per defecte de l'algorisme.

- Un cop trobem un paràmetre pels steps prou bo començarem a experimentar amb valors de k i λ

  Per fer-ho simplement seleccionarem 10 llavors aleatòries i executarem l'algorisme amb $stiter = 100$, amb el valor d'steps seleccionat i anirem executant vaires vegades Simmulated Annealing variant els 2 paràmetres que volem ajustar.

  Quan trobem una bona combinació inicial de k i λ, farem un segon experiment (amb les mateixes 10 llavors) amb valors de k i λ més propers als òptims trobats previament.

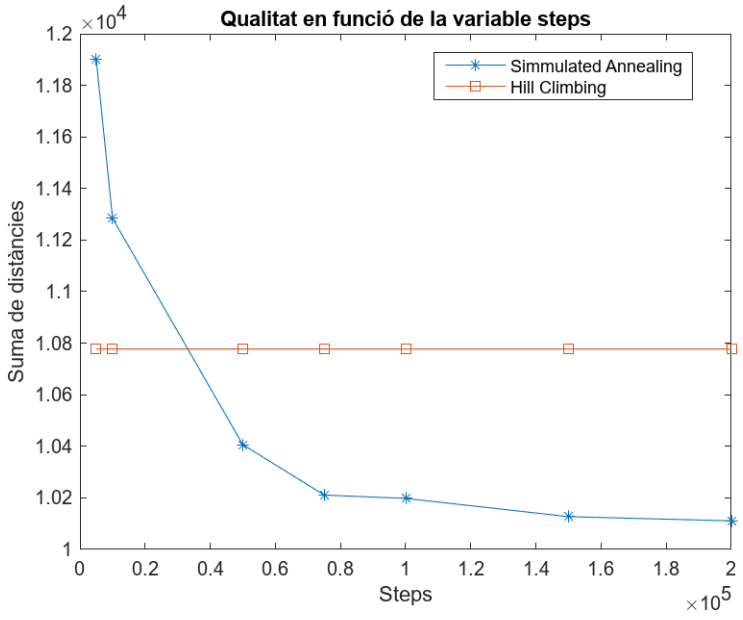
- Un cop tinguem bons valors de k i λ provarem diversos valors d'stiter. Per fer-ho seleccionarem 10 llavors aleatòries i executarem un cop cada llavor per diversos paràmetres d'stiter.

- Finalment, amb k, λ i stiter fixats, tornarem a provar quin és el millor nombre d'steps. Per fer-ho seleccionarem 10 llavors aleatòries i executarem l'algorisme de Simmulated Annealing un cop per cada llavor i mirarem com evoluciona la bondat de la solució en funció del nombre de steps.

4.6.1 Nombre de *Steps*

El primer experiment consisteix, com ja hem explicat abans, en seleccionar un nombre d'steps inicials per realitzar la resta de l'experimentació.

Després d'executar Simmulated Annealing per diverses llavors (i fer la mitjana) hem obtingut el següent gràfic:



Amb aquest primer gràfic podem observar que a partir de 40000 steps, Simmulated Annealing ja obté millors solucions que Hill Climbing. També

podem veure que al principi hi ha una gran millora en augmentar els steps, però que ràpidament es frena, indicant que amb un valor de steps entre 60000 i 80000 és suficient. Després d'aquest experiment hem decidit agafar el valor de 75000 steps per fer els següents experiments, ja que ens proporciona una bona qualitat de la solució i agafar un nombre major tampoc proporcionaria millores a l'hora d'experimentar (cal recordar que ara mateix no estem buscant l'òptim).

4.6.2 Experimentació amb k i λ

Aquest segon experiment té com a objectiu trobar una parella de valors k i λ que optimitzi la qualitat de la solució obtinguda amb el valor d'steps fixat en l'anterior apartat. Primer començarem provant combinacions de $k = \{1, 5, 25, 125\}$ i de $\lambda = \{1, 0.01, 0.0001\}$

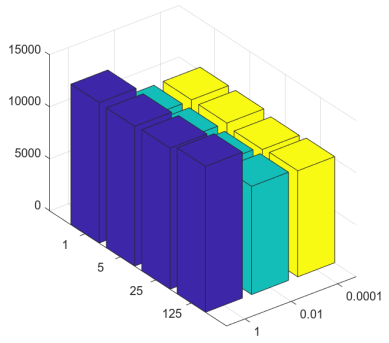


Figure 11: Figura amb els 3 valors de λ

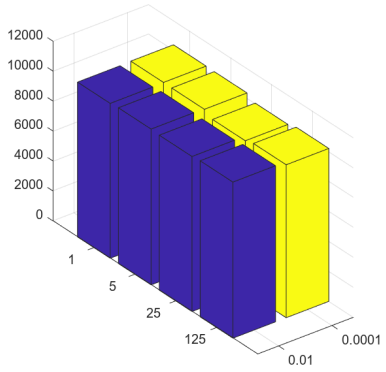
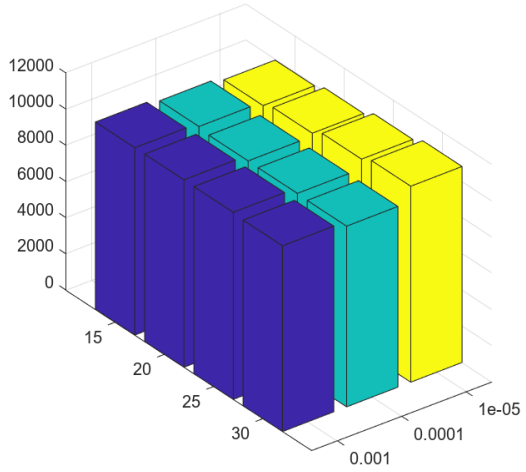


Figure 12: Figura sense $\lambda = 1$

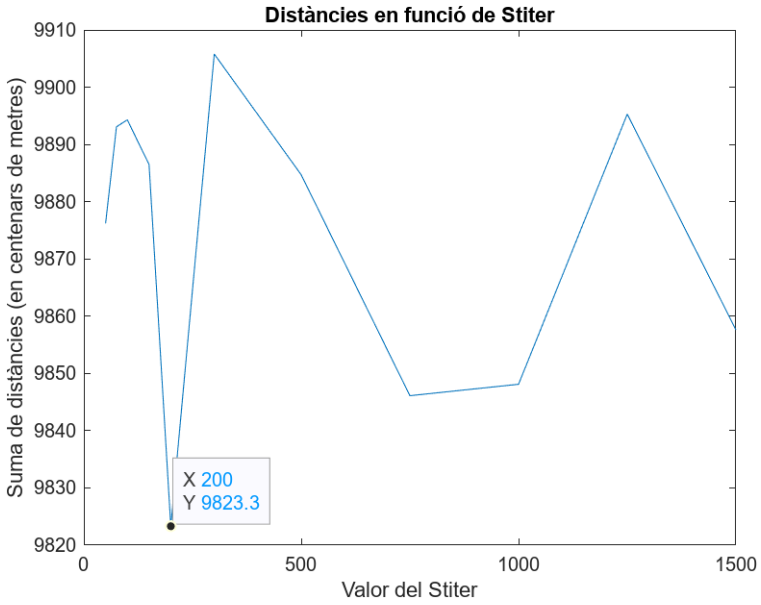
El primer que podem observar és que $\lambda = 1$ no és una bona opció. Després, amb la resta de valors el resultat és bastant semblant, tot i que es pot veure una millora amb $\lambda = 0.0001$ i k = 25.



Ara partint dels valors de k i λ obtinguts anteriorment, busquem uns valors més concrets. Trobem que els valors k = 15 $\lambda = 0.00001$ són algo millors, per tant, són els que utilitzarem per als següents experiments.

4.6.3 Experimentació amb *Stiter*

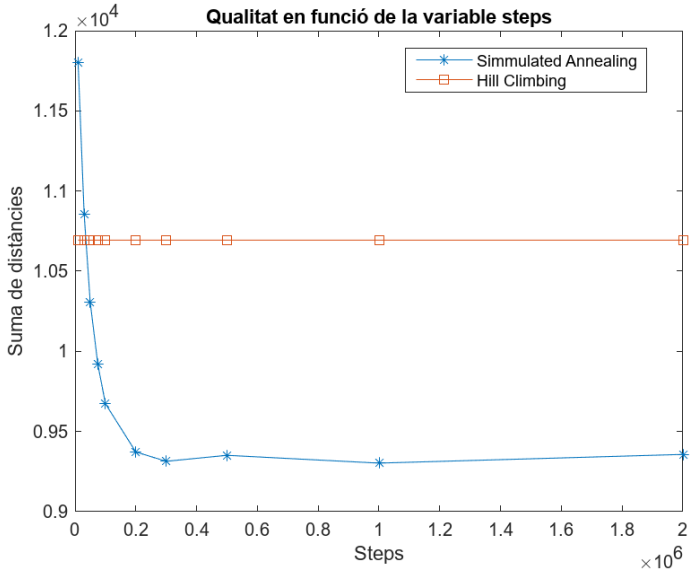
Després de fixar k i λ ara passem a experimentar amb el valor de Stiter. Després d'haber executaat l'experiment per a valors de $stiter = \{50, 75, 100, 150, 200, 250, 300, 500, 750, 1000, 1250, 1500\}$ (Tots divisors de 75000) Hem obtingut la seguent gràfica:



Aquí no veiem un patró clar ni cap tendència, però el valor steps = 200 és el que millor ens ha funcionat experimentalment, així que és el que seleccionarem per a futurs experiments.

4.6.4 Nombre de *Steps* amb k, λ i stiter fixats

Finalment, tornem a executar el primer experiment per, ara si, trobar l'òptim d'steps amb els altres paràmetres fixats. Els valors provats per al paràmetre steps han sigut: $\{10000, 30000, 50000, 75000, 100000, 200000, 300000, 500000, 1000000, 2000000\}$



Ara veiem un resultat similar al previ, on millora molt al principi, però es frena ràpidament. Veiem que entre 200000 i 400000 ja no millora massa (de fet empitjora lleugerament) per això hem decidit escollir el valor steps = 300000 com a valor definitiu.

4.7 Experiment 6: Augmentem la mida del problema

Per al següent experiment observarem com es comporta el temps d'execució de l'algorisme Hill Climbing. Començarem amb un valor de 200 usuaris i anirem augmentant de 100 en 100 fins a trobar un patró que ens indiqui a quin ritme creix. Amb totes les execucions mantindrem la mateixa proporció d'usuaris i conductors.

El plantejament de l'experiment és el següent:

Observació	Augmentar el nombre de persones a l'escenari també fa augmentar el temps
Plantejament	Augmentem el nombre de persones a l'escenari per veure com es comporta en temps
Hipòtesi	· Augmentar el nombre d'usuaris també té molta influència en el temps d'execució
Mètode	· Escollim 10 llavors aleatòries · Per a cada llavor executem Hill Climbing i mesurem temps d'execució · Considerem una relació d'1 cotxe per cada 2 persones

Table 10: Plantejament de l'experiment 6

Després d'executar l'experiment hem pogut obtenir els següents resultats:

Usuaris	200	300	400	500	600	700
Temps (segons)	9'1471	50'2545	193'8242	481'5238	1038'2206	2263'407

Table 11: Mitjanes de l'experiment 6

Després de recopilar les dades, ens hem plantejat buscar diferents models polinòmics que s'ajustin a les nostres dades específiques.

A continuació mostrem els 4 models que hem provat: Amb polinomis de grau 2, 3 i 4 i amb una corba exponencial.

L'aproximació que més s'ajusta és la de grau 4, però com més augmentem el grau del polinomi més fàcil serà forçar que passi (o s'apropi) per tots els punts. A més el coeficient de grau 4 de l'aproximació ens ha sortit tan petit que creiem que el nostre algorisme de Hill climbing té un comportament cúbic.

Els errors al quadrat obtinguts de les diferents aproximacions són els següents:

Aproximació	Grau 2	Grau 3	Grau 4	Exponencial
Errors al quadrat	6.1425e+10	4.6299e+09	1.3751e+08	6.3967e+12

Table 12: Suma d'errors al quadrat per diferents models d'aproximació

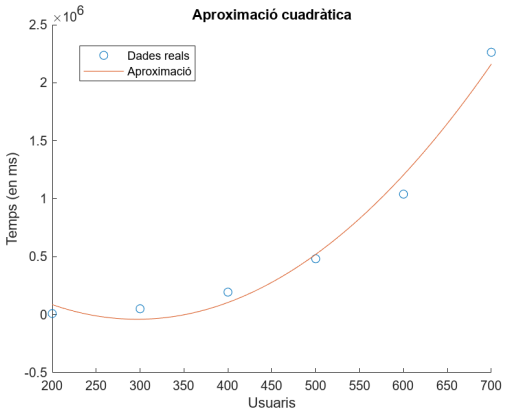


Figure 13: Aproximació quadràtica

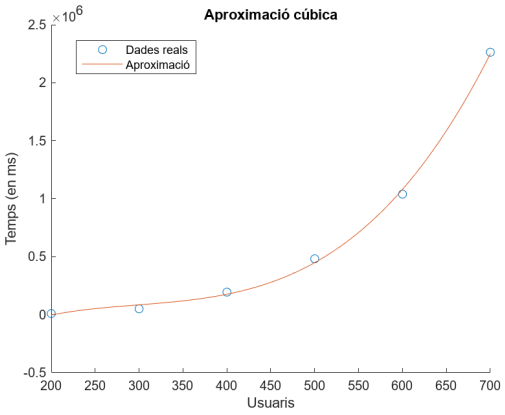


Figure 14: Aproximació cubica

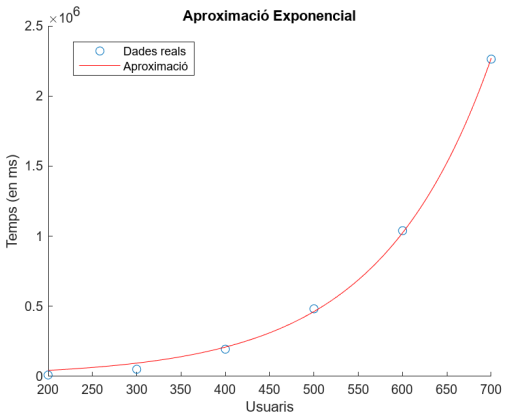


Figure 15: Aproximació exponencial

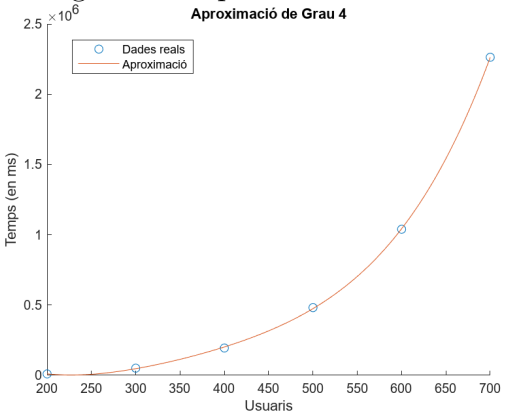


Figure 16: Aproximació amb polinomi de grau 4

L'algorisme de Hill Climbing genera r successors a cada iteració, selecciona la millor opció i després continua la cerca des de la nova solució, per tant, podríem considerar que té un cost de $O(rp)$ on p és la profunditat fins a on arriba i r és el factor de ramificació.

Veient-ho d'aquesta manera, té sentit que l'aproximació exponencial sigui la pitjor. El nostre factor de ramificació és la suma dels 3 factors de ramificació quadràtics, així que l'algorsime ha de tenir un creixement, com a mínim quadràtic. Experimentalment sembla que podria comportar un creixement tant cúbic com de grau 4.

4.8 Experiment 7: Nou criteri de qualitat

En següent experiment canviarem de criteri de qualitat. Ja no mirarem només que la distància recorreguda sinó que també mirarem el nombre de cotxes utilitzats.

Recordatori que ens referirem als heurístics com:

- h_1 = Suma de distàncies
- h_4 = Distàncies per nombre cotxes
- h_5 = Distàncies per (nombre de cotxes)^2

El plantejament de l'experiment és el següent:

Observació	Tenint en compte el nombre de cotxes en l'heurístic es veu una reducció dels cotxes utilitzats
Plantejament	Utilitzem diferents heurístics que tinguin en compte (o no) el nombre de cotxes de diferent manera
Hipòtesi	· Els heurístics que tinguin en compte el nombre de cotxes són millors · També tarden més a ser calculats.
Mètode	· Escollim 10 llavors aleatòries · Per a cada llavor executem Hill Climbing amb les tres heurístiques i mesurem temps, distància total i nombre de cotxes · Considerem sempre 100 conductors i 200 usuaris

Table 13: Plantejament de l'experiment 7

Després d'executar l'experiment hem pogut obtenir els següents resultats:

Heurístic	h_1	h_4	h_5
Suma de distàncies (Centenars de m)	10904,5	10544	10508,8
Temps (segons)	10,4921	24,2249	24,780
Nombre de cotxes	90,6	55,1	54,7

Table 14: Mitjanes de l'experiment 7

El primer que podem observar és que els heurístics funcionen correctament i hem aconseguit reduir quasi a la meitat el nombre de cotxes utilitzats. No només això, sinó que també hem reduït la distància recorreguda. També veiem que la diferència entre els heurístics h_4 i h_5 és poca.

Hem realitzat diversos t-tests per a comprovar realment les diferències entre els heurístics, però no els incloem aquí pel nombre tan elevat que hem realitzat. Si el lector desitja accedir als resultats, els pot trobar o bé a la carpeta de *graphics* o bé a l'script de MATLAB.

Igualment, la conclusió dels t-tests ha sigut que l'heurístic h_1 té millor rendiment en l'àmbit temporal que els altres 2, però aconsegueix resultats pitjors en suma de distàncies i nombre de cotxes. Per als altres 2 heurístics no podem confirmar ni descartar que un sigui millor que l'altre.

4.9 Experiment 8: Nous criteris amb Simmulated Annealing

En aquest experiment repetirem la mateixa comparació que en l'apartat anterior però utilitzant Simmulated Annealing.

El plantejament de l'experiment és el següent:

Observació	Els heurístics es poden comportar de forma diferent en SA respecte a Hill Climbing
Plantejament	Utilitzem els 3 mateixos heurístics que en l'apartat anterior i comparem a veure quin es comporta millor
Hipòtesi	· Els resultats obtinguts seran similars a l'experiment anterior
Mètode	· Escollim 10 llavors aleatòries · Per a cada llavor executem SA i mesurem temps, qualitat i nombre de cotxes

Table 15: Plantejament de l'experiment 8

Després d'executar l'experiment hem pogut obtenir els següents resultats:

Heurístic	h_1	h_4	h_5
Suma de distàncies (Centenars de m)	9419,3	10653,7	10523,4
Temps (s)	1'6948	2'054	1'9939
Nombre de cotxes	59,1	50,1	50,2

Table 16: Mitjanes de l'experiment 8

A diferència de l'execució amb Hill Climbing ara h_1 aconsegueix millor resultats en la suma de distàncies i molts bons resultats en la quantitat de cotxes. També continua sent més eficient en termes temporals que els altres heurístics. En canvi, els heurístics h_2 i h_3 continuen sent bastant similars, com passava amb Hill Climbing.

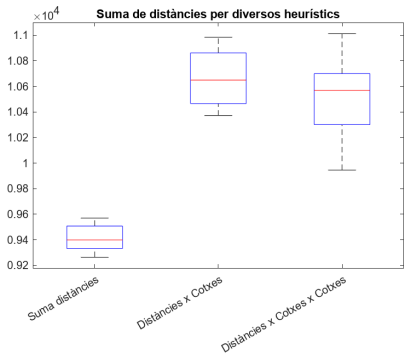


Figure 17: Boxplot sobre la suma de distàncies

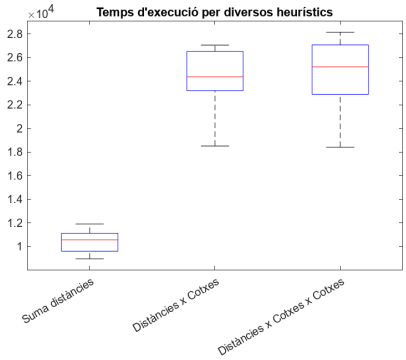


Figure 18: Boxplot sobre el temps total

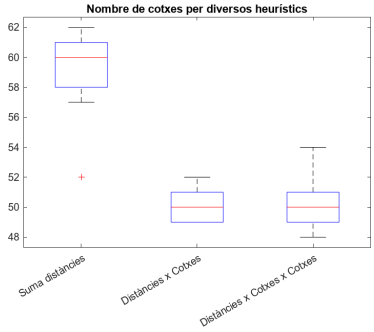


Figure 19: Boxplot sobre el nombre de cotxes

En conclusió, veient tots els resultats i gràfics, Simmulated Annealing

funciona molt bé amb h_1, tant en distància recorreguda, com amb temps i cotxes. Podem dir que aquesta combinació és la que millors resultats obté.

4.10 Experiment 9: Solucions inicials més restrictives

Perl l'últim experiment de tots, hem provat si, reduint el nombre inicial d'usuaris, podem accelerar la convergència del nostre algorisme.

El plantejament de l'experiment és el següent:

Observació	En baixar el nombre de cotxes inicial es pot obtenir una convergència més ràpida cap a la solució final.
Plantejament	Reduïm el nombre de conductors per veure si la solució convergeix abans
Hipòtesi	· Reduir el nombre de conductors es pot obtenir una solució inicial amb menys temps i fins i tot amb més bondat.
Mètode	· Primer mirem quina és la proporció aproximada necessària per a 10 llavors aleatòries amb Simmulated Annealing · Seleccionem 10 noves llavors aleatòries i comparem els resultats que ens dona SA per a la proporció original i per a la més restrictiva. Mesurem temps, qualitat i nombre de cotxes. · Considerem una relació d'1 conductor per cada 2 persones de forma inicial · Provem diverses quantitats d'usuaris · Els paràmetres de SA utilitzats han sigut steps = 300 000, stiter = 200, k = 15 i $\lambda = 0.00001$ · L'heurístic utilitzat ha sigut h_5

Table 17: Plantejament de l'experiment 9

Hem provat diferents quantitats d'usuaris amb les 10 llavors aleatòries. En concret hem provat quantitats des de 200 fins a 650 augmentant de 50 en 50 les quantitats. Els resultats amb les ràtios obtinguts han sigut els següents:

	200	250	300	350	400
Nombre de cotxes	48,4	61,3	73,2	85,4	97
Usuaris/Cotxes	4,132231405	4,078303426	4,098360656	4,098360656	4,12371134

Table 18: Mitjanes de l'experiment 9

	450	500	550	600	650
Nombre de cotxes	107,2	120,3	131,6	142,9	155,2
Usuaris/Cotxes	4,197761194	4,156275977	4,179331307	4,198740378	4,18814433

Table 19: Mitjanes de l'experiment 9

A continuació es poden veure els resultats en una gràfica, es pot apreciar que la ràtio entre persones i usuaris es manté entre 4,1 i 4,2 usuaris per cotxe.

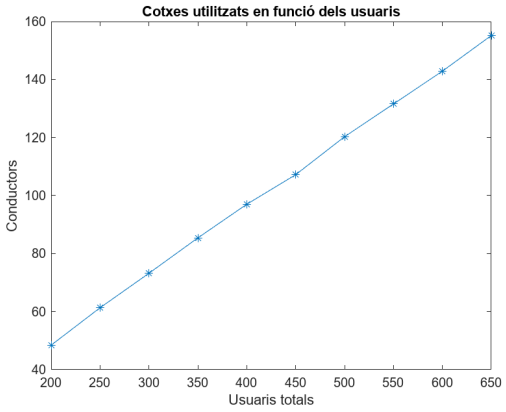


Figure 20: Gràfic sobre els conductors utilitzats per usuaris

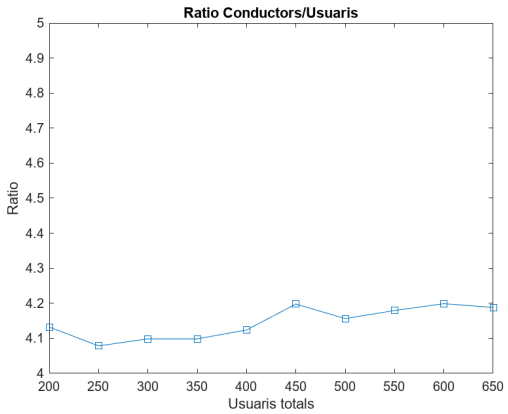


Figure 21: Gràfic sobre la ràtio dels conductors utilitzats

A partir d'aquest hem intentat baixar la proporció inicial a un terç en lloc d'un mig. Però ens hem trobat amb un problema: La nostra solució inicial no pot generar (en la majoria dels casos) solucions vàlides per a proporcions tan baixes.

Per aquest motiu hem "relaxat" una mica la proporció, i hem vist que utilitzant una proporció de 5/14 (simplement hem intentat pujar la relació d'un terç), l'algorisme ens proporciona una solució vàlida la gran majoria de vegades, així que hem decidit fer els experiments amb aquesta proporció.

Gràcies a aquest problema ens hem adonat d'un problema que té la nostra solució inicial: No sempre dona solucions vàlides. Fins ara no havíem tingut cap problema, ja que utilitzant la relació d'un mig falla amb un percentatge molt baix de les llavors (tan baix que de fet no hem tingut problemes amb

els anteriors experiments).

Aquesta característica de la solució inicial s'haurà de tenir força en compte si volem ajustar la proporció d'usuaris i conductors.

De totes maneres, hem aconseguit seguir l'experiment comparant la proporció original d'un mig amb la nova ràtio de 5/14:

Distància total (centenars de metres)	10335.1
Nombre de cotxes	47.8
Temps(s)	1.9117

Table 20: Mitjanes de l'experiment amb proporció 5/14

Distància total (centenars de metres)	10405.4
Nombre de cotxes	49.4
Temps(s)	1.9263

Table 21: Mitjanes de l'experiment amb proporció 1/2

Amb aquestes dades també vam obtenir els següents boxplots i vam realitzar els següents t-tests:

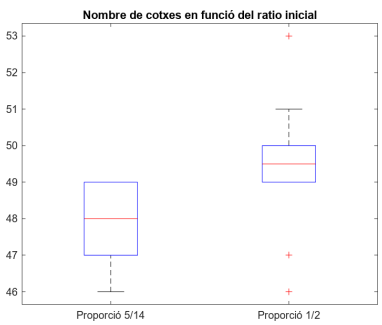


Figure 22: BoxPlot sobre els cotxes

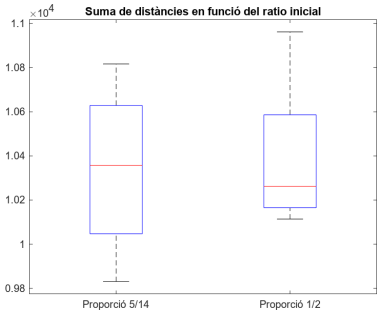


Figure 23: BoxPlot sobre les distàncies

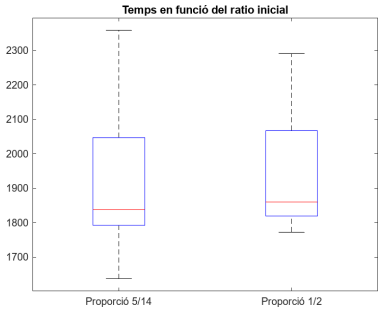


Figure 24: BoxPlot sobre el ràtio

```
ht =
     0
pt =
   0.859792271427086
intervt = 2×1
10^2 ×
    -1.963007167024440
     1.671007167024441

statst = struct with fields:
    tstat: -0.181768653290125
       df: 9
       sd: 2.540000874890488e+02
```

Figure 25: t-test sobre el temps

```
hd =
     0
pd =
   0.581068262612437
intervd = 2×1
10^2 ×
    -3.481314285054730
     2.075314285054729

statsd = struct with fields:
    tstat: -0.572396180663129
       df: 9
       sd: 3.883815563763724e+02
```

Figure 26: t-test sobre la distància

```
hc =
     1
pc =
   0.032990725306176
intervc = 2×1
    -3.038640264906229
    -0.161359735093771

statsc = struct with fields:
    tstat: -2.515883608132604
       df: 9
       sd: 2.011080417199781
```

Figure 27: t-test sobre els cotxes

En els tests i els boxplots es pot apreciar com no podem fer cap afirmació sobre el temps que triga cada ràtio ni sobre la bondat de la solució que generen, ja que ens donen p-valors de 0.86 i 0.58. L'únic que podem afirmar és que la ràtio de 5/14 sembla minimitzar millor el nombre de cotxes.

En un principi ens semblava que iniciar amb una solució inicial més restrictiva podria fer que la solució convergís més ràpidament i, fins i tot, que produís millors resultats, però sembla que la diferència real és molt petita.

Igualment, i encara que la proporció més restrictiva sembli ser lleugerament superior (encara que caldrien més experiments per confirmar-ho del tot), donada la similitud entre les dues proporcions proposades, nosaltres escolliríem la relació d'un mig, ja que ens assegura poder trobar una solució inicial vàlida amb probabilitat molt més alta que una proporció més restrictiva.

Simmulated Annealing al augmentar la mida del problema

Finalment, en executar aquest algorisme amb Simmulated Annealing teníem la sospita que potser el nombre d'steps fixat en l'experiment 5 no seria suficient per nombres majors d'usuaris.

Com que hem anat augmentant el nombre d'usuaris amb l'algorisme de Simmulated Annealing, hem aprofitat per veure com es comporta aquest algorisme en augmentar la mida del problema.

Com es pot apreciar en les següents gràfiques, la relació entre el nombre d'usuaris i la distància total va reduint en augmentar el nombre d'usuaris. Aquesta comprovació ens indica que, probablement, el nombre d'steps, encara que pugi no ser òptim per a mides més grans, no provoca un problema de convergència. Si haguéssim vist que la ràtio augmentava en lloc de disminuir ens hauríem plantejat augmentar el nombre d'iteracions.

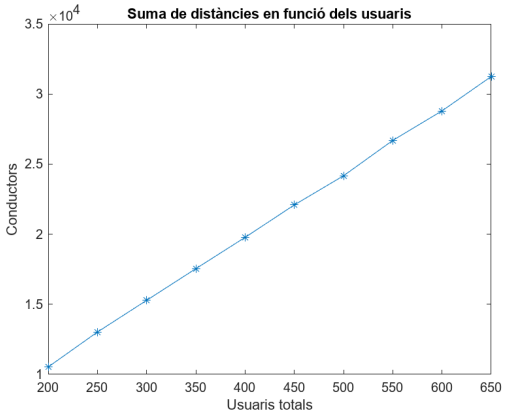


Figure 28: Gràfic sobre les distàncies en funció dels usuaris

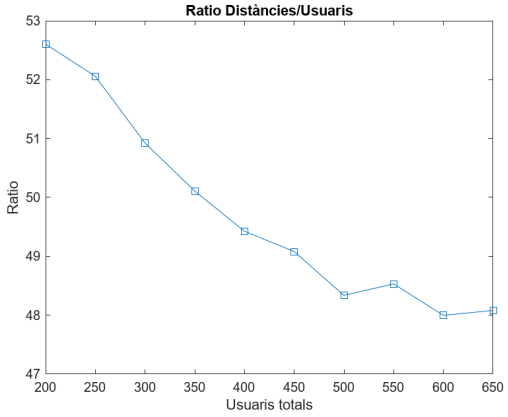


Figure 29: Gràfic sobre el ràtio de distàncies

Finalment, hem aprofitat per comparar el temps d'execució entre Hill Climbing i Simmulated Annealing (Encara que siguin amb llavors diferents). Com que el nombre d'iteracions no varia, el temps d'execució només depèn del cost temporal de trobar la solució inicial, d'aplicar els operadors i de comprovar que la solució final és vàlida (hem decidit fer la comprovació en el cas que la solució inicial falli).

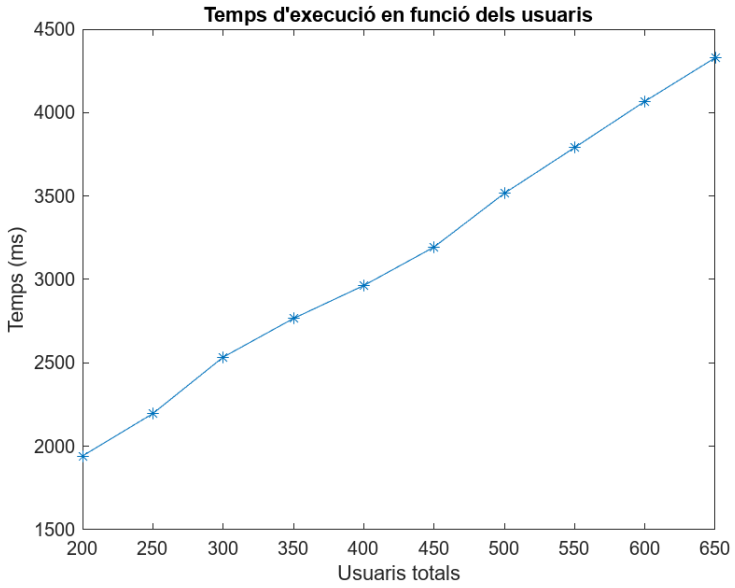


Figure 30: Plot sobre el temps d'execució en funció del temps

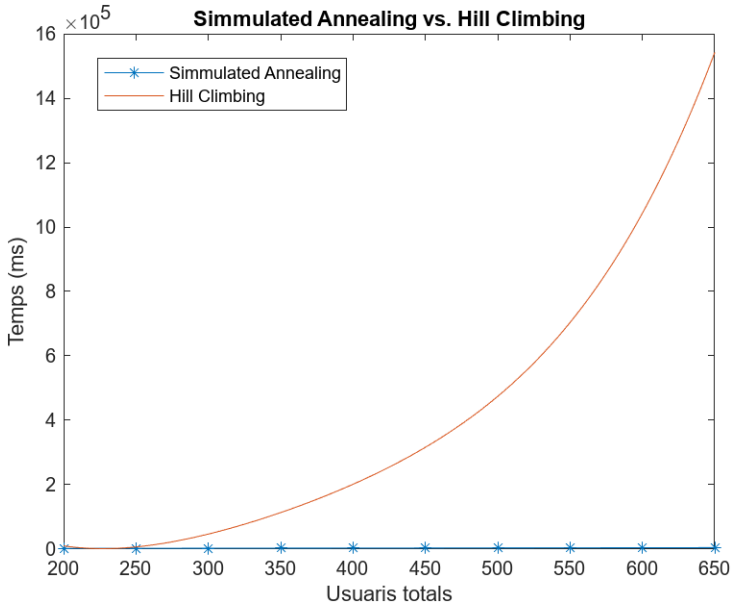


Figure 31: HC vs SA

Encara que no sigui una comparació del tot rigorosa, hem decidit veure com es comportava el temps d'execució del Simmulated Annealing en contra del model de grau 4 que hem trobat pel Hill Climbing (que podríem considerar com una aproximació). En les gràfiques anteriors ja es pot apreciar que el Simmulated Annealing té un rendiment increïblement superior el Hill Climbing en augmentar el nombre d'usuaris.

5 Conclusions

Utilitzar cerca local en aquest problema ha sigut una idea molt encertada, al final aquest problema té una forta relació amb el TSP i no tenim bons algorismes per solucionar aquest. També aquest problema, igual que el TSP, no té un heurístic clar per resoldre amb algorismes com A* el qual et podria trobar un òptim si tinguéssim un molt bon heurístic.

Sobre la comparació entre SA i HC ens ha sorprès molt tant la qualitat com el temps que triga a executar-se SA enfront de HC. SA dona unes millors solucions de manera experimental que HC tant en temps, com en distància total com a nombre de cotxes. També hem vist que l'heurístic de suma de distàncies millora molt en SA convertint-se en un dels millors per aquest. Sí que és cert que igualment els dos millors heurístics que tenim donen igual de rendiment tant en HC com a SA.

Pel que a les funcions heurístiques fa, disposant de tal varietat de funcions hem arribat a diverses conclusions: Primerament, el primer heurístic h_1[1] i el tercer h_3 donen resultats molt similars i no podem confirmar que cap sigui millor que l'altre. També cal recalcar que h_2 ha donat pitjors resultats tal com esperàvem, pel que a distàncies es refereix. Pel que fa a l'optimització del nombre de cotxes, tenim els heurístics h_4 i h_5, que sorprenentment, no semblen oferir cap diferència notòria, llavors estadísticament tampoc podem confirmar que cap de les dues sigui millor que l'altre. A la pràctica hem vist que h_4 i h_5 donen molt bons resultats tant en HC com a SA. Sí que és cert que h_1 en SA dona molt millor resultat quant a distància a canvi d'utilitzar no més de 3 cotxes.

Contra més temps passava i més experimentàvem més ens adonàvem que els heurístics no eren massa bons alguns i que altres no els havíem pensat bé de partida directament, com el h_3 per exemple. Però a pesar d'això volíem provar i experimentar-los. Ambdós h_4 i h_5 serien millors si en comptes de tenir les seves variables multiplicant-se entre si estiguessin sumades, ja que podríem afegir a cadascuna d'elles uns factors constants ε_i per jugar i experimentar amb la importància de les dues variables, ja que potser seria interessant donar valors majors a 1 al factor dels cotxes (Com és evident les unitats

[1]Recordar que ens referim als heurístics per el seu nom de l'apartat 3.4

de les variables no són les mateixes, llavors la multiplicació no té gaire sentit).

Hem sigut sorpresos del fet que la solució *greedy* dona una millor qualitat a la solució final que la solució generada de forma aleatòria. Al final com totes les eleccions es fan de forma aleatòria possiblement serà pitjor i estigui més lluny d'un màxim local.

Sobre la solució inicial també cal dir que, experimentant, hem vist que a partir d'1 conductor cada 3 passatgers no sempre genera una solució correcta. Això es deu al fet que probablement l'assumpció de repartició equitativa no sempre és l'encertada. El que ens diu que, si haguéssim de tornar a fer aquest experiment, escolliríem una estratègia alternativa per generar aquestes.

Finalment, cal destacar que, si volguéssim fer una extensió del problema, no tindríem un gran problema (si no canviés l'essència d'aquest).