Pràctica Cerca Local

Algorismes Bàsics per la Intel·ligència Artificial

Joan Bennàssar Martín
Joan Bernaus Casadesús
Jaume Mora i Ladària

GRUP 11 - ABIA
Grau en Intel·ligència Artificial
Universitat Politècnica de Catalunya
Barcelona, a octubre de 2023

ÍNDEX

Introducció

En aquest document podem trobar la documentació referent a les descripcions i justificacions de la implementació que hem realitzat així com dels resultats que hem obtingut a la pràctica de Cerca Local feta per en Joan Bennàssar, en Joan Bernaus i en Jaume Mora per a l'assignatura d'Algoritmes Bàsics de la Intel·ligència Artificial.

L'objectiu principal d'aquesta primera pràctica de laboratori és familiaritzar-se amb diferents tècniques de resolució de problemes basades en cerca local. Concretament veurem i utilitzarem les tècniques Hill Climbing i Simulated Annealing, dues tècniques d'optimització matemàtica molt populars i utilitzades en el món de la intel·ligència artificial.

Per fer-ho ens ajudarem de la llibreria de Python AIMA en la qual trobem les dues tècniques esmentades anteriorment ja programades. Com veurem, li haurem de passar a aquests algoritmes el nostre estat actual del problema, els seus successors i heurística per tal que ens retorni el nou estat que millor s'ajusta a la solució desitjada.

Partint d'aquesta implementació realitzarem diferents experiments, en els quals durem a terme un anàlisi exhaustiu de la nostra implementació, així com la realització de comparacions per observar les diferències entre Hill Climbing i Simulated Annealing.

El Problema

El problema que es planteja consisteix a optimitzar la distribució de bicicletes en una xarxa d'estacions de Bicing en una ciutat virtual de 10x10 quilòmetres. L'objectiu és millorar la gestió de la redistribució de les bicicletes entre les estacions a través de l'ús de furgonetes. Per resoldre aquest problema, Bicing proporciona informació sobre la demanda i l'oferta de bicicletes a cada estació durant una hora específica del dia. Això inclou:

1. L'estimació de les bicicletes que no es faran servir en una estació durant una hora i que es podrien traslladar a una altra estació.
2. L'estimació del nombre de bicicletes que s'espera trobar en una estació a l'hora següent.
3. L'estimació del nombre de bicicletes que hauria d'haver-hi en una estació a l'hora següent per satisfer la demanda prevista.

A més, es proporciona una funció de distància per calcular la distància entre dues ubicacions a la ciutat. Aquesta funció s'utilitza per determinar les distàncies entre les estacions:

$$d(i, j) = |ix - jx| + |iy - jy|$$

També hi ha informació sobre una flota de furgonetes:

- Hi ha un total de F furgonetes disponibles, i cadascuna pot transportar fins a 30 bicicletes.
- Cada furgoneta pot realitzar un sol viatge en una hora, transportant bicicletes des d'una estació a, com a màxim, a dues estacions.

L'objectiu és optimitzar el transport de bicicletes perquè el nombre de bicicletes a cada estació s'apropi a la demanda prevista. Bicing pagaria 1 euro per cada bicicleta que es transporti i que s'apropi a la demanda prevista d'una estació, però cobraria 1 euro per cada bicicleta transportada que allunyi una estació de la seva demanda prevista. A més a més, el transport de bicicletes a les furgonetes té un cost addicional basat en el cost de la benzina, que varia en funció de la distància que recorre la furgoneta i el nombre de bicicletes transportades. La funció que podem veure a continuació representa el cost en euros per quilòmetre recorregut suposant que nb representa el nombre de bicicletes transportades:

$$Cost = (nb + 9) / 10$$

Per resoldre aquest problema, és necessari desenvolupar un algorisme d'optimització que determini quantes furgonetes s'han de fer servir i com distribuir les bicicletes entre les estacions de manera que es minimitzi el cost total i es compleixin les demandes previstes. Això implica calcular les distàncies entre les estacions, seleccionar les estacions d'origen i destí per a cada furgoneta i decidir quantes bicicletes es transportaran.

Preguntes inicials

1. **Quins elements intervenen en el problema?**

- Les estacions de Bicing repartides per la ciutat.
- El nombre de bicicletes que es troben en aquestes estacions.
- La flota de furgonetes amb una capacitat màxima de 30 bicicletes cadascuna.

2. **Quin és l'espai de cerca?**

   L'espai de cerca està format per totes les possibles combinacions de moviments de bicicletes entre estacions fent anar les furgonetes disponibles, respectant les restriccions imposades.

3. **Quina mida té l'espai de cerca?**

   La mida de l'espai de cerca pot ser molt gran, depenent del nombre d'estacions i de furgonetes. En el pitjor cas, podria ser totes les combinacions possibles de distribucions de bicicletes entre totes les estacions utilitzant totes les furgonetes disponibles.

4. **Què és un estat inicial?**

   L'estat inicial és la distribució de bicicletes en totes les estacions abans de realitzar qualsevol moviment amb les furgonetes. També pot ser la distribució de furgonetes en cada estació i a les estacions a les que ha d'anar. Podria ser també el mapa inicial d'estacions sense cap furgoneta assignada a cap estació.

5. **Quines condicions compleix un estat final?**

   -Les furgonetes no han excedit la seva capacitat.
   -Les furgonetes no han visitat més de dues estacions.
   -Les bicicletes s'han carregat només en l'estació d'origen de la furgoneta.
   -La distribució final de bicicletes s'apropa el màxim possible a la demanda prevista per cada estació.
   -El benefici és el més alt possible

6. **Quins operadors permeten modificar els estats i quin factor de ramificació tenen?**

   1. Intercanviar Estacions
   2. Eliminar Segona Estació
   3. Afegir Furgoneta
   4. Esborrar Furgoneta
   5. Canviar Estació d'Origen
   6. Carregar Dues Bicicletes Més i Carregar Dues Bicicletes Menys
   7. Transferència de Bicicletes entre Estacions

   Aquests són els operadors plantejats, en l'apartat **3. Descripció i justificació dels operadors escollits** s'explica el funcionament d'aquests operadors i els seus factors de ramificació.

Representació del Problema

Per representar el problema hem creat la classe estat, així com una classe furgonetes que a continuació expliquem detalladament:

Classe furgonetes

La classe "Furgonetes" ha estat dissenyada per representar l'estat d'una furgoneta utilitzada en el sistema de Bicing. Aquesta classe té diversos atributs i mètodes que ens ajuden a calcular diversos aspectes importants en la redistribució de bicicletes entre les estacions de Bicing. A continuació, expliquem cadascun dels seus atributs i la seva justificació:

- **`origen`**: Aquest atribut indica l'estació des de la qual la furgoneta carrega les bicicletes. És fonamental conèixer l'origen per calcular els costos i els ingressos relacionats amb el moviment de bicicletes.

- **`primera_est`**: Aquest atribut representa la primera estació de destí a la qual la furgoneta pot traslladar les bicicletes carregades.

- **`segona_est`**: Aquest atribut indica la segona estació de destí, si n'hi ha. Algunes vegades, una furgoneta pot traslladar bicicletes a dues estacions diferents en un sol viatge. Aquest atribut permet gestionar aquest escenari.

- **`bicis_carregades`**: Aquest atribut especifica el nombre de bicicletes que es carreguen en l'estació d'origen. És fonamental per calcular els costos, els ingressos i les pèrdues relacionades amb el transport de bicicletes.

- **`bicis_primera`**: Aquest atribut indica el nombre de bicicletes que es deixen a la primera estació de destí. Aquesta informació és important per calcular els ingressos i les pèrdues relacionats amb la primera estació.

- **`bicis_segona`**: Aquest atribut representa el nombre de bicicletes que es deixen a la segona estació de destí, si hi ha una segona estació. Com en el cas de la primera estació, aquest atribut és essencial per calcular els ingressos i les pèrdues associades a la segona estació.

Cada un d'aquests atributs és crític per a la gestió efectiva del transport de bicicletes entre estacions i permet calcular els costos, els ingressos i les pèrdues associades a cada viatge de la furgoneta. Les funcions per calcular els costos, els ingressos i les pèrdues de cada furgoneta també estan definides en aquesta classe.

Classe estat

La classe "Estat" permet representar el problema en la seva totalitat. És com si fos un foto de l'estat en el qual ens trobem en cada moment, és a dir, un instant de temps en el qual les furgonetes estan distribuïdes d'una certa manera pel la ciutat (mapa). A continuació podem trobar una descripció dels atributs de la classe "Estat":

- **`Contador_Estats`**: Aquest és un atribut de classe (variable estàtica) que s'utilitza per comptar el nombre d'instàncies de la classe "Estat" que s'han creat. Això ens és útil per fer un seguiment de les instàncies que es creen i facilitar la seva identificació.

- **`params`**: Aquest atribut és instància de la classe *Paràmetres* que conté, com el nom indica, aquells paràmetres inicials que són iguals en tots els estats i que són indicats per l'usuari. Aquests paràmetres inclouen el número d'estacions, el número de bicis, la llavor se creació de les estacions, el número màxim de furgonetes i el màxim de bicicletes que pot transportar cada furgoneta (30). Aquest atribut es fa servir per emmagatzemar la informació sobre aquests paràmetres que defineixen l'estat del sistema.

- **`flota`**: Aquest atribut és una llista d'instàncies de la classe "Furgonetes." Aquest atribut emmagatzema les dades de les furgonetes actuals, permetent un accés fàcil per a les operacions i càlculs relacionats amb la gestió de la flota.

- **`estacions`**: Aquest atribut és una instància de la classe *Estaciones,* dins de l'arxiu *abia_bicing,* que conté la informació sobre les estacions. Les estacions són elements importants del sistema, i aquest atribut emmagatzema les dades de les estacions, com ara la seva ubicació i les bicicletes disponibles.

- **`estacions_origen`**: Aquest atribut és un conjunt (set) que emmagatzema les estacions que s'han marcat com a "origen" dins de l'aplicació. Aquest atribut ens és útil per fer un seguiment de les estacions que es consideren com a orígens per a les furgonetes, i així evitar que es seleccionin com a destinacions.

Generació de la solució inicial

Per el problema del Bicing hem pensat en crear tres estats incials diferents els quals a continuació descriurem i justificarem detalladament. Hem pensat que tres estats era la justa mesura per poder tenir punts de partida molt diferents entre ells que ens ajudessin a resoldre el nostre problema.

El tres estats que hem creat els hem anomenat al codi de la següent manera:

1. Estat inicial 0: és un estat inicial que es genera totalment buit, sense cap furgoneta i sense estacions. En altres paraules, és el que coneixem com a estat inicial nul. De totes maneres, és útil quan es vol iniciar l'algorisme sense cap precondició o influència inicial. La solució, per això, es troba molt lluny d'una solució ideal per al problema o que s'hi s'apropi.

2. Estat inicial 1: es genera un estat inicial on es crea una flota amb el nombre màxim de furgonetes. Les diferents instàncies de la classe furgoneta es creen amb unes estacions d'origen, destí 1 i destí 2 en funció de com estan ordenades aquestes estacions, ja que un iterador va recorrent aquestes estacions i les va assignant a cadascuna de les furgonetes. Cada furgoneta es carrega a l'estació d'origen amb el nombre de bicicletes no usades, sense superar les màximes que pot carregar i deixa la meitat d'aquesta càrrega a al primer destí i l'altre meitat al segon. Aquest estat inicial és una forma d'iniciar l'algorisme amb una distribució molt senzilla de les furgonetes. Pot ser útil per avaluacions inicials i proves. És un punt de partida simple i equitatiu. Ens serveix molt bé per fer proves perquè al ser un estat inicial molt dolent és més fàcil veure-hi les millores.

3. Estat inicial 2: en aquest estat inicial les furgonetes es creen en aquelles estacions amb un major excés de bicicletes i porten aquestes a les estacions amb demanda més properes. A cada furgoneta se li assignen dues estacions de descàrrega. A més a més, classifiquem les estacions segons la quantitat de bicis excedents i les furgonetes es carreguen des de les estacions amb els excedents més alts, a aquestes estacions de càrrega les anomenarem estacions origen. Aquest estat inicial és el millor dels tres. És a dir, es comença amb un estat inicial bastant òptim que costa molt de millorar. Això fa que sigui molt més selectiu a l'hora d'aplicar els operadors i ens ajuda a veure quan és que pot millorar.

En resum, cada estratègia d'estat inicial té un propòsit i un impacte diferents. L'elecció de cada estat inicial depèn dels objectius puntuals a l'hora de resoldre el problema. És possible que a vegades intentem resoldre el mateix problema amb més d'un estat inicial diferent per poder compararar resultats i veure en cada cas quin estat inicial funciona millor.

Operadors

Els operadors escollits tenen com a objectiu modificar l'estat del sistema de gestió de bicicletes compartides. Cadascun d'aquests operadors representa una acció que pot ser realitzada per les furgonetes en el context del problema. A continuació, es descriuen i justifiquen els operadors seleccionats:

- **Intercanviar Estacions:** L'operador Intercanviar Estacions permet intercanviar les estacions de destí d'una furgoneta entre elles. Aquesta acció pot ser útil per optimitzar la ruta de descàrrega. A més a més, és possible que pugui millorar l'eficiència dels recorreguts de les furgonetes, permetent-los ajustar les seves rutes si per algun motiu la segona estació d'un principi estava més a prop que la primera.

  Si tenim n estacions i suposem que una furgoneta pot tenir com a destinació qualsevol d'aquestes estacions (excepte l'estació d'origen), llavors el factor de ramificació seria $n-1$ pel primer destí i $n-2$ pel segon, ja que no pot ser ni l'origen ni el primer destí. Però, com estem intercanviant, hi hauria $n(n-1)/2$ intercanvis possibles.

- **Eliminar Segona Estació:** Aquest operador elimina la segona estació de destí d'una furgoneta. Aquest operador és útil quan la segona estació de destí no és necessària o quan anar fins a ella suposa un cost més gran que el benefici que genera. Cada furgoneta que tingui una segona estació de destí pot realitzar aquesta operació. Si tenim f furgonetes amb una segona estació, el factor de ramificació seria f.

- **Afegir Furgoneta:** Afegir Furgoneta crea una nova furgoneta i la càrrega amb bicicletes de l'estació amb més bicicletes no utilitzades. Afegir noves furgonetes pot ser crucial quan la demanda és molt gran i amb la flota actual no s'arriba a tot o quan és millor crear una furgoneta nova que fer que una furgoneta que ja tenim vagi molt lluny. Si hi ha e d'aquestes estacions, llavors el factor de ramificació seria e.

- **Esborrar Furgoneta:** Aquest operador elimina una furgoneta del sistema, retornant les bicicletes carregades a l'estació d'origen. És útil quan es vol reduir la flota o eliminar una furgoneta que ja no és necessària. Si hi ha f furgonetes en el sistema, el factor de ramificació seria f, ja que qualsevol furgoneta podria ser eliminada.

- **Canviar Estació d'Origen:** Aquest operador canvia l'estació d'origen d'una furgoneta per una altra estació que tingui moltes bicicletes en excedent i que no sigui l'origen de cap altra furgoneta. Ens ajuda a distribuir correctament les bicicletes perquè a part de que ens ajuda a millorar la ruta, també fa que la furgoneta comenci a estacions que tenen més excedent i pot carregar més. Si hi ha f furgonetes i e estacions amb bicicletes en excés que no són origen de cap furgoneta, llavors el factor de ramificació seria $f * e$.

- **Carregar Dues Bicicletes Més i Carregar Dues Bicicletes Menys:** Aquests operadors permeten carregar o descarregar dues bicicletes de la furgoneta en una estació determinada. Aquests operadors fan la funció d'ajustament i distribució de les bicicletes una mica aleatòriament. De totes maneres van bé per fer que bicis que potser a vegades no s'havien calculat per portar a una estació, es portin o al revés bicicletes que s'havien de portar, no ho facin. Per a cada furgoneta i per a cada estació, hi ha una possible acció de càrrega o descàrrega. Si hi ha f furgonetes i n estacions, el factor de ramificació seria f * n.

- **Transferència de Bicicletes entre Estacions:** Aquests operadors transfereixen bicicletes que han de ser transportades a les estacions d'una estació a una altra, ja sigui de la primera a la segona estació o viceversa. Aquesta transferència també té la seva part d'aleatorietat que ajuda a vegades millorar la situació. Ja que bicicletes no previstes s'afegeixen a una estació que en teoria no els correspon. El plantejament és similar al dels operadors anteriors. Si una furgoneta té dos destins, pot transferir bicicletes entre aquests dos destins. Si f és el nombre de furgonetes amb dos destins, el factor de ramificació seria f.

En conjunt, aquests operadors ofereixen una gran varietat d'accions que ens ajuden a millorar l'estat del problema. La selecció d'operadors usats es basa en trobar la millor combinació que ens ajudi a obtenir i/o optimitzar la solució que ens permet distribuir correctament les bicicletes mitjançant l'ús d'una flota de furgonetes a un espai de diverses estacions.

Funcions heurístiques

En la resolució d'un problema de cerca local és imprescindible tenir una funció heurística per tal que representi les relacions de qualitat entre les diferents solucions. Aquesta ha d'aproximar la qualitat de la solució, per a fer-ho, ha d'avaluar els diferents elements d'un estat i maximitzar-los o minimitzar-los segons es desitgi.

En aquest cas hem creat dues funcions heurístiques que utilitzen criteris diferents per a avaluar les solucions. D'aquesta manera podrem comparar i veure quin dels dos ens interessa més per a implementar la solució al nostre problema.

Heurística 1: Heurística basada en ingressos i pèrdues

Aquesta heurística està dissenyada per avaluar una solució donada pel benefici resultant d'operar la flota de furgonetes, sense tenir en compte els costos del transport. El càlcul es realitza de la següent manera:

- Se sumen tots els ingressos obtinguts per les furgonetes en la solució actual.
- Es calculen les pèrdues per cada furgoneta que són la diferència entre les bicicletes carregades i les que realment es poden descarregar sense incomplir la demanda.

L'heurística 1 busca maximitzar la diferència entre els ingressos i les pèrdues, ja que aquesta diferència reflecteix el benefici net de la flota de furgonetes. En cas de que les furgonetes puguin atendre la demanda sense tenir grans pèrdues importants, tindrem un bon resultat heurístic.

Heurística 2: Heurística amb costos de gasolina

Aquesta heurística amplia la primera heurística al tenir en compte els costos de gasolina de les furgonetes. El càlcul es realitza de la següent manera:

- Es sumen tots els ingressos obtinguts per les furgonetes en la solució actual.
- Es sumen les pèrdues, tal com es calculen a través de la primera heurística.
- Es calculen els costos de gasolina per cada furgoneta, que depenen de la distància recorreguda per la furgoneta.

L'heurística 2 busca maximitzar la diferència entre els ingressos, les pèrdues i els costos de gasolina. Per tant, l'heurística 2 pot ser una bona opció si, a part de maximitzar els ingressos es volen minimitzar les pèrdues i els costos de la gasolina. Això pot ser especialment important en situacions on els costos de transport són significatius en comparació amb els ingressos.

Experiments

Experiment 1

Determinar el millor conjunt d'operadors que dona resultat per a la primera heurística

Observació i plantejament: L'Experiment 1 busca trobar el conjunt d'operadors que produeix la màxima millora sabent que en aquest cas la funció heurística no té en compte els costos de la gasolina. Aquest experiment es realitza en un escenari amb 25 estacions de bicicletes, un total de 1250 bicicletes i 5 furgonetes. L'algorisme utilitzat en aquest experiment és el Hill Climbing.

Hipòtesi:

- **Hipòtesi Nul·la (Ho):** No existeix un cap conjunt d'operadors que millori el benefici inicial.
- **Hipòtesi Alternativa:** Existeix un conjunt d'operadors que augmenta significativament el benefici inicial en el sistema de bicicletes compartides.

Mètode:

- **Elecció dels subjectes:** 5 furgonetes, 25 estacions i un total de 1250 bicicletes amb una llavor aleatòria.
- **Procediment:** Hem utilitzat Hill Climbing.
- **Variables considerades:** les variables considerades per determinar el conjunt d'operadors ha sigut bàsicament el resultat de l'heurística 1 que es determina amb la diferència entre els beneficis i les pèrdues sense tenir en consideració el cost de la gasolina.

Resultats: Hem vist que l'únic operador que aplica una millora a l'estat incial 0 (que recordem que és aquell estat inicial que s'inicialitza sense furgonetes) és afegir_furgoneta. Sembla que hem tingut algun problema i realment l'unic operador que comporta benefici en el primer cas amb l'heurística més bàsica és afegir_furgoneta. Això se'n va una mica de la nostra hipòtesi inicial que pensàvem que seria que tinguéssim més d'un operador que generés benefici. Ens hem trobat amb que no, que per algun tipus d'error

En **conclusió,** l'experiment 1 no té el resultat que esperàvem. Suposàvem que obtindríem més d'un operador que ens donés benefici però per algun motiu que malauradament encara desconeixem, no ha estat així. L'únic operador que ens comporta un benefici d'entre uns 15€ i 35€ és afegir_furgoneta.

Experiment 2

Determinar quina estratègia de generació inicial de la solució és millor

Observació i plantejament: Amb l'experiment 2 busquem determinar quina és la millor estratègia de generació de la solució és millor. És a dir no sol ha de començar en un valor alt, sino que és més important encara que l'algoritme Hill Climbing no es quedi encallat en màxims locals i no doni el millor resultat. Per tant, volem determinar amb quin estat inicial, obtindrem millor solució final a l'aplicar l'algoritme.

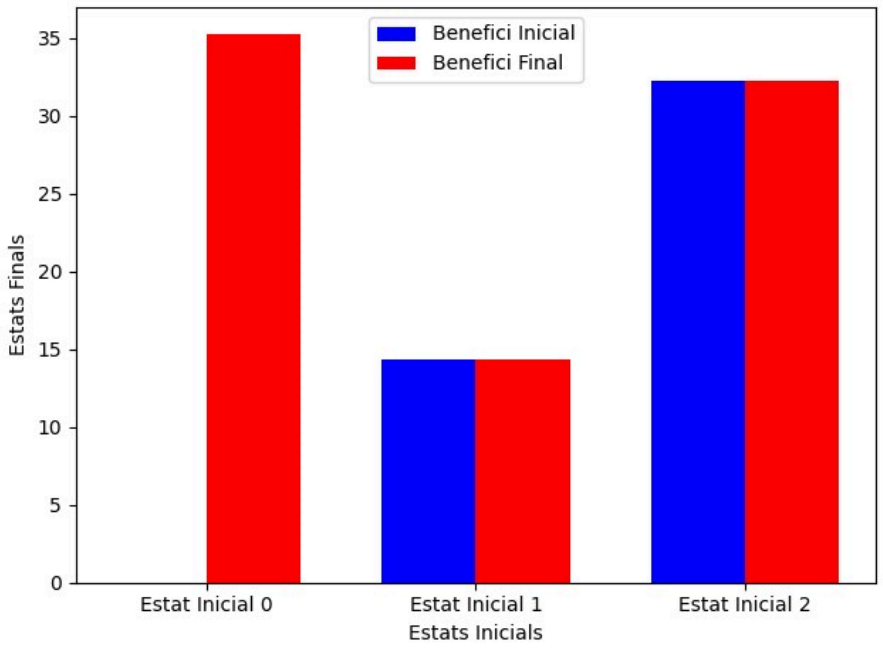
Hipòtesi:

- **Hipòtesi Nul·la (Ho):** No hi ha diferència en els resultats finals obtinguts amb l'algoritme Hill Climbing independentment de l'estratègia de generació inicial de la solució utilitzada.
- **Hipòtesi Alternativa:** Depenent de l'estat inicial que fem anar, obtindrem resultats finals diferents, per tant, hi hauran millors solucions inicials.

Mètode:

- **Elecció dels subjectes:** 5 furgonetes, 25 estacions i un total de 1250 bicicletes amb una llavor aleatòria.
- **Procediment:** Hem executat l'algoritme hill climbing amb l'heuristica1. Hem fet 10 iteracions per estat inicial, cal tenir en compte que la llavor canviarà aleatòriament per cada iteració.
- **Variables considerades:** Les variables considerables són el valor inicial i el final per poder determinar quin és el millor estat inicial.

Resultats: Hem fet un plot amb la mitjana de les 10 iteracions on veiem que l'estat inicial que millor ens funciona és l'estat inicial 0 que ens dona una mitjana del valor final 35,2. Es genera un estat inicial sense furgonetes. A més a més és des de l'únic estat en que es millora.



Conclusió: No podem treure una conclusió clara ja que els estats inicials 1 i 2 no milloren els resultats. El fet de que no ens funcioni bé el programa i no puguem tenir un estat de cerca més ample ens fa dubtar de si realment els dos estats s'estan quedant encallats en un màxim local o si realment el problema és del codi.

Experiment 3 i 5

Relacionats amb el Simmulated Annealing

Malauradament al dedicar tant de temps a intentar trobar l'error del nostre Hill Climbing, no hem implementar correctament el Simmulated Annealing cosa que no ens ha permès fer els experiments relacionats amb aquesta tècnica.

Experiment 4

Observació i plantejament: Hem observat que el temps d'execució de l'algoritme Hill Climbing augmenta amb l'increment del nombre d'estacions, furgonetes i bicicletes. Volem determinar si aquest augment és significatiu i si segueix una tendència previsible.

Hipòtesi:

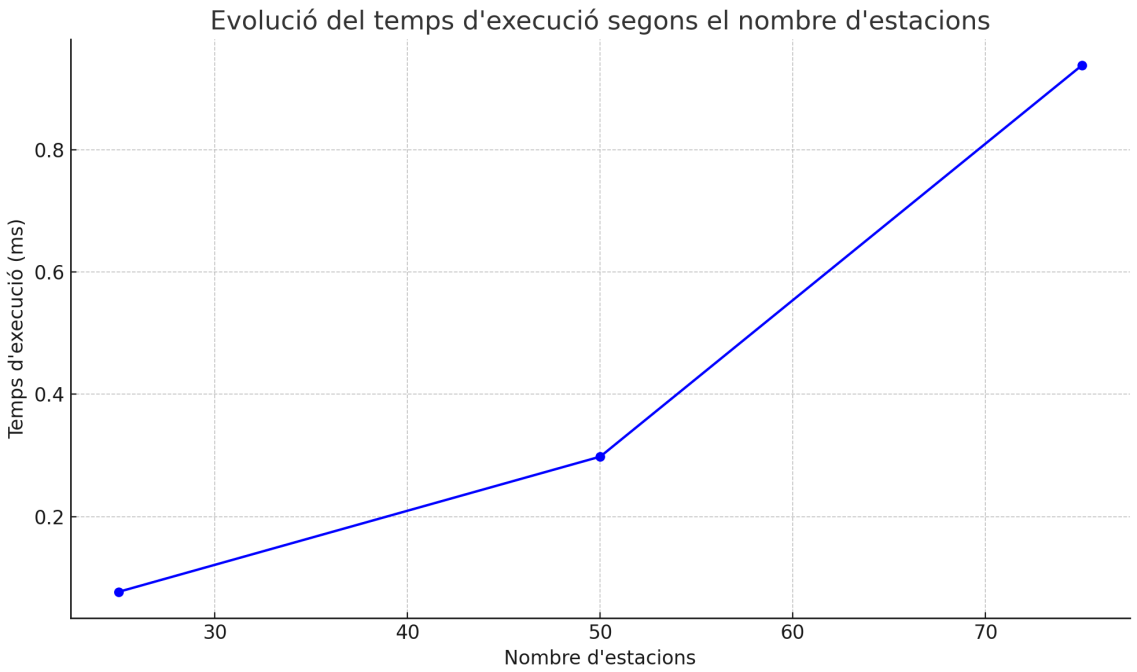
- **Hipòtesi Nul·la (Ho):**L'increment del nombre d'estacions, furgonetes i bicicletes no té un efecte significatiu en el temps d'execució de l'algoritme Hill Climbing.
- **Hipòtesi Alternativa:** L'increment del nombre d'estacions, furgonetes i bicicletes augmenta el temps d'execució de l'algoritme Hill Climbing de manera significativa.

Mètode

- **Elecció dels subjectes:** Hem escollit l'heuristica 1 i el genera estats inicial 0 com hem fixat en liexperiment 1 i 2 a l'igual que l'algoritme hill climbing.
- **Procediment:**Configurem un nombre inicial d'estacions, furgonetes i bicicletes.
  Mesurem el temps d'execució de l'algoritme Hill Climbing.
  Augmentem el nombre d'estacions, furgonetes i bicicletes segons una proporció fixa.
  Repetim els passos 2 i 3 fins a arribar a un límit.

Variables considerades: Nombre d'estacions, furgonetes i bicicletes i el temps d'execució de l'algoritme.

Resultats: Amb 25 estacions, el temps d'execució és de 0.0769 ms.
Amb 50 estacions, el temps d'execució és de 0.2981 ms.
Amb 75 estacions, el temps d'execució és de 0.9382 ms.



Conclusió: Basant-nos en els resultats obtinguts, podem rebutjar l'hipòtesi nul·la i acceptar l'hipòtesi alternativa. Hi ha una relació significativa entre el nombre d'estacions, furgonetes i bicicletes i el temps d'execució de l'algoritme Hill Climbing. La tendència observada indica que l'algoritme pot no ser escalable per a problemes molt grans.

Experiment Especial

Benefici total, longitud total del recorregut i temps total en un escenari específic.

Observació i plantejament: L'experiment especial demana el resultat del benefici total (ingressos - pèrdues), el recorregut de les furgonetes i el temps necessari per trobar la solució. Tot això en un plantejament específic que consta de 25 estacions, 1250 bicicletes i 5 furgonetes. La llavor és 42 sempre, cosa que fa que ens surti sempre el mateix resultat i puguem anar fent proves i comprovacions. Utilitzem tots els operadors mencionats anteriorment i l'heurística 1 (que no té en compte els costos de la gasolina).

Hipòtesis:

- **Hipòtesi Nul·la (Ho):** El benefici total és igual a l'inicial, la longitud de les furgonetes és 0 i el temps de resolució també.
- **Hipòtesi Alternativa:** Tenim un benefici total significatiu respecte al benefici inicial. Les furgonetes fan un recorregut diferent de 0 i el temps de resolució és raonable, no triga ni massa ni massa poc.

Mètode:

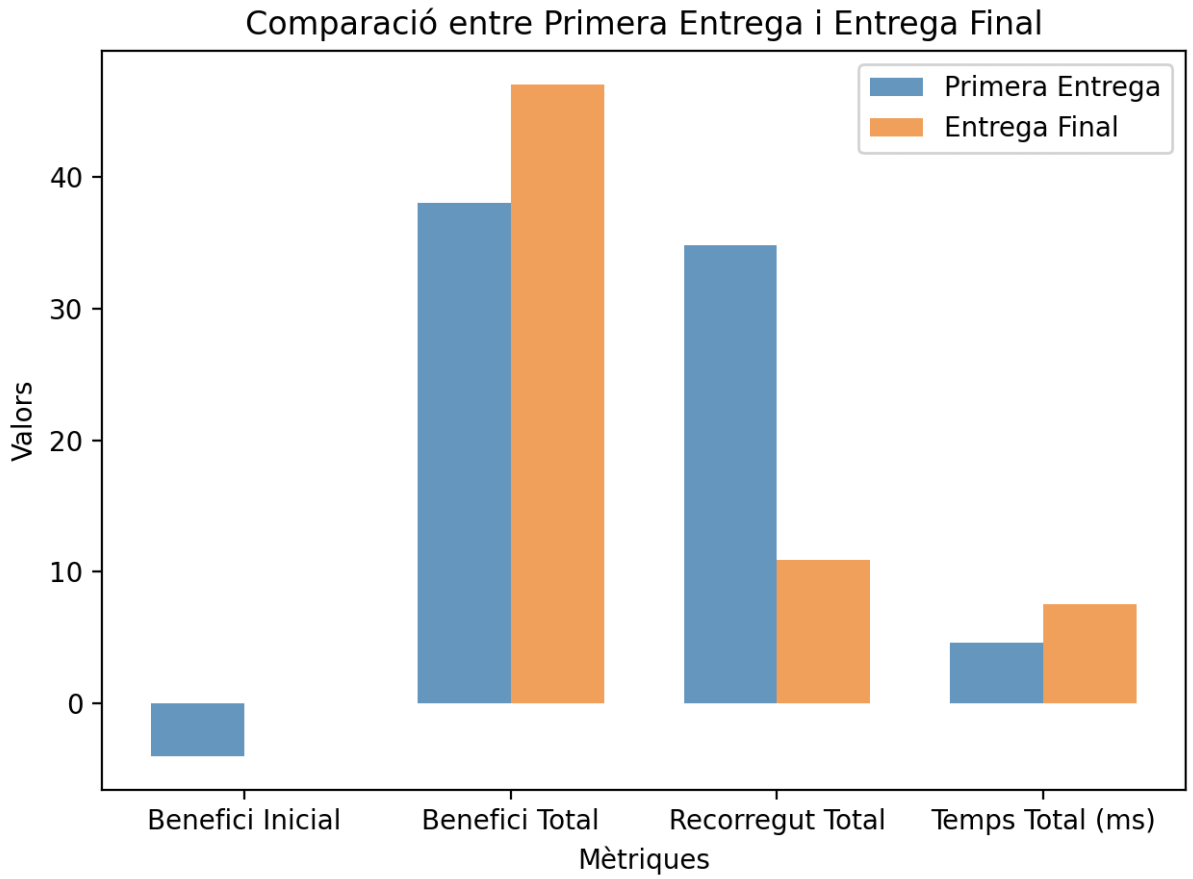
- **Elecció dels subjectes:** Utilitzem un escenari específic de 25 estacions, 1250 bicicletes, 5 furgonetes i una llavor de 42.
- **Procediment:** Apliquem l'algoritme Hill Climbing utilitzant tots els operadors, la generació de l'estat inicial 0 (que no té cap furgoneta al principi) i l'heurística 1 (que no té en compte els costos de la gasolina).

Variables a tenir en compte a l'analitzar el resultat: Benefici total obtingut, longitud total del recorregut de les furgonetes, temps de resolució en mil·lisegons.

Resultats:

Tenim dos resultats diferents. El primer de fa una setmana quan no teníem la pràctica tan avançada i el segon del moment abans de l'entrega de la pràctica. Com que hem aconseguit uns operadors que ens funcionen millor, hem aconseguit molt més benefici amb menys recorregut.

	Primera entrega per correu	El dia de l'entrega final
Benefici Inicial	-4€	0€
Benefici total	38€	47€
Recorregut total	34800m	10900m
Temps total (en ms)	0.04656195640563965ms	0.07526922225952148ms



En conclusió, comparant amb l'experiment realitzat la setmana passada, l'experiment especial mostra una millora en termes de benefici total obtingut i una disminució significativa en la longitud total del recorregut de les furgonetes.

Satisfacció amb el resultat final del treball: Més enllà de les conclusions de cada experiment, creiem que els resultats de la pràctica no reflexen les hores i esforç dipositat en aquesta pràctica. Tot i que el treball en equip opinem que ha estat bo i equitatiu, o bé pel fet de no haver sapigut resoldre els problemes que se'ns presentaven quan tocava o bé per manca d'organització el resultat del treball no ha estat el que esperàvem.