UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

PRÀCTICA DE CERCA LOCAL

ALGORISMES BÀSICS PER LA INTEL·LIGÈNCIA ARTIFICIAL

LLUC FURRIOLS
MIQUEL ROPERO
POL MARGARIT

1. EL PROBLEMA..... 3
1. 1. ELEMENTS DEL PROBLEMA..... 4
1. 2. SOLUCIÓ DEL PROBLEMA..... 5
1. 3. PROBLEMA DE CERCA LOCAL..... 6
2. IMPLEMENTACIÓ DE L'ESTAT..... 7
2. 1. ESTRUCTURA DE DADES DE L'ESTAT..... 7
2. 2. REPRESENTACIÓ DEL PROBLEMA..... 7
2. 3. ANÀLISIS TAMANY DE L'ESPAI DE BÚSQUEDA..... 9
3. JUSTIFICACIÓ DELS OPERADORS ESCOLLITS..... 10
4. ESTRATÈGIES PER TROBAR L'ESTAT INICIAL..... 12
4. 1. GENERACIÓ ESTAT INICIAL NO 'GREEDY'..... 12
4. 1. 1. JUSTIFICACIÓ DE L'ESTAT INICIAL NO 'GREEDY'..... 13
4. 2. GENERACIÓ DE L'ESTAT INICIAL 'GREEDY'..... 13
4. 2. 1. JUSTIFICACIÓ DE L'ESTAT INICIAL 'GREEDY'..... 14
5. FUNCIONS HEURÍSTIQUES..... 15
5. 1. FACTORS QUE INTERVENEN EN LA HEURÍSTICA DEL PROBLEMA..... 15
5. 2. FUNCIONS HEURÍSTIQUES ESCOLLIDES..... 16
5. 2. 1. FUNCIÓ HEURÍSTICA 1..... 16
5. 2. 2. FUNCIÓ HEURÍSTICA 2..... 18
5. 3. EFECTES DE LES FUNCIONS HEURÍSTIQUES EN LA CERCA..... 19
5. 4. JUSTIFICACIÓ DE LA PONDERACIÓ..... 19
6. EXPERIMENTS..... 20
6. 1. EXPERIMENT 1..... 20
6. 2. EXPERIMENT 2..... 23
6. 3. EXPERIMENT 3..... 26
6. 4. EXPERIMENT 4..... 34
6. 5. EXPERIMENT 5..... 36
6. 6. EXPERIMENT 6..... 39
6. 7. EXPERIMENT ESPECIAL..... 41

1. EL PROBLEMA

El problema que se'ns presenta és un problema que afecta els serveis urbans de préstec de bicicletes, com el sistema de Bicing. El repte principal és la distribució òptima de les bicicletes a les estacions per assegurar que els usuaris puguin trobar una bicicleta disponible quan la necessitin. Per abordar aquest problema, Bicing ha recopilat estadístiques sobre la demanda i els moviments de les bicicletes en les seves estacions a cada hora del dia.

La informació que té Bicing inclou:

- Previsió de bicicletes no utilitzades: Aquesta informació indica quantes bicicletes es preveu que no siguin utilitzades en una estació durant una hora específica i que podran ser traslladades a una altra estació. Això implica identificar les estacions on hi ha un excedent de bicicletes que podrien ser necessàries en altres llocs.

- Previsió de bicicletes per a l'hora següent: Aquesta previsió estima quantes bicicletes hi haurà en una estació l'hora següent a l'actual. Aquesta dada només té en compte els canvis en la quantitat de bicicletes a una estació causats pels usuaris (és a dir, no inclou bicicletes que puguin ser traslladades entre estacions per altres mitjans).

- Previsió de bicicletes necessàries per cobrir la demanda: Aquesta previsió estima quantes bicicletes hauria de tenir una estació l'hora següent per satisfer la demanda prevista. És a dir, s'identifiquen les estacions on la demanda prevista pot superar el nombre total de bicicletes disponibles.

L'objectiu principal és optimitzar la distribució de les bicicletes entre les estacions en funció d'aquestes previsions, de manera que les bicicletes es traslladin de manera eficient per garantir que les estacions estiguin ben equipades i que els usuaris puguin accedir a les bicicletes quan les necessitin. Aquest problema implica tenir en compte la disponibilitat actual, les previsions de demanda i les rutes òptimes per traslladar les bicicletes entre les estacions.

1. 1. ELEMENTS DEL PROBLEMA

Aquesta descripció detallada del problema estableix els elements clau, les restriccions i els objectius del problema plantejat. La tasca principal és trobar una distribució de bicicletes òptima per satisfer la demanda prevista i minimitzar els costos associats al transport.

- Ciutat on es genera el problema: La ciutat es considera un quadrat de 10x10 quilòmetres, amb carrers que formen una quadrícula on cada illa mesura 100x100 metres. Les estacions de Bicing es troben als encreuaments de carrers.

- Distància: La distància entre dos punts de la ciutat es calcula utilitzant la funció d(i, j), que mesura la distància en metres entre dos punts a la quadrícula, utilitzant les seves coordenades x i y.

- Estacions de Bicing: Hi ha E estacions de bicicletes repartides per la ciutat, i en total hi ha B bicicletes, que estan distribuïdes entre totes les estacions. El nombre total de bicicletes és constant i no es poden afegir ni treure bicicletes del sistema. S'assumeix que el nombre de bicicletes que poden cabre en una estació és il·limitat.

- Demanda de Bicicletes: La demanda de bicicletes en cada estació pot variar i és diferent d'una estació a una altra.

- Flota de Furgonetes: Hi ha una flota de F furgonetes disponibles per moure les bicicletes d'una estació a una altra. Cada furgoneta pot transportar com a màxim 30 bicicletes i només pot fer un viatge per hora. Cada furgoneta pot transportar bicicletes d'una estació a, com a màxim, 2 estacions en un sol viatge i no pot haver-hi dues furgonetes que prenguin bicicletes de la mateixa estació. El nombre de bicicletes que poden cabre en una estació no està limitat.

- Acord Econòmic: Bicing paga 1 euro per cada bicicleta que es transporti i que faci que el nombre de bicicletes en una estació s'aproximi a la demanda prevista. Es paga per les bicicletes addicionals que hi ha en una estació en comparació amb la previsió, sempre que no superin la demanda prevista. No obstant això, es cobra 1 euro per cada bicicleta que es mogui i que allunyi una estació de la seva previsió.

- Cost del Transport: El cost del transport de les bicicletes depèn del nombre de bicicletes transportades en una furgoneta. El cost en euros per quilòmetre recorregut és ((nb + 9) div 10), on div és la divisió entera i nb el nombre de bicicletes que transporta la furgoneta.

1. 2. SOLUCIÓ DEL PROBLEMA

La solució ha d'indicar els trasllats de les bicicletes que realitzaran les furgonetes, incloent-hi l'origen de la furgoneta, les destinacions (com a màxim dues) i l'ordre en què les visitarà. També ha d'indicar quantes bicicletes es deixen a cada destinació.

El problema que s'ha de resoldre és l'optimització del que obtenim per trasllat de bicicletes i la minimització dels costos de trasllat de les furgonetes.

Criteris i Restriccions:

1. Capacitat de les Furgonetes: El nombre de bicicletes que es carreguen en una furgoneta no pot superar el límit de 30. Això garanteix que es compleixi la capacitat de les furgonetes.

2. Visita a Estacions: Les furgonetes no poden visitar més de dues estacions en un sol viatge. Aquesta restricció limita les furgonetes a efectuar com a màxim dues entregues de bicicletes en un únic viatge. Això podria implicar una recollida i una entrega, dues entregues o només una recollida.

3. Càrrega de Bicicletes a l'Estació d'Origen: Les furgonetes només poden carregar bicicletes a l'estació d'origen.

4. Deixar Bicicletes en la mateixa Estació: Més d'una furgoneta pot deixar bicicletes en la mateixa estació. Això permet una flexibilitat en la distribució de bicicletes, ja que no totes les bicicletes s'han de moure a estacions diferents. Pot ser útil per equilibrar la disponibilitat en estacions amb una gran demanda.

Criteris d'Avaluació:

1. Maximització del Benefici Obtingut pels Trasllats de Bicicletes: Aquest criteri es refereix a l'objectiu de maximitzar els ingressos obtinguts per Bicing pels trasllats de les bicicletes. Això implica assegurar que les bicicletes es mouen de manera que s'aproximin a la demanda prevista, la qual cosa permet guanyar ingressos. Les furgonetes han de transportar bicicletes que s'ajustin a aquesta demanda per assolir pagaments positius.

2. Minimització dels Costos de Transport de les Bicicletes: Aquest criteri es centra en la minimització dels costos associats al transport de les bicicletes. Això implica optimitzar les rutes de les furgonetes per reduir la distància recorreguda i, per tant, minimitzar els costos de transport. És important assegurar que els costos no superin els ingressos obtinguts pels trasllats de les bicicletes.

Aquests criteris i restriccions guiaran la cerca d'una solució que sigui tan eficient com sigui possible, assegurant al mateix temps que es compleixin totes les restriccions específiques establertes per al problema. La combinació d'aquests dos criteris ha de permetre equilibrar els ingressos i les despeses per assegurar que la solució sigui econòmicament viable.

1. 3. PROBLEMA DE CERCA LOCAL

El conjunt del problema es pot considerar un problema de cerca local perquè implica l'exploració d'un espai d'estats, ja que les diferents configuracions possibles de distribució de bicicletes entre les estacions són un possible estat. Cada estat representa una possible distribució de bicicletes en un moment concret.

Implica l'avaluació de l'eficàcia de les distribucions mesurant la qualitat d'una distribució de bicicletes en termes d'ingressos obtinguts i costos de transport.

La cerca de moviments locals corresponen a les accions que es poden prendre per canviar d'un estat a un altre. En aquest cas, els operadors de moviment podrien ser les accions de carregar bicicletes en una estació d'origen i deixar-les en una o dues estacions de destinació, i l'ordre en què es realitzen aquests moviments.

L'estat inicial representa la distribució de bicicletes en un moment determinat. L'estat final desitjat és aquell que optimitza els ingressos i minimitza els costos.

L'objectiu no és trobar la millor solució global, sinó trobar una millora de la solució actual en la distribució de bicicletes que pugui augmentar els ingressos i minimitzar els costos, tot respectant les restriccions imposades. Això s'ajusta al paradigma de cerca local, que es centra a trobar una solució òptima dins d'una regió limitada de l'espai d'estats.

2. IMPLEMENTACIÓ DE L'ESTAT

2. 1. ESTRUCTURA DE DADES DE L'ESTAT

Inicialment, ens hem plantejat diverses solucions per representar l'estructura de dades de l'estat. Primerament, vam pensar a crear un diccionari on cada clau feia referència a una furgoneta la qual tenia una llista amb les estacions que visitava. També ens vam plantejar fer una classe per les furgonetes que tingués els atributs necessaris per a cada furgoneta. Tot i això, finalment ens hem decantat per representar l'estat utilitzant només llistes, perquè les llistes són estructures de dades eficients per a la majoria de les operacions bàsiques, com l'accés als elements, la cerca, l'afegit o l'eliminació d'elements. A més, són una de les estructures de dades més senzilles i intuïtives en Python. Cada element de la llista pot ser un altre objecte o una altra llista, cosa que permet una estructura de dades jeràrquica i de fàcil comprensió. Les llistes s'accedeixen mitjançant índexs, això fa que sigui senzill accedir als elements individuals d'aquesta.

La nostra implementació és una llista que conté per cada posició l'identificador de la furgoneta. Cada posició, és a dir, cada furgoneta és una llista que conté en la primera posició una altra llista que fa referència a l'identificador de l'estació origen i quantes bicicletes agafa d'aquesta estació; la segona posició és una llista que conté una altra llista per cada estació destí que visita, on també inclou quantes bicicletes deixa en l'estació destí; finalment l'última posició de la llista per cada furgoneta és un número que indica el benefici obtingut en total per la furgoneta.

2. 2. REPRESENTACIÓ DEL PROBLEMA

La representació del problema es basa en la definició d'un estat, que és gestionat a través de la classe `Estat`.

Aquesta classe representa l'estat del sistema i les dades associades. Té un atribut pels paràmetres del problema, un atribut per les furgonetes, un altre per les estacions d'origen visitades i un últim per les bicicletes que hi ha a cada estació. A continuació els detallem:

- `params`: Aquest atribut emmagatzema els paràmetres del problema, com ara la capacitat de les furgonetes, la llista d'estacions que conté totes les instàncies d'Estació i les furgonetes que es poden utilitzar. És una instància de `ProblemParameters` que permet accedir als paràmetres del problema.

- `furgonetes_util`: Aquesta llista emmagatzema les furgonetes i les seves dades associades. Cada furgoneta és una llista que inclou la seva estació d'origen, la llista de destins i el benefici obtingut per la furgoneta. La llista `furgonetes_util` conté les dades de totes les furgonetes.

- `estacions_origen`: Aquesta llista conté les estacions d'origen assignades a les furgonetes. Cada furgoneta té una estació d'origen que es registra en aquesta llista.

- 'bicis_en_estacions': Aquest atribut no s'inicialitza en la creació de l'estat, es crea quan apliquem una nova acció sobre un estat. És una llista que conté per cada estació una llista amb les bicis que no es mouran, les bicicletes que hi haurà en la següent hora i la demanda.

L'atribut `params` conté informació rellevant sobre els paràmetres del problema que l'estat ha de gestionar. L'estat utilitza els paràmetres del problema per saber quina és la capacitat de les furgonetes o la ubicació de les estacions per calcular el cost associat a les accions i l'estat actual.
En la generació de les accions i dels nous estats successors es necessita aquest atribut per recórrer totes les estacions.
L'atribut `params`, per tant, és necessari perquè permet a l'estat tenir accés als paràmetres del problema i fer que les seves operacions i decisions siguin coherents amb les restriccions i les necessitats específiques del problema que es vol resoldre.

L'atribut `furgonetes_util` és necessari per fer un seguiment detallat de les furgonetes, assignar destinacions, gestionar les bicicletes transportades i calcular el benefici.
Aquest atribut emmagatzema una llista de furgonetes que s'utilitzen per a les operacions del problema. Cada furgoneta té una representació pròpia que inclou dades com la seva ubicació d'origen, les estacions de destinació i la quantitat de bicicletes que transporta, i el benefici obtingut.
També es permet assignar destinacions a les furgonetes. Cada furgoneta té una llista d'estacions de destinació associades que indiquen on ha d'entregar les bicicletes recollides. Això és fonamental per gestionar la distribució de les bicicletes entre les estacions i assegurar-se que les furgonetes segueixin les rutes correctes.
Inclou informació sobre la quantitat de bicicletes que cada furgoneta transporta, així com la quantitat de bicicletes que deixa a cada estació de destinació. És essencial per calcular el cost de cada acció i l'estat actual, ja que depèn del moviment de les bicicletes.
El benefici està directament relacionat amb la quantitat de bicicletes recollides i lliurades, i aquestes dades es modifiquen en les llistes de cada furgoneta en `furgonetes_util`. Això permet avaluar quina furgoneta està generant més benefici en un estat concret.

L'atribut 'estacions_origen' ens permet saber quines estacions s'han assignat com a origen a una furgoneta específica, d'aquesta manera podem complir amb la restricció de no assignar una estació com a origen a una furgoneta si aquesta estació ja ha estat assignada com a origen a una altra furgoneta. No obstant això, també pot tenir alguns inconvenients respecte al consum de memòria o el temps computacional. Mantenir una llista de les estacions d'origen per a cada furgoneta en cada estat pot augmentar el consum de memòria, sobretot si tenim una gran quantitat de furgonetes en el problema.

Hem implementat l'atribut 'bicis_en_estacions' per fer un seguiment detallat de la quantitat de bicicletes presents en cada estació al llarg del procés de resolució.
El necessitem per gestionar i actualitzar l'estat de les estacions durant les missions de les furgonetes, i per mantenir un seguiment actualitzat de la quantitat de bicicletes que romanen en les estacions per cada estat i prendre decisions més informades sobre quines estacions visitar i quan.

En resum, l'atribut 'bicis_en_estacions' és essencial per gestionar les operacions de recollida i entrega de bicicletes a les estacions, així com per assegurar-se que la demanda i l'oferta de bicicletes es mantingui equilibrada i optimitzada al llarg del temps. Aquest atribut ajuda a garantir que el sistema funcioni de manera eficient i que es compleixin els objectius del problema. Tot i això, aquest atribut pot augmentar el temps de càlcul del problema, ja que cada operació requereix actualitzar les quantitats de bicicletes en totes les estacions. També és possible que contingui dades redundants o innecessàries, perquè no totes les estacions podrien ser rellevants en un estat concret. Això pot fer que sigui difícil optimitzar l'ús de la memòria.

2. 3. ANÀLISIS TAMANY DE L'ESPAI DE BÚSQUEDA

Per determinar la mida de l'espai de cerca en aquest context, cal tenir en compte diversos factors:

1. Nombre de furgonetes: El nombre de furgonetes utilitzades pot variar en cada estat, però inicialment, el nombre de furgonetes pot ser un factor en l'espai de cerca. Suposem que tenim `n` furgonetes. Aquest espai de cerca té una dimensió equivalent al nombre total de furgonetes disponibles, que és una quantitat fixa.

2. Nombre d'estacions disponibles com a destinacions: El nombre d'estacions que poden ser destinacions per a les furgonetes és una variable rellevant. Si hi ha "n" furgonetes i "k" estacions de destí, aquesta dimensió té "n^(k*2)" possibilitats.

3. Nombre d'estacions d'origen: El nombre d'estacions d'origen, inicialment assignades a les furgonetes, també pot variar. Aquesta dimensió depèn del nombre de furgonetes i del nombre d'estacions d'origen possibles. Si hi ha "n" furgonetes i "m" estacions d'origen, aquesta dimensió té "n^m" possibilitats.

4. Accions addicionals: Hi ha altres accions que poden modificar l'espai de cerca, com afegir o eliminar estacions de destí. El nombre total d'accions addicionals possible determinarà la dimensió d'aquest espai.

La mida de l'espai de cerca depèn d'aquests factors i pot ser bastant gran. Hauríem de considerar totes les possibles combinacions d'estats de les furgonetes i les estacions (assignacions d'estacions d'origen i destinació per a cada furgoneta, quantitat de bicicletes recollides i deixades, etc.). Aquesta combinació fa que l'espai de cerca sigui potencialment enorme i creixi exponencialment amb la quantitat de furgonetes i estacions disponibles.

La mida exacta de l'espai de cerca es pot calcular a partir dels valors concrets de 'n', 'm', 'k', i les accions addicionals.Per tant, l'espai de cerca total és causat per la multiplicació d'aquestes dimensions:

Espai de cerca total = Dimensió de les furgonetes x Dimensió d'estacions d'origen x Dimensió d'estacions de destí x Dimensió d'accions addicionals.

3. JUSTIFICACIÓ DELS OPERADORS ESCOLLITS

La tria dels operadors és una tasca crucial en el desenvolupament d'un projecte d'aquesta envergadura. Has de crear uns operadors que siguin capaços d'explorar tot l'espai de solucions, per així garantir una solució òptima al problema. Les nostres primeres aproximacions a l'hora de crear els operadors van ser crear simplement dos operadors: un per carregar bicis, i l'altre per descarregar bicis. Però després de donar-li unes voltes ens vam adonar que només aquests dos operadors no eren suficients per englobar tot l'espai de solucions. Així doncs, els operadors que tenim en el nostre projecte són:

-**Estacio_origen:** Aquest operador s'encarrega d'assignar una estació d'origen a una furgoneta. S'assegura que cap altra furgoneta està agafant bicis d'aquesta furgoneta. Aquest operador, però, no agafa bicicletes. És en la funció de cost on es calcula quantes bicicletes ha agafat aquella furgoneta. El factor de ramificació d'aquest operador és de (N*(E-1)) sent N el nombre de furgonetes i E el nombre d'estacions. Restem al nombre d'estacions 1 perquè no podem tornar a assignar la mateixa estació d'origen que ja té.

-**Estacio_desti**: Aquest operador és el responsable d'assignar estacions de destí a les furgonetes.El factor de ramificació d'aquest operador és de (N*(E-1)) sent N el nombre de furgonetes i E el nombre d'estacions. Restem al nombre d'estacions 1 perquè no podem tornar a assignar la mateixa estació de destí que ja té.

-**Afegir_estacio_desti**: Aquest operador s'encarrega d'assignar una estació de destí addicional (la segona) a una furgoneta. Realment podríem haver fusionat aquest operador amb el d'estacio_desti, però hem cregut més convenient aquesta distribució.El factor de ramificació d'aquest operador és de (N*(E-1)) sent N el nombre de furgonetes i E el nombre d'estacions. Restem al nombre d'estacions 1 perquè no podem tornar a assignar la mateixa estació de destí que ja visita en la primera parada.

-**Intercanviar_destins:** Aquest operador intercanvia l'ordre dels destins d'una furgoneta.
El factor de ramificació d'aquest operador és de (N) sent N el nombre de furgonetes, ja que aquest operador intercanvia d'ordre els destins de cada furgoneta.

-**Eliminar_destins**: Operador de destrucció d'una de les estacions destí.
El factor de ramificació d'aquest operador és de (N*2) sent N el nombre de furgonetes, es multipliquen per dos perquè si hi ha dos destins, primer s'elimina només un i després s'elimina l'altre.

-**Pivotar_destins**: Operador que consisteix a provar totes les possibles combinacions de destí en una furgoneta amb dos destins. És a dir, d'una furgoneta que té dues estacions de destí, deixa una de les dues estacions de destí tal com està, i l'altre la canvia per totes les possibles estacions de destí.El factor de ramificació d'aquest operador és de (N-1) sent N el nombre de furgonetes i li'n restem 1 perquè no pot anar a la mateixa estació de la qual ve.

Aquests són els operadors que hem plantejat nosaltres. Està clar que podrien haver-n'hi més, i inclús menys, però hem decidit que el resultat d'aquests operadors en conjunt és una bona aproximació a recórrer tot l'espai de solucions i tenen una bona proporció entre factor de ramificació i exploració dels possibles estats. Per justificar la tria d'aquests operadors hem fet un experiment (experiment 1), on provem aquests operadors en diversos escenaris diferents i avaluem la seva capacitat d'arribar a una resposta òptima. Allà trobareu una demostració detallada del perquè de la tria d'aquests operadors.

4. ESTRATÈGIES PER TROBAR L'ESTAT INICIAL

4. 1. GENERACIÓ ESTAT INICIAL NO 'GREEDY'

Una vegada ens vam decidir per l'elecció de l'estructura de dades per l'estat vam endinsar-nos en les estratègies per trobar la solució inicial.Primerament, ens vam fixar en generar la solució inicial no 'greedy' (generate_initial_state):

L'algorisme `generate_initial_state` comença inicialitzant una llista `furgonetes_util` que conté les dades de les furgonetes. Cada element d'aquesta llista correspon a una furgoneta. En el bucle for, s'itera a través del nombre màxim de furgonetes 'params.furgonetes_max'.

Per a cada furgoneta, es crea una llista amb tres elements:

- El primer element [i, 0] identifica la furgoneta, amb la primera posició com la identificació de l'estació d'origen de la furgoneta i la segona posició com el nombre de bicicletes que té la furgoneta. En un primer moment, totes comencen a la mateixa estació amb 0 bicicletes.

- El segon element [[i+1, 0]] és una llista que conté una altra llista per les estacions de destinació i el nombre de bicicletes que la furgoneta deixa en aquestes estacions, i inicialment és 0. En un primer moment, a totes les furgonetes se'ls hi assigna una única estació de destinació, la qual és l'estació consecutiva a l'estació assignada com a origen en la llista estacions.

- L'últim element és el benefici de la furgoneta, que comença amb un valor inicial de 0.

Seguidament, es fa el càlcul de bicicletes en estacions:Es crea una llista 'bicis_en_estacions' que guarda la informació sobre la quantitat de bicicletes en cada estació, utilitzant la funció 'bicis_estacions' amb la llista d'estacions 'params.estaciones.lista_estaciones'.

Després, l'algorisme utilitza la funció `calcul_estat_inicial` per a calcular les bicicletes que cada furgoneta agafa a l'estat inicial, les que deixa a les estacions i el benefici inicial. Aquest càlcul es fa per cada furgoneta i es basa en la capacitat de les furgonetes, la quantitat de bicicletes disponibles a l'estació d'origen i la demanda de les estacions de destí.

A més, es crea una llista anomenada 'estacions_origen' que guarda la informació sobre quines estacions d'origen han estat visitades per les furgonetes.
Finalment, la funció retorna un objecte 'Estat' que representa l'estat inicial del problema, utilitzant les dades i la configuració generades anteriorment. Conté els paràmetres del problema, les dades de les furgonetes i la llista d'estacions d'origen visitades.

4. 1. 1. JUSTIFICACIÓ DE L'ESTAT INICIAL NO 'GREEDY'

La generació d'aquest estat inicial és poc costosa en temps i en termes de recursos computacionals.

La selecció de les estacions d'origen per a cada furgoneta (dins de la funció 'generate_initial_state') és relativament eficient, ja que implica la iteració de les furgonetes que hi pot haver com a màxim per l'assignació de les estacions.

El cost de la generació de l'estat inicial es pot considerar baix en termes de memòria i potència de càlcul. Les dades emmagatzemades, com les llistes de furgonetes, les estacions i els seus estats, no són particularment grans ni exigents en memòria.

Aquesta generació de l'estat inicial és relativament senzilla d'implementar i de comprendre. Les operacions que es realitzen per calcular les furgonetes i les bicicletes associades no són molt complexes.

4. 2. GENERACIÓ DE L'ESTAT INICIAL 'GREEDY'

L'algorisme `generate_initial_state_greedy` és una variant que crea un estat inicial de manera avariciosa. Aquest algorisme ordena les estacions d'origen en funció de la quantitat d'excedents de bicicletes, i les assigna a les furgonetes d'acord amb aquest ordre. Això significa que les furgonetes aniran a les estacions amb més bicicletes sobrants. El procediment és el mateix en altres aspectes, com el càlcul de les bicicletes agafades i el benefici.

En la generació de l'estat inicial 'greedy' l'algorisme comença creant una llista 'maxims' que emmagatzema les estacions amb els majors excedents de bicicletes. Cada element d'aquesta llista és una tupla que consta de l'identificador de l'estació i la quantitat de bicicletes sobrants en aquesta estació.

Càlcul dels excedents de les estacions: L'algorisme recorre totes les estacions de la ciutat i calcula els excedents de bicicletes en cada estació. El càlcul dels excedents es realitza mitjançant la funció 'excedente', que considera la diferència entre les bicicletes següents (les disponibles en la següent hora) i la demanda de bicicletes de cada estació. Si aquesta diferència és positiva, significa que l'estació té bicicletes sobrants.

Selecció de les estacions amb majors excedents: Un cop calculats els excedents de totes les estacions, l'algorisme selecciona les estacions amb els majors excedents i les afegeix a la llista 'maxims'. Aquest procés assegura que les furgonetes comencin la seva ruta a les estacions amb més bicicletes sobrants.

Ordenació de les estacions d'origen: Un cop totes les estacions estan a la llista màxima, l'algorisme les ordena en ordre decreixent de bicicletes sobrants. D'aquesta manera, les estacions amb els majors excedents de bicicletes estaran al principi de la llista.

Inicialització de les furgonetes: L'algorisme crea una llista de furgonetes ‘furgonetes_util’ on cada furgoneta està representada com una llista que consta de tres elements:

- La identificació de l'estació d'origen, que s'assigna segons la llista ordenada ‘estacions_origen_ordenades’, i les bicicletes que recull la furgoneta (inicialment 0).
- Una llista buida per a les estacions de destí i les bicicletes que agafarà per a cada destinació (ja que encara no s'ha definit cap ruta).
- Un valor inicial de benefici (inicialment 0).

Seguidament, el procediment és el mateix que el que es segueix per l'altra generació de l'estat.

4. 2. 1. JUSTIFICACIÓ DE L’ESTAT INICIAL ‘GREEDY’

La nostra elecció de la generació de l'estat inicial greedy (generate_initial_state_greedy) es basa en una sèrie de consideracions que fan que sigui una opció vàlida per a abordar el problema d'optimització.

Produeix una solució de qualitat acceptable, encara que la generació de l’estat inicial no garanteix una solució òptima. La seva eficàcia es deu a l'ús d'una heurística simple: assignar furgonetes a estacions amb el màxim excés de bicicletes. Això pot ajudar a millorar significativament el valor de la funció objectiu.

En aquest mètode, les furgonetes simplement es distribueixen a les estacions segons la disponibilitat d'excedent de bicicletes. Això permet crear una solució inicial en un temps relativament breu, tot i que depèn també de la quantitat d'estacions que es presenten en el problema, ja que s'ha de recórrer totes les estacions per determinar a quines els hi sobra més bicicletes.

L'algorisme és senzill de programar i entendre. No requereix bucles complicats ni algorismes sofisticats. La seva simplicitat facilita la seva implementació i depuració.

No obstant això, cal tenir en compte que la generació 'greedy' pot ser sensible als valors inicials. Les solucions generades poden ser millorades mitjançant tècniques d'optimització posteriors. Això significa que, tot i que aquesta generació produeix una solució inicial ràpida i raonable, pot no ser la millor possible. Com a resultat, la seva utilització pot requerir l'ús posterior d'algoritmes d'optimització més avançats per a millorar la solució i obtenir resultats més bons.

5. FUNCIONS HEURÍSTIQUES

5. 1. FACTORS QUE INTERVENEN EN LA HEURÍSTICA DEL PROBLEMA

En l'heurística del problema intervenen la quantitat de bicicletes que es mouen i la seva conformitat amb les demandes de les estacions.

Solució Inicial: La solució inicial és l'estat inicial del sistema, on es decideixen les estacions d'origen de les furgonetes i les quantitats inicials de bicicletes per a cada furgoneta. Aquesta solució es pot generar mitjançant diferents mètodes, com la generació greedy o no 'greedy'.

Costos de Transport: El cost de transport és un element clau a tenir en compte en el problema. Es relaciona amb la quantitat de bicicletes que es mouen i pot ser positiu o negatiu segons si les bicicletes estan sent entregades o recollides. L'objectiu és minimitzar aquests costos.

Restriccions: Les restriccions són regles que limiten com les furgonetes poden ser assignades a les estacions i com les bicicletes poden ser transportades. Això inclou restriccions sobre la capacitat de les furgonetes, la quantitat de bicicletes que poden ser recollides en una estació i la limitació de visites de les furgonetes a un màxim de dues estacions.

Optimització: L'objectiu principal del problema és trobar la millor manera de moure les bicicletes per a satisfer la demanda de les estacions i minimitzar els costos de transport al mateix temps.

5. 2. FUNCIONS HEURÍSTIQUES ESCOLLIDES

5. 2. 1. FUNCIÓ HEURÍSTICA 1

L'estructura del codi que calcula l'heurística 1 del problema és la següent:

Funció `calcul_cost`:
Aquesta funció calcula el cost associat a l'acció d'agafar bicicletes d'una estació i deixar-les a altres estacions. Aquesta és la implementació i la justificació de cada plantejament:

1. **Bicicletes disponibles a l'estació d'origen**:

   - Es comença per identificar l'estació d'origen de la furgoneta i es verifica quantes bicicletes no s'han utilitzat fins ara en aquesta estació.

   - Justificació: Aquest pas estableix la quantitat màxima de bicicletes que es poden recollir a l'estació d'origen.

2. **Càlcul del nombre de bicicletes que pot recollir la furgoneta**:

   - Es compara la capacitat de la furgoneta amb les bicicletes no utilitzades disponibles a l'estació d'origen.

   - Si la capacitat de la furgoneta és igual o major que les bicicletes no usades, la furgoneta recull totes aquestes bicicletes.

   - En cas contrari, la furgoneta recull la màxima quantitat possible, que és la seva capacitat màxima.

   - Justificació: Aquest pas garanteix que la furgoneta no reculli més bicicletes del que pot emmagatzemar.

3. **Càlcul de les bicicletes que es deixen a les altres estacions**:

   - S'itera per les estacions de destí assignades a la furgoneta i es calcula la diferència entre la quantitat de bicicletes necessàries i les que realment es deixa a l'estació de destí.

   - Si l'estació de destí té més bicicletes de les necessàries, s'emmagatzemen les bicicletes necessàries a la furgoneta i s'actualitza la quantitat que es deixa a l'estació de destí.

   - Si l'estació de destí té una demanda superior a les bicicletes que la furgoneta pot proporcionar, la furgoneta deixa totes les bicicletes que ha recollit.

- Justificació: Aquest pas garanteix que les bicicletes es distribueixin de manera eficient, cobrint la demanda si n'hi ha, i evitant emmagatzemar bicicletes innecessàries.

4. **Càlcul del benefici total de la furgoneta**:

- Finalment, es calcula el benefici total generat per la furgoneta. Això implica comptabilitzar les bicicletes recollides i les deixades, així com aplicar penalitzacions en cas que s'hagi agafat més bicicletes de les necessàries.

- Justificació: Aquest càlcul és essencial per determinar l'eficàcia de les accions realitzades per la furgoneta i ajudar a prendre decisions optimitzades.

Funció `benefici`:
Aquesta funció calcula el benefici total generat per una furgoneta després de distribuir bicicletes. Detallem el seu funcionament i justifiquem les decisions preses:

1. **Càlcul de la diferència entre les bicicletes i la demanda a l'estació d'origen**:

- S'inicia amb el càlcul de la diferència entre la quantitat de bicicletes disponibles i la demanda a l'estació d'origen de la furgoneta.

- Justificació: Aquesta diferència representa la quantitat d'excés o la demanda no coberta a l'estació d'origen.

2. **Càlcul del benefici basat en la diferència**:

- Si la diferència és negativa (hi ha més demanda que bicicletes), el benefici es redueix pel nombre de bicicletes recollides.

- Si la diferència és positiva (hi ha més bicicletes que demanda), es calcula un benefici afegit basat en les bicicletes no utilitzades.

- Justificació: Aquest pas premia la distribució eficient de les bicicletes i penalitza l'excés d'agafar bicicletes quan no és necessari.

3. **Actualització de les quantitats a l'estació d'origen**:

- Es modifiquen les quantitats de bicicletes disponibles i no utilitzades a l'estació d'origen segons les bicicletes recollides per la furgoneta.

- Justificació: Això assegura que les quantitats reflecteixin amb precisió la situació de l'estació després de la distribució.

4. **Suma del benefici de les bicicletes deixades**:

   - Es calcula el benefici addicional basat en les bicicletes que es deixen a les estacions de destí, si s'han deixat.

5. 2. 2. FUNCIÓ HEURÍSTICA 2

L'estructura del codi que calcula l'heurística 2 del problema és la següent:

1. **Funció `calcul_estat_inicial`**:

   - Aquesta funció calcula l'estat inicial d'un escenari específic amb furgonetes, bicicletes en estacions i paràmetres del problema donats.

   - Determina l'estació d'origen de la furgoneta i la quantitat de bicicletes no utilitzades a aquesta estació.

   - Calcula quantes bicicletes pot recollir la furgoneta, tenint en compte la capacitat màxima de la furgoneta.

   - Calcula quantes bicicletes es deixen en cada estació de destí de la furgoneta.

   - Canvia el benefici obtingut de la furgoneta mitjançant la crida a la funció `benefici_furgoneta`.

2. **Funció `benefici_furgoneta`**:

   - Aquesta funció calcula el benefici obtingut per una furgoneta després de distribuir les bicicletes.

   - Es calcula tenint en compte les bicicletes recollides i deixades, així com les penalitzacions pel cost excessiu si s'agafen més bicicletes de les necessàries.

   - La funció rep l'identificador de la furgoneta, la llista de furgonetes utilitzades, la llista de bicicletes en estacions i els paràmetres del problema.

3. **Funció `cost_transport`**:

   - Aquesta funció calcula el cost del transport basat en el nombre de bicicletes, l'estació d'origen i l'estació de destí.

   - Utilitzem la fórmula donada a l'enunciat de la pràctica que implica el nombre de bicicletes i la distància entre les estacions per calcular el cost.

   - També rep els paràmetres del problema.

4. **Funció `distancia`**:

- Aquesta funció calcula la distància entre dues estacions. Utilitza les coordenades X i Y de les estacions per calcular la distància, i la converteix de metres a quilòmetres.
- El resultat es redueix a dues xifres decimals per arrodonir-ho.
- Rep els paràmetres del problema i els identificadors de les dues estacions.

En resum, aquest conjunt de funcions ens permet calcular de manera eficient el cost i el benefici associats a les accions de recollir i distribuir bicicletes per part de les furgonetes.

5. 3. EFECTES DE LES FUNCIONS HEURÍSTIQUES EN LA CERCA

La funció heurística proporciona una estimació de la qualitat de la solució, en la cerca local l'utilitzem per poder guiar a l'algoritme cap a la millor solució. Aquesta funció ha de ser precisa i apropiada per fer que es trobi una solució el més ràpid possible. En el nostre cas hem fet servir una heurística el més simple possible evitant bucles innecessaris per a calcular el cost del transport i el benefici del trasllat. D'aquesta manera evitem que el cost de la funció sigui gran i aquest afecti en el temps d'execució del problema. Finalment, restem els beneficis dels trasllats i els costos del transport de manera que al final acabem sumant els beneficis de cada furgoneta per obtenir el benefici total.Amb aquesta heurística evitem que el cost i el temps d'execució del problema augmenti.

5. 4. JUSTIFICACIÓ DE LA PONDERACIÓ

En la nostra heurística no hi ha ponderacions, ja que el que volem és maximitzar el benefici dels trasllats de bicicletes i en conseqüència minimitzem els costos en transport de bicicletes, perquè quan fem la suma de tots els beneficis es quedarà amb el més alt que és el que té el màxim de bicicletes traslladades amb el cost mínim de transport.

6. EXPERIMENTS

6. 1. EXPERIMENT 1

Determinar qué conjunto de operadores da mejores resultados para una función heurística que optimice el primer criterio (el transporte es gratis) con un escenario en el que el número de estaciones es 25, el número total de bicicletas es 1250 y el número de furgonetas es 5. Deberéis usar el algoritmo de Hill Climbing. Escoged una de las estrategias de inicialización de entre las que proponéis. A partir de estos resultados deberéis fijar los operadores para el resto de experimentos.

La tria dels operadors és una part crucial en el desenvolupament d'un projecte d'aquestes característiques. El conjunt d'operadors ha de ser capaç de recórrer l'espai de solucions per tal de trobar una solució òptima.

El nostre grup hem pensat un total de 6 operadors, que considerem que són adequats per recórrer l'espai de solucions i que seran capaços d'arribar a una bona solució. No obstant, ara procedirem a experimentar amb els operadors per veure si realment són necessaris o són prescindibles.

El procediment d'aquest experiment serà agafar subconjunts d'operadors i avaluar la seva capacitat d'arribar a una solució.
Per fer l'experiment utilitzarem la solució inicial aleatòria, d'aquesta manera estem explotant el fet de veure la capacitat de cerca dels nostres operadors. Si comencéssim amb una solució inicial molt bona, l'avaluació dels operadors seria més complicada.
L'experiment consistirà en 5 iteracions en cada subconjunt d'operadors. Per les 5 iteracions, fem una mitja on avaluarem el benefici obtingut, i també el temps d'execució.
Per aquest experiment farem servir el segon heurístic (el que té en compte el cost de transport), ja que així també estem incitant a què s'hagi de recórrer més en l'espai de solucions.

OBSERVACIONS	Poden haver operadors més o menys rellevants
PLANTEJAMENT	Escollim diferents subconjunts d'operadors i analitzem el seu resultat
HIPÒTESIS	Tots els operadors plantejats són imprescindibles (H_0)
MÈTODE	● Provarem l'experiment en dos escenaris diferents (utilitzant dos llavors diferents) ● Realitzarem 5 iteracions per cada subconjunt d'operadors ● Utilitzarem l'algorisme de Hill Climbing. ● Mesurarem diferents paràmetres per a realitzar la comparació.

Abans de començar l'experiment, els resultats que esperem són que tots els operadors són necessaris, és a dir, serem incapaços de refutar la hipòtesis nul·la. I també esperem que com menys operadors s'utilitzin, pitjors resultats trobarem.A continuació presentem la taula

amb els resultats obtinguts. Hem provat els resultats en 2 escenaris diferents (amb llavor 42 i 21). El benefici i el temps d'execució que veureu a la taula és una mitja de les 10 iteracions fetes per a cada subconjunt d'operadors.

Semilla = 42:

OPERADORS	TEMPS D'EXECUCIÓ (ms)	BENEFICI
TOTS	290	40.83
TOTS - {Estacio_origen}	120	-4.65
TOTS - {Estacio_desti}	90	1.66
TOTS- {Intercanviar_destins}	250	39.9
TOTS - {Afergir_estacio_desti}	145	37.08
TOTS - {Eliminar_estacions}	265	40.83
TOTS - {Pivotar_destins}	255	39.1
TOTS - {Eliminar_estacions, Intercanviar_estacions}	280	40.83
TOTS - {Eliminar_estacions, Intercanviar_estacions, Pivotar_destins}	270	40.83
CAP OPERADOR		-75.8

Semilla = 21:

OPERADORS	TEMPS D'EXECUCIÓ (ms)	BENEFICI
TOTS	330	70.45
TOTS - {Estacio_origen}	130	29.89
TOTS - {Estacio_desti}	190	37.4
TOTS - {Intercanviar_destins}	300	68.9
TOTS - {Afergir_estacio_desti}	145	51.3
TOTS - {Eliminar_estacions}	360	70.45
TOTS - {Pivotar_destins}	220	57.3
TOTS - {Eliminar_estacions, Intercanviar_estacions}	390	70.45

TOTS - {Eliminar_estacions, Intercanviar_estacions, Pivotar_destins}	241	61.81
CAP OPERADOR		-53.3

Després d'observar els resultats obtinguts en l'experiment en la taula superior, podem treure algunes conclusions.
En primer lloc, podem veure que quan provem d'executar utilitzant tots els operadors menys l'operador 'Eliminar_estacions', tant el benefici com el temps d'execució és semblant a utilitzar tots els operadors. Una cosa semblant passa amb l'operador 'Intercanviar_destins'. Inclus prescindint dels dos operadors simultàniament els resultats són semblants a utilitzar tots els operadors. Això pot suggerir que aquests operadors no són tan rellevants com els altres, i inclús potser valdria la pena suprimir-los per així reduir el factor de ramificació. No obstant això, el nostre grup considerem adient continuar tenint aquests dos operadors, ja que el temps d'execució no varia gaire respecte usar-los o no, i per tant millor no perjudiquen molt el temps d'execució del programa. A més, és probable que pels experiments que hem realitzat aquests operadors no siguin tan útils, però potser en altres escenaris sí que ho serien.
No obstant això, tenim conscient que aquests operadors són els operadors que ens aporten menys, així que en l'hipotètic cas que volguéssim reduir la complexitat del programa, per així reduir el seu temps i cost, probablement podríem suprimir un o ambdós operadors.També podem observar que operadors com 'estació_origen' i 'estacio_desti' si no s'utilitzen, els resultats siguin dolents. I realment això té sentit, ja que aquests operadors són els encarregats d'assignar una estació on agafar bicis i assignar una estació on deixar bicis. I si no hi ha aquests operadors les furgonetes no agafaràn o no deixaràn bicis.

Per aquest experiment, no hem pogut modificar els valors des del fitxer main, pel fet que els operadors no estan disponibles des del main. És per això que no hem modificat el main, i per tant no adjuntem cap tros de codi de l'experiment.Tanmateix, la manera com hem dut a terme l'experiment ha estat la següent. Per cada subconjunt d'operadors, manualment hem modificat el fitxer estat, de manera que no generés successors amb aquells operadors. Manualment, hem anat recollint els resultats de cada subconjunt d'operadors i fent la mitja i anotant-los a la taula.

6. 2. EXPERIMENT 2

Determinar qué estrategia de generación de la solución inicial da mejores resultados para la función heurística usada en el apartado anterior, con el escenario del apartado anterior y usando el algoritmo de Hill Climbing. A partir de estos resultados deberéis fijar también la estrategia de generación de la solución inicial para el resto de experimentos.

Una de les parts del problema en la cerca local que més pot afectar en el cost de la solució final és la solució inicial. És difícil determinar si una solució inicial és millor o pitjor que un altre, ja que no sabem la forma de l'espai de cerca. Per tant, per determinar l'estratègia de la generació de la solució inicial durem a terme un experiment per esbrinar quina és la millor.

En aquest cas utilitzarem dos mètodes de generació de solucions inicials, un el qual assignarà estacions d'origen i destí ordenadament segons l'identificador de l'estació. Aquesta generació de la solució inicial pot semblar ordenada, però les estacions es generen de manera aleatòria, per tant, la generació de la primera solució inicial serà aleatòria. Per fer la segona solució inicial ordenarem les estacions per excedent més gran i les assignarem per a cada furgoneta d'excedent més gran a més petit.

Per a fer aquest experiment realitzarem deu rèpliques de l'experiment on provarem cada mètode d'inicialització. Per a cada experiment individual usarem la mateixa llavor de nombres aleatoris, de manera que cada rèplica sigui idèntica en les proves per a cada mètode d'inicialització. Per a veure els resultats haurem de mesurar alguna cosa que ens doni informació per a determinar quina generació de solució inicial és millor, per exemple podem usar el temps per comparar el cost de les dues generacions, i a més també podem mesurar el resultat per veure quin dels dos generadors és més eficient.

En aquest experiment utilitzarem el primer heurístic el qual no té en compte el cost del transport. A més farem servir els operadors que hem determinat en l'experiment 1, ja que són els que ens han donat millors resultats. En aquesta taula podem resumir les característiques d'aquest experiment:

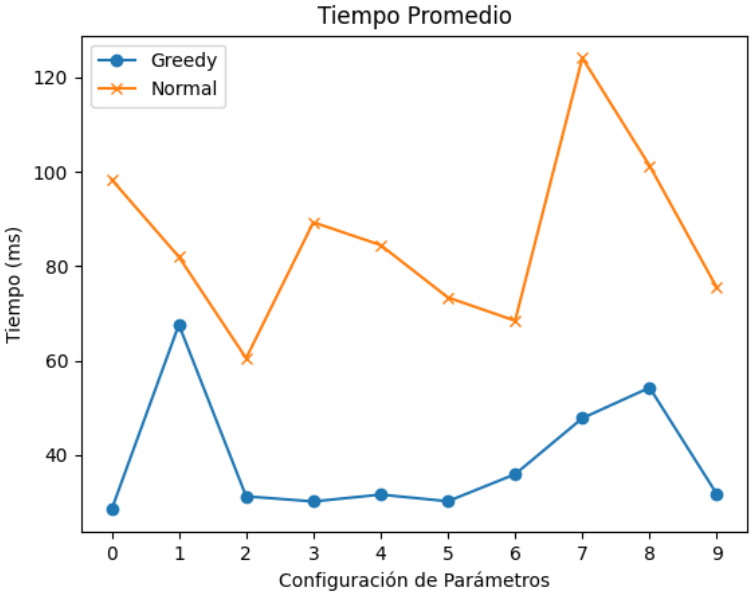
OBSERVACIONS	Poden haver mètodes d'inicialització que obtenen millors resultats
PLANTEJAMENT	Escollim diferents mètodes d'inicialització i notem les seves solucions
HIPÒTESIS	Tots els mètodes d'inicialització són iguals (H0) o hi ha mètodes millors que altres
MÈTODE	● Triarem 10 llavors aleatòries, una per a cada rèplica ● Realitzarem 5 rèpliques per a cada llavor utilitzant la inicialització aleatòria i avariciosa. ● Farem servir l'algorisme de Hill Climbing. ● Mesurarem diferents paràmetres per a realitzar la comparació.

Abans de dur a terme aquest experiment esperàvem millors resultats amb la solució inicial greedy que amb la solució inicial aleatòria. En aquest cas no ens equivocàvem, ja que la solució inicial greedy fa que la cerca local estigui més a prop de maximitzar el benefici.

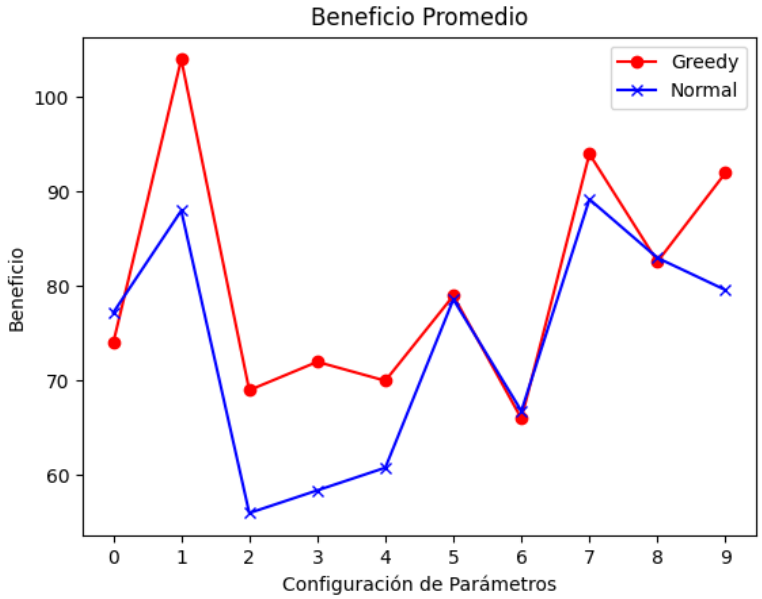
En la següent taula podem veure els resultats després de realitzar les execucions pertinents amb les diferents solucions inicials:

	TEMPS		SOLUCIÓ	
RÈPLICA	GREEDY	ALEATORI	GREEDY	ALEATORI
1	78.67	102.72	74.0	76.4
2	79.04	101.02	104.0	88.0
3	43.7	73.73	69.0	57.8
4	40.01	100.12	72.0	64.6
5	34.08	91.16	70.0	62.2
6	35.99	67.57	79.0	79.0
7	45.77	65.72	73.0	72.2
8	91.01	130.61	94.0	88.8
9	75.3	107.4	83.0	80.8
10	37.64	104.47	92.0	80.8

Com podem observar en la taula els temps greedy són molt millors en totes les rèpliques que hem executat. Per veure la relació entre la solució inicial greedy i l'aleatòria hem calculat les mitges de totes les repliques realitzades, en aquest cas la solució inicial greedy té una mitjana de 55,7 i la solució inicial aleatòria una mitja de 94,3. Si fem la divisió per veure la relació en els temps d'execució el resultat és d'1,69, per tant, la solució inicial greedy fa que el temps d'execució sigui gairebé dues vegades més ràpid. En el següent gràfic podem observar clarament la diferència:



Respecte a la solució podem veure que els beneficis són força semblants. En la majoria dels casos la solució inicial greedy dona millors beneficis que la solució inicial aleatòria, en alguns casos la solució inicial greedy té els mateixos beneficis que la solució inicial aleatòria. Amb aquestes dades podem veure com amb una solució inicial greedy es pot arribar a un millor resultat que amb la solució inicial aleatòria. En el següent gràfic també podem veure aquestes diferències:



Per tant, podem arribem a la següent conclusió, la solució inicial greedy és més bona respecte al temps d'execució i el benefici obtingut que la solució inicial aleatòria, ja que com hem analitzat abans els temps i les solucions són millors.

6. 3. EXPERIMENT 3

Determinar los parámetros que dan mejor resultado para el Simulated Annealing con el mismo escenario, usando la misma función heurística y los operadores y la estrategia de generación de la solución inicial escogidos en los experimentos anteriores.

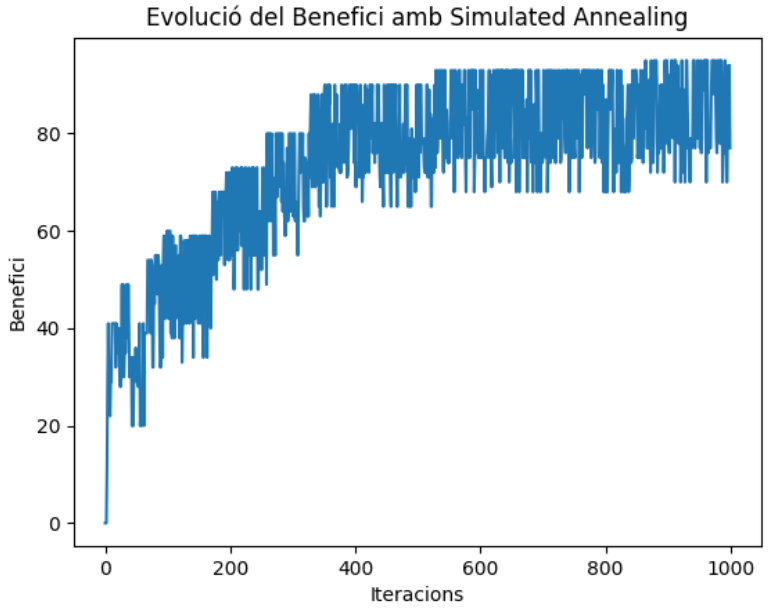
Els paràmetres que hem de determinar del Simulated Annealing són els següents:

- Número total d'iteracions
- Paràmetre k de la funció d'acceptació d'estats
- Paràmetre λ de la funció d'acceptació d'estats

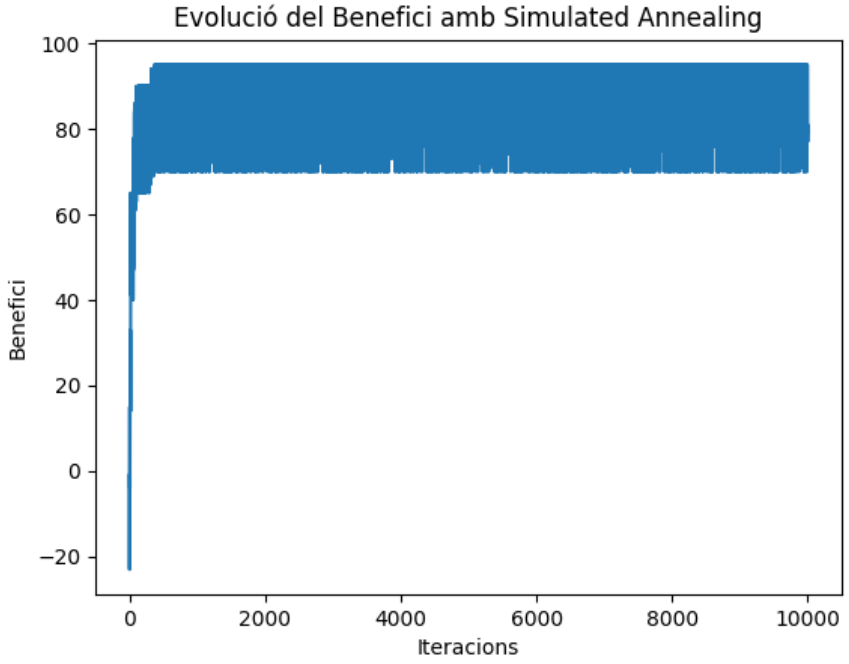
En aquest experiment veurem quins són els paràmetres que donen millors resultats per al Simulated Annealing en un escenari en el qual el número d'estacions està fixat en 25, el número de bicicletes que hi ha són 1250 i el número de furgonetes màximes és 5. A més, utilitzarem els operadors escollits en l'experiment 1 i l'estratègia d'inicialització de l'estat inicial escollit en l'experiment 2, és a dir, la generació de l'estat inicial 'greedy', i utilitzant la primera heurística, que no té en compte el cost de transportar les bicicletes.

OBSERVACIONS	Els valors dels paràmetres òptims depenen de la mida del problema i de la heurística
PLANTEJAMENT	Escollim diferents valors pels paràmetres i escollim els més òptims conforme els resultats
HIPÒTESIS	Els valors dels paràmetres número d'iteracions, k i lambda òptims són una k petita, una lambda relativament gran i un número d'iteracions gran (H0)
MÈTODE	● Observem com evoluciona l'algorisme augmentant el número d'iteracions. ● Observem com evoluciona l'algorisme augmentant el valor de k. ● Observem com evoluciona l'algorisme disminuint el valor de lambda. ● Trobem la combinació dels valors k i lambda que dona millors resultats.

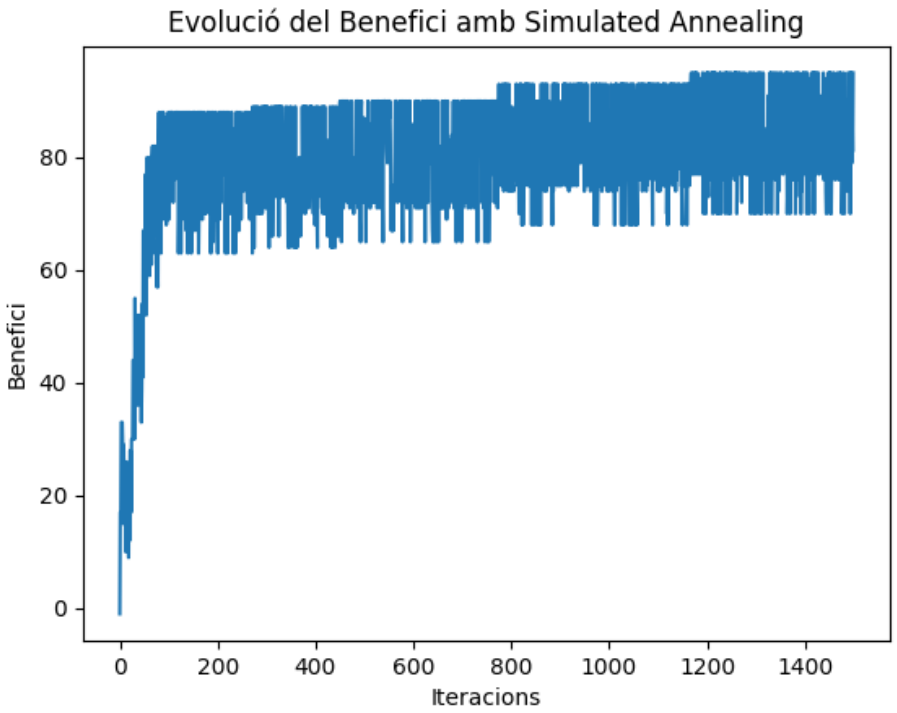
Primerament, generem la següent gràfica, on es pot veure com evoluciona el benefici per un escenari en concret utilitzant el Simulated Annealing amb 1000 iteracions, un valor de k de 5 i un valor de λ de 0.01.



Es veu clarament com el benefici va pujant i baixant en cadascuna de les iteracions, però com la tendència és un augment del benefici, de fet no arriba a descendir mai. Això pot significar que el número d'iteracions és insuficient. Per tant, provem ara d'augmentar el número d'iteracions a 10000.



Si ens fixem a partir de les 1500 iteracions l'algorisme no troba cap benefici millor, podem afirmar que hem trobat el màxim de benefici que es pot trobar amb aquest escenari, que és 95 €. D'aquí podem concloure que el número màxim d'iteracions no ha de ser necessàriament tan gran. Provem de reduir a 1500 iteracions:



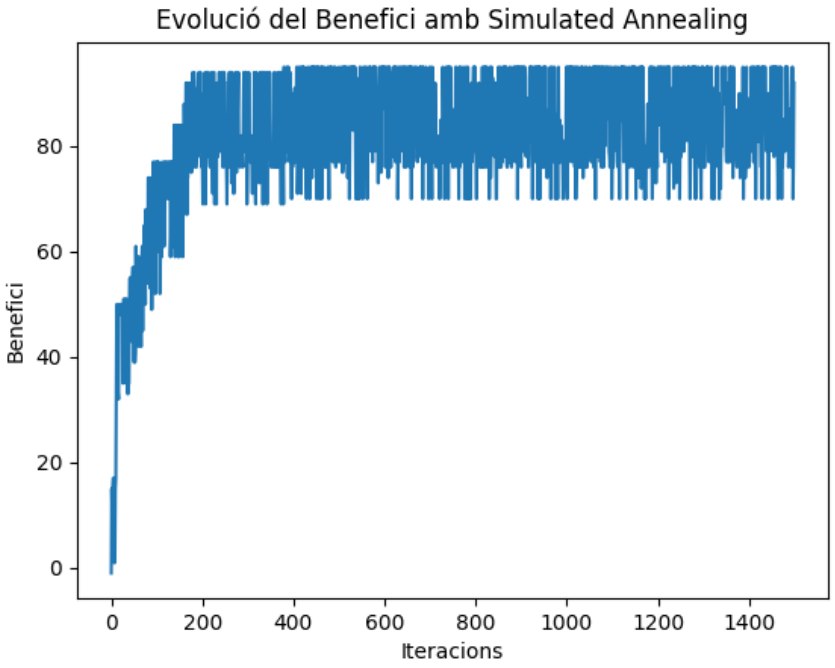
Ens trobem doncs amb l'escenari que esperàvem, en el qual a les 1500 iteracions el benefici màxim es manté constant.

Podem dir que hem determinat el número d'iteracions necessàries, ara hem de trobar les 'k' i 'λ', hem de tenir en compte que la funció amb 'k' igual a 5 i 'λ' té pujades i baixades alternades en tot el seu recorregut.

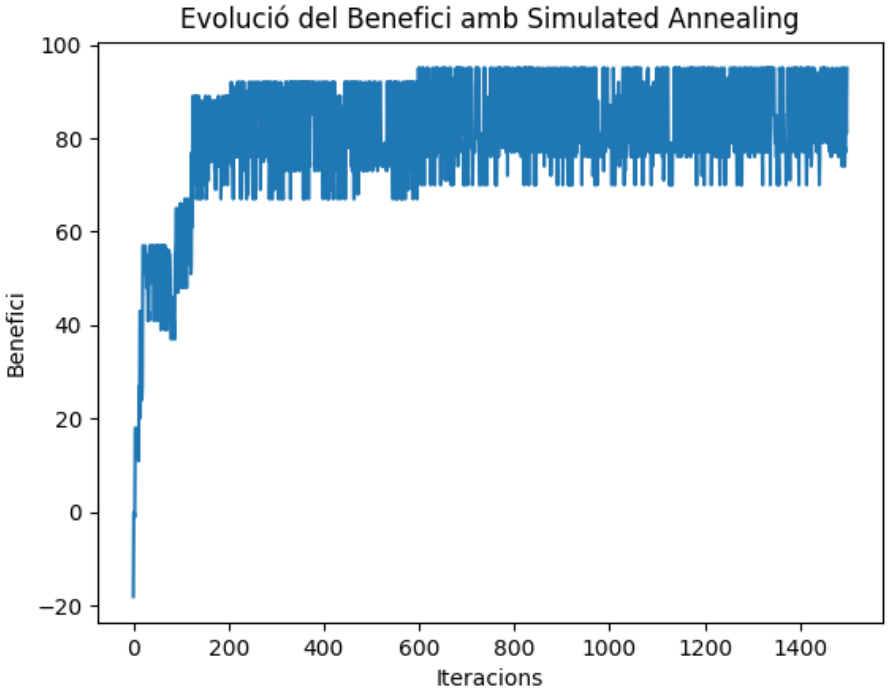
Tal com s'indica en l'enunciat de la pràctica 'Podemos ver que si vamos variando k, a medida que aumenta, el número de iteraciones en las que la probabilidad de aceptar estados peores es alta va aumentando. Si vamos variando λ, la velocidad a la que esa probabilidad desciende va aumentando cuanto mayor es, disminuyendo además el número de iteraciones que hacen falta para llegar a probabilidad 0' com més alt el valor de 'k' més probabilitats d'acceptar estats pitjors, i com més alt el valor de λ menys probabilitats d'acceptar estats pitjors.

Primer provem modificant el valor de k fixant els altres dos paràmetres per veure com varia el benefici a mesura que augmentem el valor del paràmetre. El que ens esperem és que a més valor de la 'k' més trigarà en arribar al valor màxim de benefici. Aquestes són les gràfiques resultants:

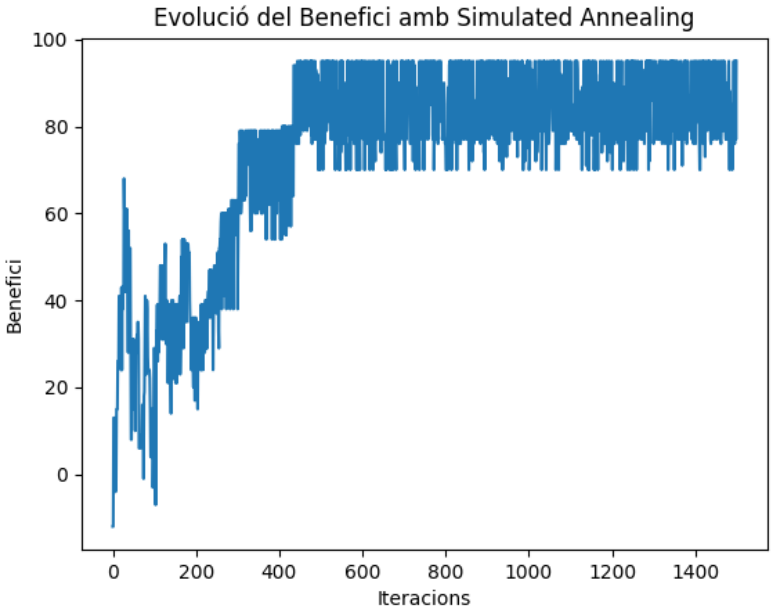
k = 1:


Evolució del Benefici amb Simulated Annealing
Benefici
Iteracions
0
20
40
60
80
0
200
400
600
800
1000
1200
1400

k = 5:


Evolució del Benefici amb Simulated Annealing
Benefici
Iteracions
−20
0
20
40
60
80
100
0
200
400
600
800
1000
1200
1400

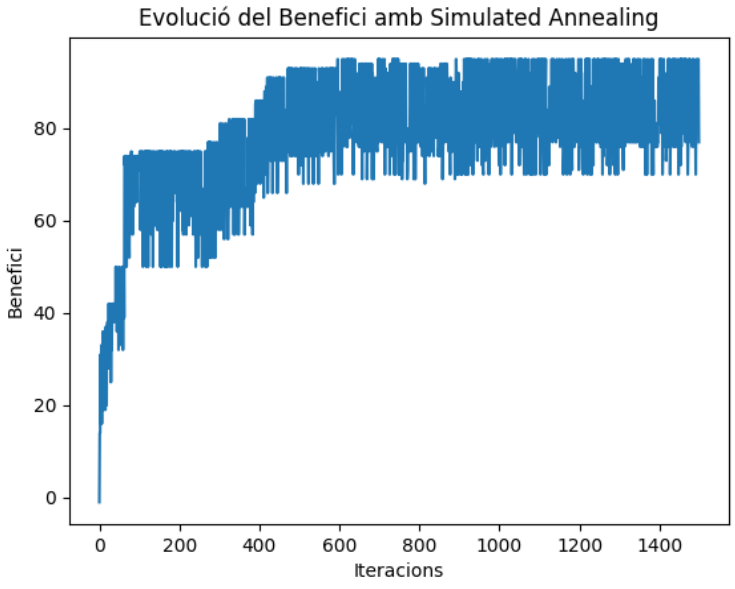
k = 25:



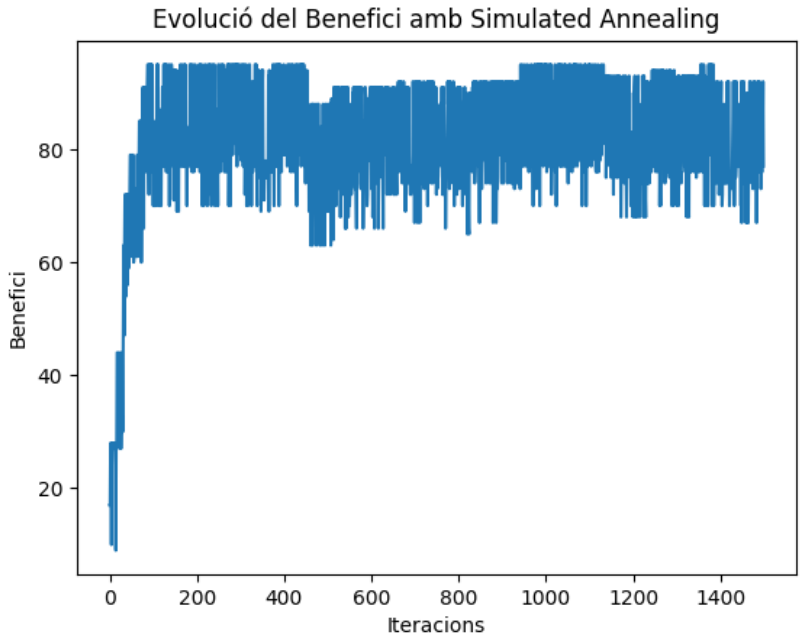
Els resultats són els que ens esperàvem, a mesura que va augmentant el valor del paràmetre 'k' més probabilitats té d'acceptar un estat que li resulta en menys benefici, tal com s'ha mencionat anteriorment.

Seguidament, modifiquem el valor de λ, mantenint constant els altres paràmetres. Com ja hem mencionat, quan reduïm el valor de λ, s'allarga el període durant el qual s'accepten estats de pitjor qualitat, allargant al mateix temps la quantitat d'iteracions necessàries per aconseguir la convergència. A més a més, això provoca que la diferència entre estats successius disminueixi amb el temps:

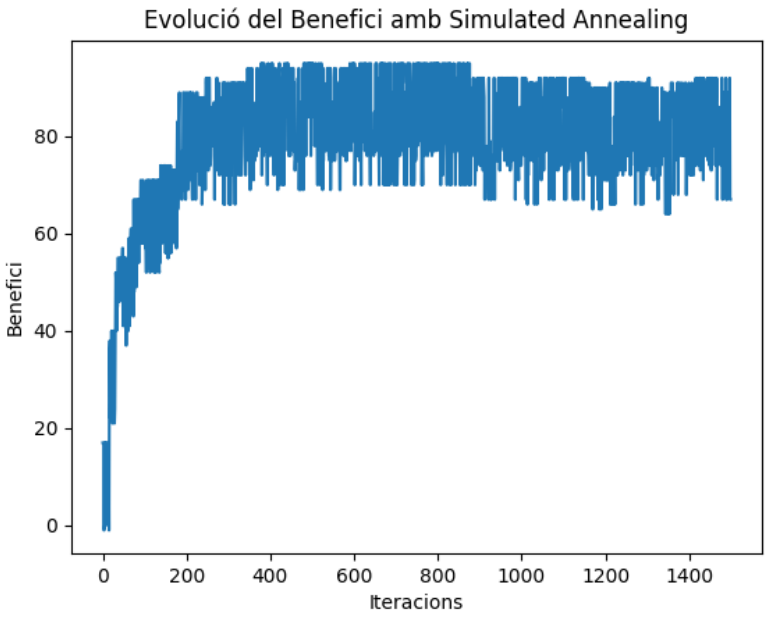
λ = 0.001:



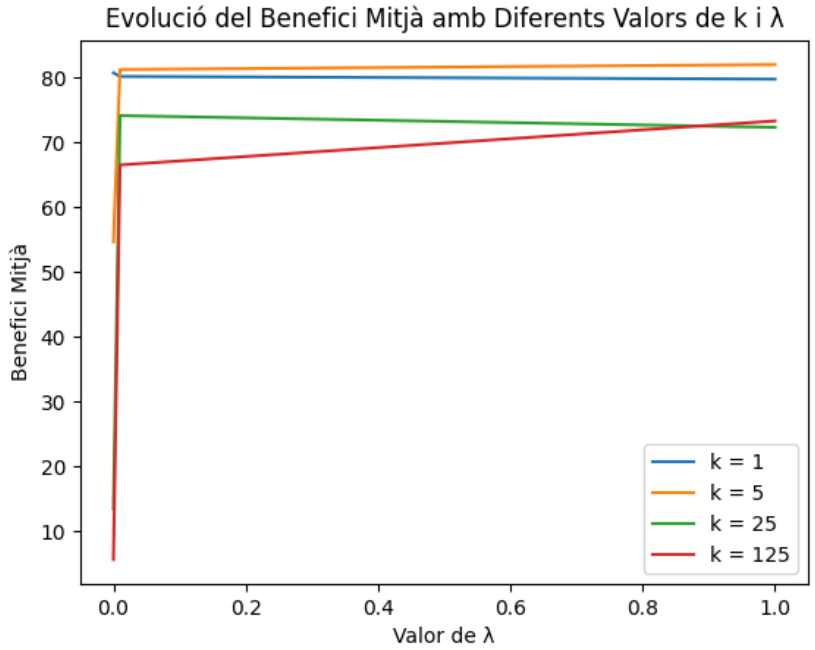
λ = 0.00001:



λ = 0.00000001:



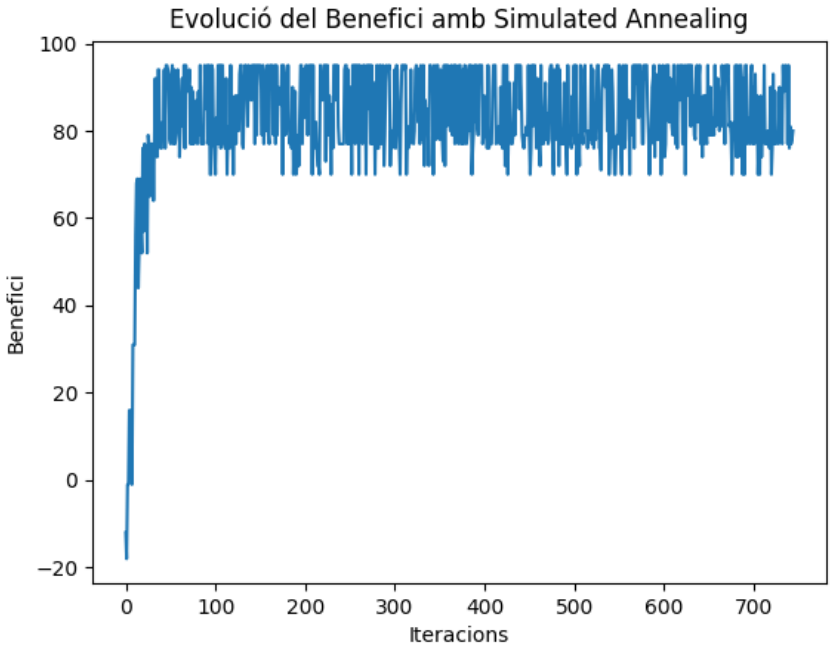
Finalment, realitzem una sèrie d'experiments per determinar quins són els valors òptims de k i λ. Utilitzem un gran nombre d'iteracions per garantir que aconseguim la convergència i provarem valors de k = {1, 5, 25, 125} i λ = {1, 0.01, 0.0001}. A través d'aquests punts, podrem tenir una idea d'on es troben els valors òptims i podrem realitzar una exploració més precisa a partir d'aquí:



Podem observar que els valors dels paràmetres que han donat un benefici mitjà més elevat han sigut amb una k igual a 1 o 5 i amb una lambda entre 0.01 i 1. Tot i que podrien no ser els valors òptims fixem els valors dels paràmetres següents:

- Número total d'iteracions: 1500
- Paràmetre k de la funció d'acceptació d'estats: 1 o 5
- Paràmetre λ de la funció d'acceptació d'estats: 0.01 i 1

Finalment, la següent figura mostra l'evolució del cost per a una única execució del problema amb aquests paràmetres:



Com a conclusió, si la magnitud del problema augmenta hauríem d'executar el problema amb una lambda més petita per a recórrer més estats o un número d'iteracions més gran.

Cal destacar que pel Simulated Annealing generem una operació aleatòria tal com se'ns demana en la pràctica que fa que el cost computacional es vegi reduït molt significativament.

Pel que fa a la hipòtesi podem afirmar que el resultat obtingut ha estat semblant al que esperàvem, tot i tenir no obtenir el resultat esperat exacte.

6. 4. EXPERIMENT 4

Dado el escenario de los apartados anteriores, estudiad cómo evoluciona el tiempo de ejecución para hallar la solución en función del número de estaciones, furgonetas y bicicletas asumiendo una proporción 1 a 50 entre estaciones y bicicletas y 1 a 5 entre furgonetas y estaciones. Para ello empezad con 25 estaciones e id aumentándolas de 25 en 25 hasta que veáis la tendencia. Usad el algoritmo de Hill Climbing y la misma heurística.

Ens pot interessar saber quant augmentarà el cost de la cerca en funció al tamany del problema. Podem experimentar utilitzant ciutats les quals tinguin més estacions per tant més bicicletes i en conseqüència més furgonetes.

Podem assumir que l'increment del temps d'execució serà causat per l'augment del factor de ramificació dels operadors degut a l'augment de l'espai de solucions.

En aquest experiment realitzarem diverses execucions per a cada mida i utilitzarem els valors mitjans per ajustar la funció del temps la qual considerarem comparable entre les rèpliques amb llavors diferents. Podem resumir l'experiment en:

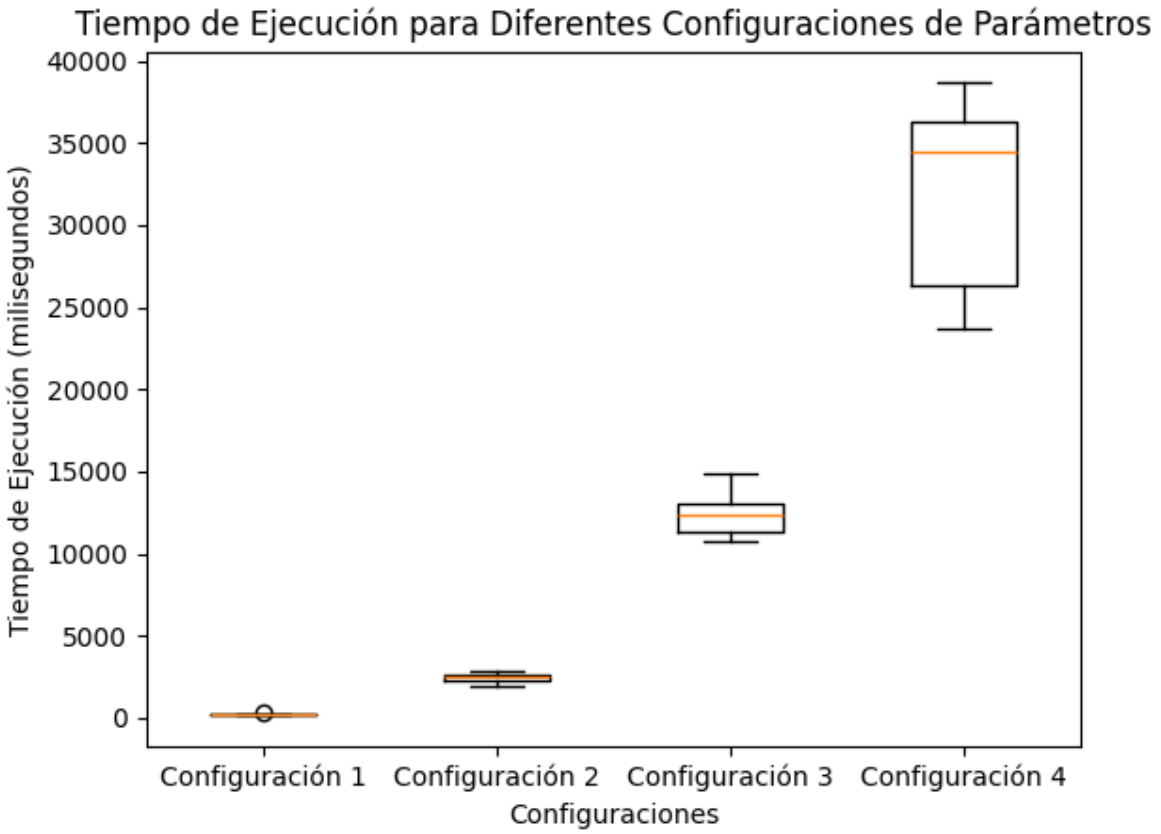
OBSERVACIONS	El temps segueix una funció creixent respecte a la mida del problema
PLANTEJAMENT	Escollim diferents mides de problema i observem els seus temps d'execució
HIPÒTESIS	La funció serà almenys quadràtica respecte a la mida del problema
MÈTODE	● Triarem 10 llavors aleatòries per a cada mida del problema ● Executarem 10 vegades per a cada mida (una per cada llavor diferent)i farem mitges dels resultats ● Experimentarem amb problemes de 25, 50, 75 i 100 ciutats ● Farem servir l'algorisme de Hill Climbing amb inicialització avariciosa ● Mesurarem el temps de còmput per realitzar la comparació

Abans de fer aquest experiment esperàvem que el temps d'execució augmentés a mesura que la mida del problema fos més gran, ja que com hem explicat anteriorment els factors de ramificació dels operadors. Un cop executat l'experiment hem vist que teníem raó i realment passava el que hem explicat anteriorment.

Com podem veure en aquesta taula el temps d'execució augmenta considerablement:

MIDA	25	50	75	100
MITJA	58.45	651.32	3033.43	7482.14
DESVIACIÓ	24.44	239.31	1032.17	1820.28

En aquest boxplot també podem veure clarament com augmenta el temps d'execució i a més la desviació, ja que en problemes de mida més gran hi ha més possibles solucions i, per tant, es poden donar temps més diferents.



Després de veure els resultats en la taula i en la gràfica de box plots observem que el temps creix de manera exponencial a mesura que la mida del problema creix. Això és degut tal com explicat anteriorment a l'augment del factor de ramificació i, per tant, algoritme té més estats per recórrer.

Per tant, podem veure que la mida del problema sí que afecta en el temps d'execució d'aquests. Perquè aquest temps sigui el mínim possible és important que hi hagi una heurística amb un cost mínim igual que la copia, ja que són mètodes que s'utilitzen constantment. Si el cost d'aquests mètodes són elevats a la llarga el temps d'execució del programa augmenta.

6. 5. EXPERIMENT 5

Dado el escenario del primer apartado, estimad la diferencia entre el beneficio obtenido, la distancia total recorrida y el tiempo de ejecución para hallar la solución con el Hill Climbing y el Simulated Annealing para las dos heurísticas que habéis implementado (los experimentos con la primera ya los tenéis).

En aquests problemes és important l'elecció de l'algorisme de cerca local, ja que aquest pot afectar tant en el temps d'execució, com en el benefici del problema que estem resolent, en aquest cas compararem quin dels dos algoritmes que hem implementat, el hill Climbing o el Simulated Annealing és millor per a resoldre aquest problema.
A més d'escollir un dels dos algoritmes veurem com aquests afecten en les dues heurístiques que hem implementat per poder solucionar el problema del Bicing, la primera heurística no comptarà els costos de transport i, en canvi, la segona heurística sí que tindrà en compte el cost del transport.
Per a dur a terme aquest experiment usarem els dos algoritmes i les dues heurístiques per a poder veure les diferències entre elles en el benefici, la distància i el temps d'execució. En aquest cas les mesures ens serviran per veure quin algorisme funciona millor per cada heurística i quin hauríem de triar per a cada una de les heurístiques, ja que una heurística té en compte el cost de transport i l'altre no el té en compte.
Per poder veure les diferències significatives entre els algoritmes executarem 10 rèpliques per a cada experiment amb una llavor per a cada rèplica, de manera puguem fer la mitjana dels resultats, ja que considerarem que els beneficis les distàncies i el temps d'execució és comparable.
Aquesta taula resumeix l'experiment:

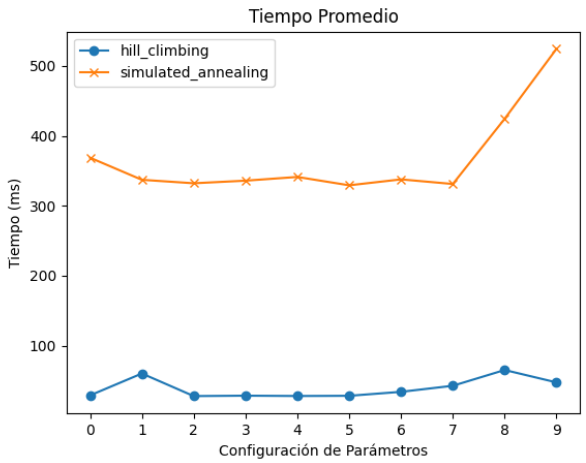
OBSERVACIONS	La selecció de l'algorisme més apropiat varia en funció de les característiques del problema.
PLANTEJAMENT	Executem els algorismes de cerca local i comparem les mesures per veure quin és el millor
HIPÒTESIS	Els dos algoritmes donaran un resultat semblant (H0)
MÈTODE	● Triarem 10 llavors aleatòries per a cada mida del problema ● Executarem 10 vegades per a cada heurística (una per cada llavor diferent) i farem mitges dels resultats ● Mesurarem el temps de còmput, el benefici i la distància per realitzar la comparació

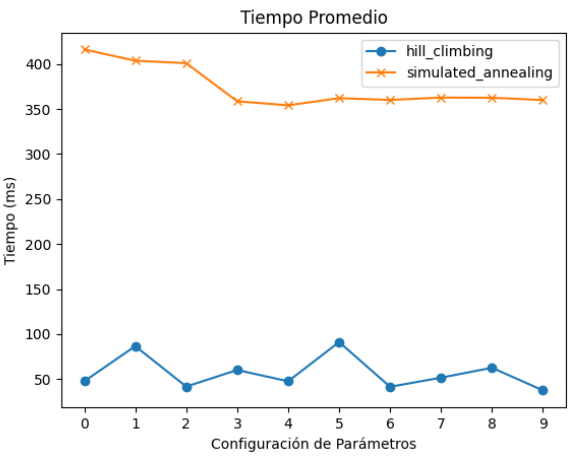
Abans de dur a terme aquest experiment esperàvem uns resultats força semblants encara que pensàvem que el Simulated Annealing podria donar uns beneficis una mica millors

encara que això suposa més temps d'execució. Un cop executat l'experiment en aquesta taula podem observar els resultats:

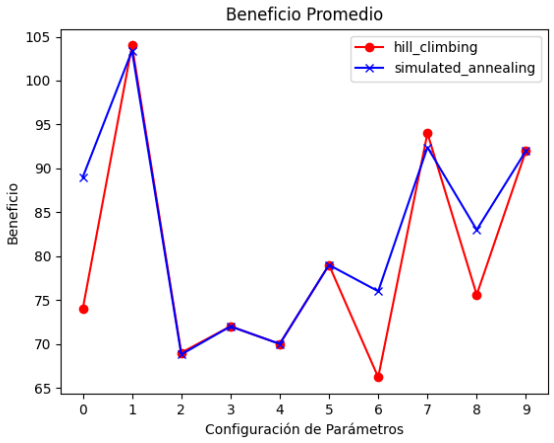
	Hill Climbing			**Simulated Annealing**		
HEURÍSTICA	BENEFICI	DISTÀNCIA	TEMPS	BENEFICI	DISTÀNCIA	TEMPS
HEURÍSTICA 1	92.0	31.03	29.52	92.0	45.8	339.14
HEURÍSTICA 2	49.2	11.57	37.24	49.6	24.9	353.86

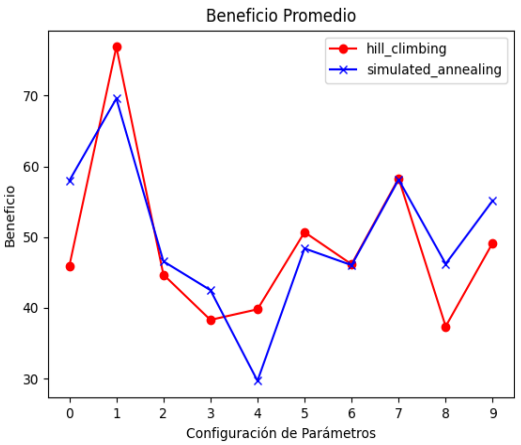
Podem observar en aquesta taula, com, tant en l'heurística 1 com en l'heurística 2 els temps d'execució és bastant més gran en el Simulated Annealing que en el HIll Climbing, ja que l'algoritme de Simulated Annealing explora l'espai de cerca de manera més amplia. En els gràfics següents podem observar la diferència en temps d'execució entre el Hill Climbing i el Simulated Annealing en l'heurístic 1 i en l'heurístic 2 respectivament:





A continuació, podem observar que els beneficis entre el Hill Climbing i el Simulated Annealing en l'heurística 1 son iguals, però, en canvi, en l'heurística 2 podem observar com el Simulated Annealing és una mica millor que el Hill Climbing, això és degut tal com hem explicat anteriorment al fet que aquest algoritme explora un espai més gran. Seguidament, podem veure els gràfics dels beneficis de l'heurística 1 i l'heurística 2 respectivament:





Finalment, podem veure com la distància recorreguda entre el Hill Climbing i el Simulated Annealing tant en l'heurístic 1 com en l'heurístic 2 és diferent. En el cas de l'heurístic 1 no té en compte la distància i, per tant, els quilòmetres que poden recorre les furgonetes són il·limitats. A més entenem que com l'espai de cerca del Simulated Annealing és més gran es troba amb solucions que recorren més quilòmetres. En l'heurístic 2 ens trobem en una situació semblant, ja que les distàncies reconegudes pel HillClimbing són més petites que les recorregudes pel Simulated Annealing, pensem que recorre més distància perquè a l'hora de calcular el cost de transport amb la fórmula que ens indiquen a la pràctica, si una furgoneta transporta 0 bicicletes, el cost de transport serà 0, pel fet que la divisió és entera, tal com expliquem en els comentaris addicionals.

Amb aquesta taula i aquests gràfics podem concloure que el Simulated Annealing és propens a donar millors beneficis, però això suposa un cost addicional en el temps d'execució perquè aquest explora més espai que l'algoritme Hill Climbing. Malgrat tot, no ens podem fiar del resultat del simulated annealing, ja que és realment aleatori i encara que en mitjana ens trobi millors resultats és possible que algún dels resultats sigui molt pitjor comparat amb el Hill Climbing, el qual sempre troba el mateix subòptim.

Comentaris addicionals:

No hem pogut realitzar gràfics dels quilòmetres totals recorreguts, ja que tal com hem implementat el codi no teníem cap opció per realitzar-los.

Pel simulated annealing utilitzem un nombre d'iteracions major a l'escollit en l'experiment 3 perquè per la segona heurística hem d'explorar més estats per acabar escollint l'òptim.

Podem observar que en el simulated Annealing la distància recorreguda és més gran això es deu al fet que el cost de transport donat per l'enunciat del problema té una divisió entera com podem veure aquí: ((nb + 9) div 10), on div és la divisió entera.

6. 6. EXPERIMENT 6

Dado el escenario del primer apartado, estimad cual es aproximadamente el número de furgonetas que son necesarias para obtener la mejor solución. Para ello empezad con 5 furgonetas e id aumentándolas de 5 en 5 hasta que no haya una mejora significativa.

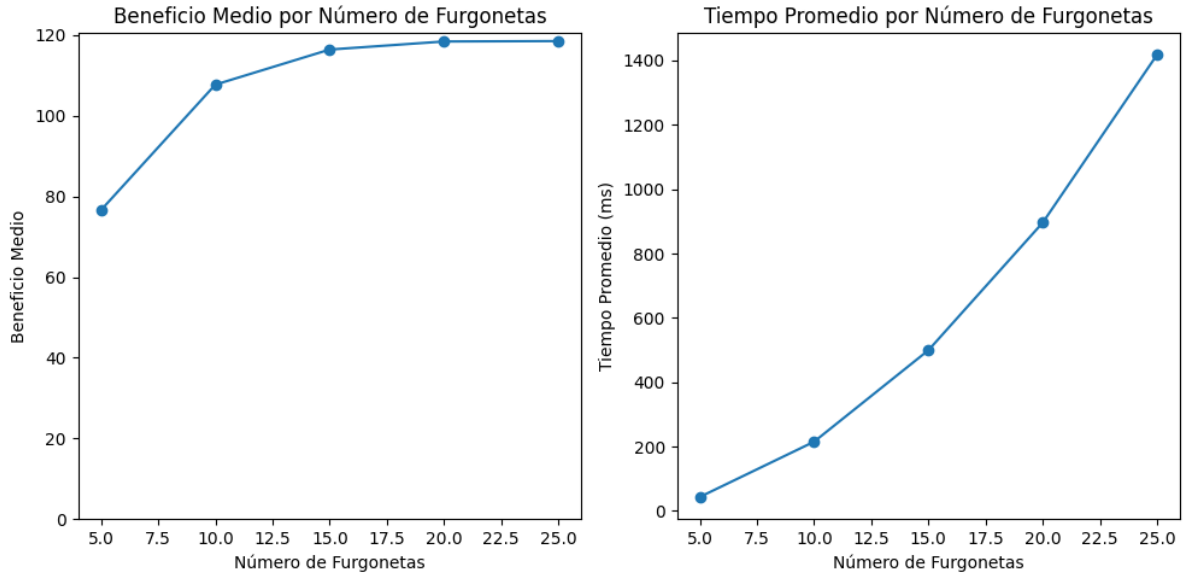
Per aquest experiment anirem augmentant de 5 en 5 el nombre de furgonetes i veurem els beneficis que obtenim, sense tenir en compte el cost de transport de furgonetes.El número de furgonetes que tenim disponibles és un factor important. Depenent del nombre de furgonetes l'algorisme trobarà solucions diferents. A més, també esperem que com més furgonetes hi hagi, més gran serà el factor de ramificació i, per tant, més tardarà a trobar una solució.Per aquest experiment veurem com afecta el nombre de furgonetes en la solució.Per dur a terme aquest experiment anirem sumant furgonetes de 5 en 5. Per a cada configuració de furgonetes, farem 10 iteracions en les quals canviarem la llavor en cada iteració. Llavors farem una mitja del benefici total obtingut i del temps d'execució.

OBSERVACIONS	El nombre de furgonetes pot afectar el benefici i temps d'execució del programa
PLANTEJAMENT	Escollim diferents nombres de furgonetes i observem els seus temps d'execució i beneficis
HIPÒTESIS	Tant el temps com el benefici creixen linealment a mesura que augmentem el nombre de furgonetes.
MÈTODE	● Triarem 10 llavors aleatòries per a cada número de furgonetes ● Executarem 10 vegades per a cada mida (una per cada llavor diferent) i farem mitges dels resultats ● Experimentarem amb problemes de 5, 10, 15, 20 i 25 furgonetes ● Farem servir l'algorisme de Hill Climbing amb inicialització avariciosa ● Mesurarem el temps de còmput i el benefici per realitzar la comparació.

Abans de començar els resultats esperats són que a mesura que augmentem el nombre de furgonetes, el temps d'execució vagi augmentant. Respecte al benefici, també creiem que anirà augmentant, però arribarà un punt en què la millora serà insignificant.

Mitjana	5 furgonetes	10 furgonetes	15 furgonetes	20 furgonetes	25 furgonetes
Benefici	76.6	107.7	116.1	118.0	117.5
Temps	41.53 ms	211.41 ms	553.41 ms	1023.25 ms	1529.8 ms

Com podem veure en la taula, aquests són els valors que hem obtingut executant el programa amb diferents nombres de furgonetes. A continuació, veurem aquestes mateixes dades amb un gràfica per tal de millorar la seva interpretació:



En els gràfics, podem veure algunes tendències. Comencem parlant de la gràfica del temps. Com podem veure, a mesura que augmentem el nombre de furgonetes, també augmentem el temps d'execució. Això té sentit, ja que el fet d'augmentar el nombre de furgonetes fa que el factor de ramificació sigui més elevat i, per tant, el temps també és més elevat.
Per altra banda, la gràfica del benefici és una mica més complexa. Podem veure que amb 5 furgonetes augmenta molt el benefici. De 5 a 10 també augmenta considerablement. A partir d'aquest nombre de furgonetes, la gràfica comença a agafar forma d'asímptota horitzontal. Això vol dir que per més que tinguem més furgonetes disponibles, no millorarem el benefici. Si ens parem a pensar, això té sentit, ja que si mirem com són les estacions, veiem que tenim un total de 25 estacions. A més, els operadors juntament amb la heurística fan que les furgonetes visitin primer aquelles estacions on sobren més bicicletes, i per això les primeres iteracions tenen un creixement més gran. A partir d'un punt, en el que les estacions que queden els hi sobren molt poques bicis o inclús no els n'hi sobren, no millora el benefici.

6. 7. EXPERIMENT ESPECIAL

Per realitzar aquest experiment hem utilitzat els operadors que hem escollit arran dels resultats obtinguts en l'experiment 1, aquests són:

- Estacio_origen
- Estacio_desti
- Afegir_estacio_desti
- Intercanviar_destins
- Eliminar_estacio
- Pivotar_destins

També generem la solució inicial amb l'estratègia que ens dona millors resultats, la generació de l'estat inicial 'greedy'.

El resultat per l'escenari amb 25 estacions, 1250 bicicletes i 5 furgonetes és el següent:

Furgoneta 1
ID Estació Origen: 14 - Bicicletes agafades: 25
ID Estació Destí 1: 23 - Bicicletes descarregades: 25 - Recorregut: 6.7km
Distancia total recorreguda de la furgoneta: 6.7km

Furgoneta 2
ID Estació Origen: 24 - Bicicletes agafades: 19
ID Estació Destí 1: 9 - Bicicletes descarregades: 19 - Recorregut: 4.2km
Distancia total recorreguda de la furgoneta: 4.2km

Furgoneta 3
ID Estació Origen: 17 - Bicicletes agafades: 18
ID Estació Destí 1: 4 - Bicicletes descarregades: 18 - Recorregut: 6.4km
Distancia total recorreguda de la furgoneta: 6.4km

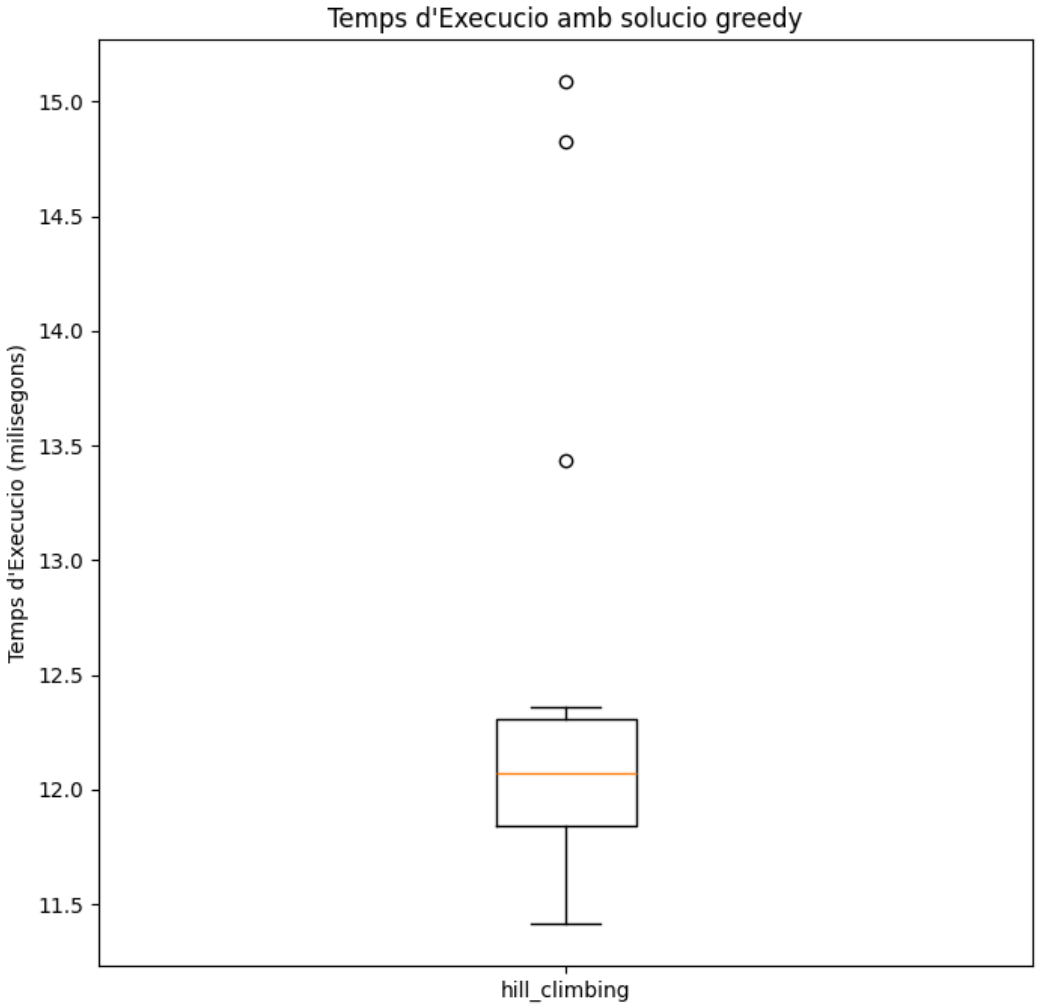
Furgoneta 4
ID Estació Origen: 20 - Bicicletes agafades: 18
ID Estació Destí 1: 1 - Bicicletes descarregades: 18 - Recorregut: 9.5km
Distancia total recorreguda de la furgoneta: 9.5km

Furgoneta 5
ID Estació Origen: 16 - Bicicletes agafades: 16
ID Estació Destí 1: 5 - Bicicletes descarregades: 16 - Recorregut: 6.5km
Distancia total recorreguda de la furgoneta: 6.5km

Benefici total: 95€
Longitud total del recorregut de les furgonetes: 33.3 km

Temps d'execució: 12.2 milisegons

A més, hem executat el codi 15 vegades i hem generat un boxplot amb els temps d'execució en milisegons:



Com podem observar en el gràfic l'algoritme triga aproximadament 12 mil·lisegons de mitja en trobar la millor solució. Podem veure que en comptades ocasions el temps d'execució augmenta alguns mil·lisegons.

Cal tenir en compte que el temps d'execució pot variar entre ordinadors degut a la seva arquitectura del processador i altres factors. En executar el codi en altres ordinadors hem trobat temps d'execució d'uns 60 mil·lisegons, i amb un altre ordinador de fins a 190 mil·lisegons.