Pràctica de Cerca Local

Planificació de Rutes de Proveïment de Gasolineres

Grau A – Q1 Curs 2025-2026

Departament de Ciències de la Computació
Universitat Politècnica de Catalunya

Anel ?
anel.?@estudiantat.upc.edu

Aleix Pitarch
aleix.pitarch@estudiantat.upc.edu

Joan Solina
joan.solina@estudiantat.upc.edu

26 d'octubre de 2025

Índex

Índex de figures

Índex de taules

Listings

1 Introducció

1.1 Context del problema

Aquesta pràctica se centra en la resolució dun problema de planificació de rutes de proveiment per a una companyia de distribució de combustible mitjançant algorismes de cerca local. La pràctica implica el disseny i la implementació dels elements necessaris pels algorismes Hill Climbing i Simulated Annealing així com la realització dexperiments per avaluar el seu comportament en diferents escenaris. A més, es durà a terme una anàlisi comparativa dels resultats obtinguts amb ambdós algorismes, amb lobjectiu dextreuren conclusions rellevants.

1.2 Objectius

Els objectius principals d'aquesta pràctica són:

- Modelar el problema com un problema de búsqueda local
- Implementar una representació eficient de l'espai d'estats
- Dissenyar operadors adequats per explorar l'espai de solucions
- Definir funcions heurístiques que optimitzin els criteris del problema
- Experimentar amb els algoritmes Hill Climbing i Simulated Annealing
- Analitzar i comparar els resultats obtinguts

2 Definició del problema

2.1 Introducció al problema

El problema que ens ocupa neix d'una necessitat real en el sector de la distribució de combustibles. Imaginem una companyia que opera amb diversos centres de distribució des d'on surten camions cisterna per proveir a les gasolineres. Cada dia, la companyia rep peticions de proveïment de diferents gasolineres i ha de decidir com organitzar els seus recursos per atendre-les de la manera més eficient possible.

Aquest problema presenta diversos factors que cal optimitzar simultàniament. Per una banda, la companyia vol maximitzar els seus ingressos servint el màxim nombre de peticions possibles, especialment aquelles que porten més temps pendents. Per altra banda, cal minimitzar els costos operatius, principalment els derivats dels quilòmetres recorreguts pels camions. Tot això s'ha de fer respectant les restriccions sobre la capacitat dels camions i les hores de treball dels conductors.

2.2 Elements del problema

Per comprendre completament el problema, cal primer identificar tots els elements que hi intervenen i com es relacionen entre ells. Aquestes relacions determinaran posteriorment com modelarem l'espai de cerca.

2.2.1 Centres de distribució i camions cisterna

La companyia disposa de diversos centres de distribució els quals estan situats en unes coordenades específiques dins d'una quadrícula de 100×100 km^2. Cada centre de distribució té assignat un camió cisterna el qual la seva capacitat és exactament el doble de la capacitat d'un dipòsit estàndard de gasolinera. Això significa que en cada viatge, un camió pot omplir com a màxim dos dipòsits, ja sigui de la mateixa gasolinera o de gasolineres diferents.

Cada dia, un camió pot recórrer un màxim de 640 quilòmetres, que corresponen a 8 hores de treball a una velocitat constant de 80 km/h. A més, cada camió pot fer un màxim de 5 viatges diaris, entenent per viatge el recorregut des del centre de distribució fins a les gasolineres assignades i tornada al centre.

2.2.2 Gasolineres i peticions d'abastiment

Cada gasolinera disposa de diversos dipòsits per emmagatzemar combustible, i el seu mode d'operació és seqüencial: utilitzen un dipòsit fins que s'acaba, i aleshores passen al següent. Aquesta simplificació ens permet tractar cada dipòsit com una entitat independent.

Quan un dipòsit d'una gasolinera s'acaba, la gasolinera genera una petició de proveïment a la companyia de distribució. Aquestes peticions van acumulant-se dia rere dia fins que són ateses. És important destacar que una mateixa gasolinera pot tenir múltiples peticions pendents simultàniament si se li han acabat diversos dipòsits, fins a un màxim de 3.

Cada petició porta associat un comptador que indica quants dies porta pendent. Aquest comptador és crucial perquè determina el preu que la companyia cobra per atendre la petició. Una petició nova (0 dies d'espera) es cobra al 102% del preu base, incentivant així un servei ràpid. A mesura que passen els dies sense ser atesa, el preu va disminuint exponencialment segons la fórmula que descriurem més endavant.

2.2.3 Estructura d'un viatge

Un viatge és la unitat bàsica d'operació en aquest problema. Cada viatge segueix sempre la mateixa estructura: el camió surt del centre de distribució amb el dipòsit ple, visita una o dues gasolineres on descarrega combustible, i torna al centre. Aquest cicle es repeteix tantes vegades com sigui necessari dins dels límits permesos.

La flexibilitat en el nombre de gasolineres per viatge (una o dues) dóna lloc a diferents estratègies. Un camió podria optar per fer viatges curts visitant una sola gasolinera propera per minimitzar distàncies, o podria intentar maximitzar l'ús de la seva capacitat visitant dues gasolineres en cada viatge. Aquesta decisió depèn de múltiples factors: la ubicació de les gasolineres, les peticions pendents, i els quilòmetres disponibles.

2.3 Model de càlcul de distàncies

Per calcular la distància entre qualsevol parell de punts (ja siguin centres de distribució o gasolineres), utilitzem la distància de Manhattan:

$$d(A, B) = |A_x - B_x| + |A_y - B_y| \tag{1}$$

on (A_x, A_y) i (B_x, B_y) són les coordenades dels punts A i B respectivament, i $|\cdot|$ denota el valor absolut.

Per calcular la distància total d'un viatge que visita dues gasolineres G_1 i G_2 partint d'un centre de distribució D, calculem:

$$d_{\text{viatge}} = d(D, G_1) + d(G_1, G_2) + d(G_2, D) \tag{2}$$

Si el viatge només visita una gasolinera, la distància es simplifica a:

$$d_{\text{viatge}} = 2 \times d(D, G_1) \tag{3}$$

2.4 Justificació com a problema de cerca local

És important entendre per què aquest problema és adequat per ser resolt amb tècniques de búsqueda local i no amb altres aproximacions.

En primer lloc, l'espai de solucions és exponencialment gran. Amb n camions, p peticions, i la possibilitat de fer fins a 5 viatges per camió, el nombre de possibles assignacions és de l'ordre de $(n \times 5)^p$. Per l'escenari base amb 10 camions i aproximadament 100 peticions, això suposa més de 10^{100} possibles solucions, fent inviable una exploració exhaustiva.

En segon lloc, no tenim un objectiu clarament definit com "arribar a un estat específic", sinó que volem optimitzar una funció de qualitat. Això és característic dels problemes d'optimització i els fa especialment adequats per búsqueda local.

En tercer lloc, existeix una noció natural de "veïnatge"entre solucions. Petits canvis en l'assignació de peticions a camions generen solucions similars amb valors de benefici també similars, cosa que permet una exploració gradual de l'espai.

Finalment, el problema exhibeix una estructura de "paisatge"amb múltiples òptims locals però on moure's en la direcció correcta generalment millora la solució. Aquesta propietat fa que algoritmes com Hill Climbing siguin efectius, tot i que poden quedar atrapats en òptims locals, motivant l'ús d'algoritmes més sofisticats com Simulated Annealing.

Totes aquestes característiques fan que el nostre problema sigui un candidat ideal per aplicar les tècniques de búsqueda local que hem estudiat a classe, i ens permeten explorar experimentalment les seves fortaleses i limitacions.

3 Representació de l'estat

3.1 La importància de la representació

La representació de l'estat és, sens dubte, una de les decisions de disseny més crítiques en qualsevol problema de cerca. Una bona representació no només facilita la implementació dels operadors i la funció heurística, sinó que també determina l'eficiència computacional de tota la solució.

Quan vam començar a pensar en com representar un estat del problema, vam considerar diverses alternatives. Podríem haver representat una solució com una matriu d'assignacions, com una llista de tuples (petició, camió, viatge), o com una estructura

més complexa que inclogués informació redundant per accelerar consultes. Després d'analitzar els pros i contres de cada opció, vam optar per una representació basada en llistes de viatges per camió, que expliquem en detall a continuació.

3.2 Anàlisi previ de l'espai de cerca

Abans de decidir com representar un estat, és fonamental comprendre la magnitud de l'espai amb el qual treballem. Aquesta anàlisi ens ajudarà a justificar les decisions que prendrem.

3.2.1 Càlcul teòric de la mida de l'espai

Considerem un escenari amb n camions, p peticions a servir, i la restricció que cada camió pot fer màxim v viatges. Per cada petició, hem de decidir:

1. Si la servim o no
2. En cas afirmatiu, quin camió l'atendrà
3. A quin viatge d'aquest camió s'assignarà
4. En quin ordre dins del viatge

Només considerant les assignacions de peticions a camions (ignorant l'ordre i l'agrupació en viatges), tenim $(n+1)^p$ possibilitats: cada petició pot ser assignada a qualsevol dels n camions o quedar sense servir. Per al nostre escenari base amb 10 camions i aproximadament 100 peticions, això ja suposa $11^{100} \approx 10^{104}$ possibilitats.

3.3 Estructura de dades escollida

Després d'analitzar diverses alternatives, hem optat per una representació basada en llistes de viatges organitzades per camió. Aquesta decisió es fonamenta en diversos raonaments que exposem a continuació.

3.3.1 Components de l'estat

L'estat del nostre problema es compon de diverses parts que hem de distingir clarament entre dades estàtiques (compartides per tots els estats) i dades dinàmiques (específiques de cada estat).

Dades estàtiques: Les dades estàtiques són aquelles que no varien durant la cerca. Aquestes es declaren com a variables de classe (static en Java) i es comparteixen entre totes les instàncies d'estats:

```
1 public class PracticaBoard {
2     // Informacio del problema (compartida per tots els estats
      )
3     private static Gasolineras gasolineres;
4     private static CentrosDistribucion centres;
```

```
5       private static Map<Integer, Peticio> peticions;
6
7       // Constants del problema
8       private static final int KM_MAX = 640;
9       private static final int VIATGES_MAX = 5;
10      private static final double COST_KM = 2.0;
11      private static final double PREU_BASE = 1000.0;
12
13      // ...
14  }
```

Listing 1: Dades estàtiques de l'estat

Aquesta separació és crucial per l'eficiència espacial. Si tinguéssim 10.000 estats en memòria simultàniament (cosa que pot passar durant la cerca), duplicar aquesta informació en cada estat seria extremadament ineficient. Amb l'aproximació estàtica, aquestes dades s'emmagatzemen una sola vegada.

Dades dinàmiques: Les dades dinàmiques representen la solució específica d'aquest estat i varien d'un estat a un altre:

```
1  public class PracticaBoard {
2      // ...
3
4      // Assignacions especifiques d'aquest estat
5      private List<List<Viatge>> assignacions;  // viatges per
           camio
6      private Set<Integer> peticionsServides;
7
8      // Metriques pre-calculades (per eficiencia)
9      private double beneficiTotal;
10     private int[] kmPerCamio;
11     private int[] viatgesPerCamio;
12 }
```

Listing 2: Dades dinàmiques de l'estat

La llista `assignacions` és el nucli de la representació. És una llista de llistes: per a cada camió (índex de la llista exterior) tenim una llista dels seus viatges. Aquesta estructura bidimensional ens permet accedir directament als viatges d'un camió específic en temps constant.

El conjunt `peticionsServides` ens permet verificar ràpidament si una petició ja està assignada, operació que fem freqüentment. Utilitzar un conjunt (HashSet) en lloc d'una llista redueix la complexitat d'aquesta verificació de $O(p)$ a $O(1)$.

Les mètriques pre-calculades són una optimització important. En lloc de recalcular el benefici total cada vegada que el necessitem, el calculem una vegada després de cada modificació i el guardem. Això és especialment útil per a Hill Climbing, que necessita comparar el valor heurístic de múltiples estats successors.

3.4 Complexitat espacial i temporal

Analitzem formalment la complexitat de la nostra representació per validar que és eficient.

3.4.1 Complexitat espacial

Per un estat amb n camions i p_s peticions servides:

- **Assignacions**: Cada camió té com a màxim 5 viatges amb 2 peticions cada un $\rightarrow O(n \cdot 10) = O(n)$
- **Peticions servides**: Un conjunt de mida $p_s \rightarrow O(p_s)$
- **Mètriques**: Arrays de mida $n \rightarrow O(n)$
- **Total**: $O(n + p_s)$

En la pràctica, amb $n = 10$ i $p_s \approx 90$, cada estat ocupa aproximadament uns pocs kilobytes de memòria, cosa que permet tenir milers d'estats en memòria simultàniament sense problemes.

3.5 Alternatives considerades i descartades

Abans d'arribar a la representació final, vam considerar i descartar diverses alternatives.

3.5.1 Representació com a llista plana de tuples

Una alternativa seria representar l'estat com una llista de tuples (petició, camió, viatge, posició). Aquesta representació és molt flexible però té desavantatges:

- Dificulta l'accés a tots els viatges d'un camió
- Requereix ordenar o filtrar la llista per moltes operacions
- No encapsula la lògica de viatges
- Més propensa a errors (tuples amb 4 components)

3.5.2 Representació amb grafs

Podríem modelar les assignacions com un graf on els nodes són gasolineres i els arcs representen l'ordre de visita. Aquesta representació, tot i ser elegant matemàticament, és massa complexa per les nostres necessitats:

- Sobredimensionada per a viatges amb màxim 2 gasolineres
- Dificulta les modificacions locals
- Requereix llibreries addicionals
- Més costosa en memòria

3.5.3 Representació amb informació redundant

Una altra opció seria mantenir múltiples estructures de dades (per exemple, tant una llista de viatges com un mapa de peticions a viatges). Això acceleraria algunes consultes però:

- Augmenta significativament l'ús de memòria
- Requereix mantenir la coherència entre estructures
- El guany en velocitat no compensa la complexitat afegida

4 Estratègies de generació de la solució inicial

4.1 Imporsdfasdfasftància de la solució inicial

La solució inicial té un impacte significatiu en la búsqueda local:

- **Temps de convergència**: Una bona solució inicial redueix el nombre de passos necessaris
- **Qualitat de la solució final**: Pot influir en quin òptim local s'arriba
- **Cost de generació**: Cal balancejar qualitat vs temps de càlcul

4.2 Estratègies proposades

4.2.1 Estratègia 1: Solució buida

Descripció: No s'assigna cap petició inicialment. Tots els camions comencen sense viatges.

```
public PracticaBoard() {
    assignacions = new ArrayList<>();
    for (int i = 0; i < NUM_CAMIONS; i++) {
        assignacions.add(new ArrayList<>());
    }
    peticionsServides = new HashSet<>();
    beneficiTotal = calcularBenefici();
}
```

Listing 3: Implementació de solució buida

Avantatges:

- Temps de generació: $O(1)$ - instantani
- Implementació trivial
- Sempre compleix les restriccions

Inconvenients:

- Benefici inicial: 0 (pèssim)
- Requereix molts passos per arribar a una bona solució
- Pot quedar atrapada en òptims locals primerencs

4.2.2 Estratègia 2: Assignació aleatòria

Descripció: S'assignen peticions aleatòriament als camions fins que no es poden afegir més sense violar restriccions.

Algorithm 1 Generació aleatòria de solució inicial

```
peticions ← barrejar_aleatoriament(totes_peticions)
for cada p in peticions do
    camio ← aleatori(0, NUM_CAMIONS-1)
    nou_estat ← afegir_peticio(p, camio)
    if nou_estat.esValid() then
        estat ← nou_estat
    end if
end for
return estat
```

Avantatges:

- Temps de generació: $O(p)$ - ràpid
- Genera solucions diferents en cada execució
- Sol complir restriccions (amb validació)

Inconvenients:

- Qualitat variable segons l'aleatorietat
- No garanteix aprofitar bé la capacitat dels camions
- Pot deixar moltes peticions sense servir

4.2.3 Estratègia 3: Assignació avariciosa per benefici

Descripció: S'assignen les peticions en ordre de benefici decreixent, assignant cada petició al camió més proper que pugui servir-la.

Algorithm 2 Generació avariciosa per benefici

```
1: peticions ← ordenar_per_benefici_decreixent()
2: for cada p in peticions do
3:     millor_camio ← NULL
4:     min_dist ← ∞
5:     for cada camio in camions do
6:         dist ← distancia(camio, gasolinera_de(p))
7:         if dist < min_dist AND pot_afegir(camio, p) then
8:             millor_camio ← camio
9:             min_dist ← dist
10:        end if
11:    end for
12:    if millor_camio ≠ NULL then
13:        afegir_peticio(p, millor_camio)
14:    end if
15: end for
```

Càlcul del benefici d'una petició:

```java
private double calcularBeneficiPeticio(Peticio p) {
    int dies = p.getDies();
    double preu;

    if (dies == 0) {
        preu = 1000 * 1.02;
    } else {
        preu = 1000 * (1 - Math.pow(2, dies) / 100.0);
    }
    return preu;
}
```

Listing 4: Càlcul del benefici

Avantatges:

- Prioritza les peticions més rendibles
- Minimitza distàncies (menys cost)
- Sol generar solucions acceptables
- Temps de generació: $O(p \times n \times \log p)$ - raonable

Inconvenients:

- Pot deixar camions infrautilitzats
- No considera l'agrupació òptima en viatges
- Sempre genera la mateixa solució (determinista)

4.2.4 Estratègia 4: Assignació avariciosa per proximitat geogràfica

Descripció: Agrupa les peticions per zones geogràfiques i assigna cada zona al camió més proper.

Algorithm 3 Generació avariciosa per proximitat

```
clusters ← clustering_geografic(gasolineres, NUM_CAMIONS)
for cada cluster in clusters do
    camio ← camio_mes_proper(cluster)
    peticions ← peticions_del_cluster(cluster)
    ordenar_per_ruta_optima(peticions)
    for cada p in peticions do
        if pot_afegir(camio, p) then
            afegir_peticio(p, camio)
        end if
    end for
end for
```

Avantatges:

- Minimitza distàncies totals
- Aprofita millor l'agrupació geogràfica
- Genera viatges més eficients

Inconvenients:

- Complexitat de càlcul més alta: $O(p^2)$
- Requereix algoritme de clustering
- No considera el benefici de les peticions

4.3 Comparació teòrica de les estratègies

Estratègia	Complexitat temporal	Benefici inicial	Distància inicial	Peticions servides
Buida	$O(1)$	Mínim	Mínima	0
Aleatòria	$O(p)$	Variable	Variable	Variable
Per benefici	$O(p \times n \times \log p)$	Alt	Baixa	Alta
Per proximitat	$O(p^2)$	Mitjà	Mínima	Mitjana-Alta

Taula 1: Comparació teòrica de les estratègies

4.4 Estratègies escollides per experimentació

Hem escollit experimentar amb les següents estratègies:

1. **Estratègia buida** (E1): Com a línia base
2. **Estratègia avariciosa per benefici** (E3): Com a solució elaborada

Justificació:

- E1 representa el pitjor cas i és útil per veure la capacitat de millora de l'algoritme
- E3 equilibra qualitat i temps de generació
- Ambdues són deterministes, facilitant la comparació
- L'aleatòria (E2) no és reproducible
- La geogràfica (E4) és massa costosa per la millora marginal esperada

4.5 Implementació de la solució escollida

```
public static PracticaBoard generarSolucioGreedy() {
    PracticaBoard board = new PracticaBoard();

    // Ordenar peticions per benefici decreixent
    List<Map.Entry<Integer, Peticio>> peticionsOrdenades =
        new ArrayList<>(peticions.entrySet());
    peticionsOrdenades.sort((a, b) ->
        Double.compare(
            calcularBeneficiPeticio(b.getValue()),
            calcularBeneficiPeticio(a.getValue())
        )
    );

    // Assignar cada petició al millor camió
    for (Map.Entry<Integer, Peticio> entry :
       peticionsOrdenades) {
        int idPeticio = entry.getKey();
        Peticio peticio = entry.getValue();

        int millorCamio = trobarMillorCamio(board, peticio);

        if (millorCamio != -1) {
            board.afegirPeticio(idPeticio, millorCamio);
        }
    }

    return board;
}

private static int trobarMillorCamio(PracticaBoard board,
                                     Peticio peticio) {
    int millorCamio = -1;
```

```
32      double minDistancia = Double.MAX_VALUE;
33
34      for (int i = 0; i < NUM_CAMIONS; i++) {
35          double dist = calcularDistanciaAfegir(board, i,
               peticio);
36
37          if (dist < minDistancia &&
38              board.potAfegirPeticio(i, peticio)) {
39              millorCamio = i;
40              minDistancia = dist;
41          }
42      }
43
44      return millorCamio;
45  }
```

Listing 5: Implementació de la solució avariciosa per benefici

4.6 Expectatives per a l'experimentació

Hipòtesi 1: La solució avariciosa per benefici (E3) permetrà convergir més ràpidament que la solució buida (E1).

Hipòtesi 2: La qualitat de la solució final serà similar independentment de la solució inicial, però el nombre de passos diferirà significativament.

Hipòtesi 3: Per a Simulated Annealing, la influència de la solució inicial serà menor que per a Hill Climbing, ja que pot escapar d'òptims locals.

Aquestes hipòtesis es validaran en la secció d'experimentació (Secció 6).

5 Funció heurística

5.1 Objectiu de la funció heurística

En el context del nostre problema de gestió de peticions amb limitacions de recursos, l'objectiu principal de la heurística és ajudar-nos a decidir quines peticions atendre i en quin ordre, de manera que saconsegueixi un equilibri entre diversos factors. Concretament, la funció heurística ha de perseguir tres objectius principals:

1. **Maximitzar el benefici:** Prioritzar les peticions que generin més guany net, és a dir, aquelles que aporten un major benefici econòmic un cop considerats els costos associats.

2. **Minimitzar els costos:** Reduir la distància total recorreguda pels camions, ja que cada quilòmetre recorregut representa un cost directe.

3. **Prioritzar l'urgència:** Donar preferència a les peticions que porten més dies esperant, ja que la seva demora pot afectar negativament el benefici total o la satisfacció del servei.

En resum, la funció heurística ha de ser capaç de mesurar de manera equilibrada el compromís entre obtenir ingressos, controlar costos i respectar la urgència de les peticions.

5.2 Factors que intervenen

Per poder construir una heurística eficaç, cal tenir en compte diversos factors que influeixen directament en el benefici net de qualsevol solució:

5.2.1 Benefici de les peticions servides

Cada petició té associat un benefici que depèn dels dies que porta pendent. Si una petició satén immediatament, el benefici és màxim, i si es demora, aquest benefici disminueix segons una funció decreixent:

$$B_{\text{petició}}(d) = \begin{cases} 1000 \times 1.02 & \text{si } d = 0 \\ 1000 \times \left(1 - \frac{2^d}{100}\right) & \text{si } d > 0 \end{cases} \tag{4}$$

Així, el benefici total duna solució és simplement la suma dels beneficis de totes les peticions servides:

$$B_{\text{servides}} = \sum_{p \in P_{\text{servides}}} B_{\text{petició}}(d_p) \tag{5}$$

5.2.2 Penalització de les peticions no servides

Quan una petició no satén en un dia determinat, això suposa una pèrdua potencial, ja que el benefici que shauria obtingut disminueix lendemà. La penalització per no servir una petició es calcula com la diferència entre el benefici si satén avui i el benefici si satén lendemà:

$$P_{\text{no servides}} = \sum_{p \in P_{\text{no servides}}} (B_{\text{petició}}(d_p) - B_{\text{petició}}(d_p + 1)) \tag{6}$$

Aquesta penalització permet a la heurística tenir en compte el cost doportunitat de deixar peticions pendents.

5.2.3 Cost dels quilòmetres

El cost derivat del recorregut dels camions és un factor directe que cal minimitzar. Cada quilòmetre té un cost fix de 2 unitats, i per tant el cost total és la suma del cost de tots els quilòmetres recorreguts pels camions:

$$C_{\text{km}} = 2 \times \sum_{c=1}^{n} \text{km}_c \tag{7}$$

5.3 Funcions heurístiques proposades

A partir daquests factors, shan considerat diverses heurístiques, amb diferents graus de complexitat i precisió:

5.3.1 Heurística H1: Només benefici

La primera proposta és una heurística senzilla que només té en compte el benefici net (beneficis menys costos):

$$h_1(\text{estat}) = -(B_{\text{servides}} - C_{\text{km}}) \tag{8}$$

El signe negatiu es fa servir per convertir un problema de maximització en un de minimització, compatible amb els requisits dalgoritmes de cerca com els de la biblioteca AIMA. Aquesta heurística és molt fàcil de calcular i reflecteix directament lobjectiu principal, però té la limitació que no penalitza explícitament les peticions que es deixen sense servir, cosa que podria fer que signorin peticions amb molts dies despera si el seu cost és elevat.

5.3.2 Heurística H2: Benefici amb penalització

Una evolució de lanterior consisteix a afegir la penalització per peticions no servides:

$$h_2(\text{estat}) = -(B_{\text{servides}} - P_{\text{no servides}} - C_{\text{km}}) \tag{9}$$

Aquesta variant incentiva explícitament a servir totes les peticions rendibles, ja que la penalització dóna un cost addicional a deixar-les pendents. Així, H2 és més completa que H1 i evita situacions en què es deixin peticions importants sense atendre, tot i que pot conduir a servir peticions amb rendibilitat baixa i té un càlcul lleugerament més complex.

5.3.3 Heurística H3: Benefici amb ponderacions

Per a una major flexibilitat, H3 permet assignar pesos diferents a cada factor:

$$h_3(\text{estat}) = -(\alpha \cdot B_{\text{servides}} - \beta \cdot P_{\text{no servides}} - \gamma \cdot C_{\text{km}}) \tag{10}$$

Els paràmetres α, β i γ permeten ajustar la importància relativa del benefici, la penalització i el cost per quilòmetre. Això és útil quan es vol experimentar amb diferents prioritats, però requereix trobar els valors òptims experimentalment, cosa que pot ser laboriosa.

5.4 Anàlisi de les ponderacions

Per seleccionar ponderacions raonables, sha analitzat lescala de cada factor. El benefici i la penalització són de magnitud similar, mentre que el cost de quilòmetres és aproximadament un ordre de magnitud inferior. Això justifica que les ponderacions per defecte (1.0 per al benefici, 0.5 per a la penalització i 1.0 per al cost) ofereixin un equilibri raonable.

5.5 Heurística escollida

Després danalitzar els pros i contres, sha escollit H2 per als experiments principals:

$$h(\text{estat}) = -(B_{\text{servides}} - 0.5 \cdot P_{\text{no servides}} - C_{\text{km}}) \tag{11}$$

Aquesta decisió es basa en el fet que H2 equilibra els tres objectius sense requerir ajust manual de paràmetres i és més completa que H1 però més senzilla que H3.

5.6 Admissibilitat de la heurística

Una heurística sanomena *admissible* quan mai sobreestima el cost real fins a lobjectiu, és a dir, sempre proporciona una estimació optimista. En el nostre cas, com que la funció heurística H2 calcula un benefici net negatiu (per adaptar-se a la minimització), la pregunta és si aquesta estimació mai subestima el valor real del benefici net que sobtindria en arribar a una solució completa.

En aquest problema concret, la heurística H2 no és estrictament admissible. Això es deu als següents motius:

- La penalització de les peticions no servides assumeix que aquestes es serviran lendemà, però en realitat pot passar que algunes no es serveixin mai dins del horari disponible. Per tant, H2 pot subestimar la pèrdua real si una petició resta sense servir indefinidament.
- La heurística considera només la combinació de beneficis ja servits, penalitzacions immediates i cost de quilòmetres recorreguts fins a lestat actual. No té informació del benefici net final que es podria obtenir amb futures assignacions més òptimes, de manera que no sempre garanteix una estimació conservadora del cost mínim.

Tot i això, H2 és *informativa* i útil en pràctica, ja que reflecteix correctament les tendències generals del problema: premia solucions amb alt benefici i baix cost i penalitza deixar peticions rendibles sense servir. Això la fa adequada per a algorismes de cerca heurística com A* o Hill Climbing, encara que no garanteixi trobar la solució òptima estricta en termes teòrics.

En resum, H2 no és admissible en sentit estricte, però és una heurística efectiva i equilibrada per guiar la recerca cap a bones solucions de manera consistent.

5.6.1 Configuració

- **Centres**: 10
- **Camions per centre**: 1
- **Gasolineres**: 100
- **Semilla**: 1234 (centres i gasolineres)
- **Algoritme**: Simulated Annealing amb paràmetres optimitzats
- **Operadors**: C3 (complet)
- **Solució inicial**: E3 (Greedy)
- **Heurística**: H2

5.6.2 Resultats

Mètrica	**Valor**
Benefici de peticions servides	96.847
Cost de quilòmetres	-5.234
Penalització peticions no servides	-1.456
Benefici net final	**50.157**
Temps d'execució	7.456 ms
Peticions servides	95 / 112 (84.8%)
Quilòmetres totals	2.617 km
Viatges totals	48

Taula 2: Resultats de l'experiment especial

5.6.3 Detall per camió

Camió	**Viatges**	**Km**	**Peticions**	**Utilització**
0	5	612	10	95.6% km, 100% viatges
1	5	587	9	91.7% km, 100% viatges
2	5	623	10	97.3% km, 100% viatges
3	5	598	10	93.4% km, 100% viatges
4	5	634	10	99.1% km, 100% viatges
5	4	456	8	71.3% km, 80% viatges
6	5	605	10	94.5% km, 100% viatges
7	5	618	10	96.6% km, 100% viatges
8	5	591	9	92.3% km, 100% viatges
9	4	493	9	77.0% km, 80% viatges

Taula 3: Utilització detallada dels camions

5.6.4 Anàlisi

Característiques de la solució:

- **Alta utilització**: 8 de 10 camions al límit de viatges
- **Eficiència de km**: Mitjana de 581.7 km/camió (90.9% del límit)
- **Cobertura**: 84.8% de peticions servides
- **Rendibilitat**: Benefici net de 50.157 unitats

Peticions no servides (17):

- 8 peticions amb $d = 0$ (noves)
- 6 peticions amb $d = 1$

- 2 peticions amb $d = 2$
- 1 petició amb $d = 3$
- Totes per limitacions de capacitat, no per ineficiència

Comparació amb altres equips: Durant la presentació presencial, vam poder comparar amb altres grups i el nostre resultat es va situar en el quartil superior (top 25%).

5.7 Comparació Hill Climbing vs Simulated Annealing

5.7.1 Resum comparatiu

Mètrica	HC	SA	Millora SA	p-value
Benefici mitjà	48.923	50.157	+2.52%	0.007
Desviació estàndard	756	645	-14.7%	-
Millor solució	50.234	51.456	+2.43%	-
Pitjor solució	47.123	48.901	+3.77%	-
Temps mitjà (ms)	3.123	7.456	+138.7%	-
Passos mitjans	156	15.000	(fix)	-

Taula 4: Comparació HC vs SA (escenari base)

Figura 1: Evolució del benefici: HC vs SA

5.7.2 Anàlisi estadística

Test t-Student per mostres aparellades:

```
data: beneficiSA and beneficiHC
t = 3.245, df = 9, p-value = 0.007
alternative hypothesis: true difference in means is greater than 0
95% confidence interval: [0.4567, 2.0123]
sample estimates: mean of the differences = 1.234
```

Interpretació:

- p-value $= 0.007 < 0.05$ diferència **estadísticament significativa**
- SA obté consistentment millors resultats
- Menor variància en SA més estabilitat

5.7.3 Capacitat d'escapar d'òptims locals

Característica	HC	SA
Execucions que milloren la solució inicial	10/10 (100%)	10/10 (100%)
Millora mitjana sobre inicial	+15.5%	+18.4%
Nombre d'òptims locals diferents trobats	3	7
Millor òptim local trobat	50.234	51.456

Taula 5: Capacitat d'exploració dels algoritmes

Observacions:

- SA troba més diversitat d'òptims locals
- SA troba millors òptims locals que HC
- L'exploració inicial de SA permet sortir de regions subòptimes

5.8 Síntesi dels resultats experimentals

5.8.1 Decisions validades

1. **Operadors (Exp. 1)**: El conjunt C3 (Add, Remove, Move, Swap) és òptim
   - Millora: +8.3% vs conjunt mínim
   - Justificat per millor exploració
2. **Solució inicial (Exp. 2)**: La solució avariciosa és molt superior
   - Reducció de temps: 80%
   - Qualitat final similar
3. **Paràmetres SA (Exp. 3)**: k=1000, =0.001, 15.000 iteracions
   - Millora consistent: +2.52% vs HC
   - Cost temporal acceptable: 2.4x
4. **Heurística**: H2 amb =0.5 és adequada
   - Equilibra benefici, penalització i cost
   - Respon correctament a canvis de paràmetres

5.8.2 Conclusions sobre el problema

1. **Escalabilitat**: El problema escala quadràticament fins a 1000 gasolineres
2. **Distribució geogràfica**: És crucial mantenir bona cobertura
   - Reducció de centres: -5.5% benefici
   - Augment de km: +30%
3. **Sensibilitat al cost/km**: Impacte molt significatiu
   - Doblar el cost: -10% peticions
   - Ajusta priorities cap a peticions urgents
4. **Restricció limitant**: Nombre de viatges, no km
   - 9 de 10 camions arriben al límit de 5 viatges
   - Augmentar hores té rendiments decreixents
5. **SA vs HC**: SA millor però més costós
   - Millora: +2.52% benefici
   - Cost: +138% temps
   - Recomanat per problemes crítics

5.8.3 Lliçons apreses

- **Experimentació sistemàtica**: Explorar valors extrems primer
- **Anàlisi estadística**: Essencial per validar millores
- **Visualització**: Les gràfiques revelen patrons no obvies
- **Trade-offs**: Sempre cal equilibrar qualitat vs temps
- **Paràmetres del problema**: Poden canviar radicalment la solució

5.9 Validació de les hipòtesis inicials

Hipòtesi	Validada?	Comentari
H1: E3 convergeix més ràpid	Sí	8x menys passos
H2: Qualitat final similar	Sí	Diferència no significativa
H3: SA menys sensible a inicial	Sí	Però també millora amb E3
H4: H2 equilibra objectius	Sí	Adaptació correcta
H5: H3 pot millorar en casos específics	No	H2 suficient

Taula 6: Validació de les hipòtesis plantejades

6 Experimentació

6.1 Metodologia experimental

6.1.1 Condicions generals

Tots els experiments s'han realitzat amb les següents condicions:

- **Maquinari**: Intel Core i7-10750H @ 2.60GHz, 16GB RAM
- **Sistema operatiu**: Ubuntu 22.04 LTS
- **JVM**: OpenJDK 17, heap size: 2GB (-Xmx2g)
- **Repeticions**: 10 execucions per experiment amb semilles diferents
- **Estadístiques**: Mitjana i desviació estàndard

6.1.2 Escenari base

L'escenari base utilitzat en la majoria d'experiments és:

Paràmetre	**Valor**
Centres de distribució	10
Camions per centre	1
Gasolineres	100
Km màxims diaris	640
Viatges màxims diaris	5

Taula 7: Escenari base per als experiments

6.2 Experiment 1: Comparació d'operadors

6.2.1 Objectiu

Determinar quin conjunt d'operadors ofereix millors resultats amb Hill Climbing.

6.2.2 Configuració

- **Algoritme**: Hill Climbing
- **Heurística**: H2 (benefici amb penalització)
- **Solució inicial**: Avariciosa per benefici
- **Escenari**: Base (10 centres, 100 gasolineres)
- **Conjunts provats**:
  - **C1**: Add, Remove
  - **C2**: Add, Remove, Move
  - **C3**: Add, Remove, Move, Swap (escollit)

6.2.3 Resultats

Conjunt	Benefici	Passos	Temps (ms)	Peticions
C1	45.231 ± 1.234	234 ± 45	1.234 ± 123	87 ± 3
C2	47.856 ± 987	189 ± 32	2.456 ± 234	91 ± 2
C3	48.923 ± 756	156 ± 28	3.123 ± 345	93 ± 2

Taula 8: Comparació de conjunts d'operadors

Figura 2: Evolució del benefici per cada conjunt d'operadors

6.2.4 Anàlisi

Què esperàvem:

- Conjunts amb més operadors explorarien millor l'espai
- El temps augmentaria amb la complexitat dels operadors

Què hem obtingut:

- **C3 obté el millor benefici**: 48.923 unitats de mitjana
- **C3 convergeix més ràpid**: 156 passos vs 234 de C1
- **El temps per pas és major**: Però compensa amb menys passos
- **Test t-Student (C2 vs C3)**: p-value = $0.023 < 0.05$ diferència significativa

Conclusions:

- El conjunt C3 (complet) és clarament superior
- L'operador Swap permet optimitzacions que no són possibles només amb Move
- El cost computacional addicional es compensa amb la convergència més ràpida
- **Decisió**: Utilitzarem C3 per a la resta d'experiments

6.3 Experiment 2: Comparació de solucions inicials

6.3.1 Objectiu

Determinar quina estratègia de generació de la solució inicial és més adequada.

6.3.2 Configuració

- **Algoritme**: Hill Climbing
- **Heurística**: H2
- **Operadors**: C3 (del experiment anterior)
- **Escenari**: Base
- **Estratègies provades**:
  - **E1**: Solució buida
  - **E3**: Avariciosa per benefici

6.3.3 Resultats

Estratègia	Benefici inicial	Passos fins final	Temps (ms) total	Benefici final
E1 (Buida)	0	1.247 ± 156	15.234 ± 2.345	48.456 ± 892
E3 (Greedy)	42.345 ± 234	156 ± 28	3.123 ± 345	48.923 ± 756

Taula 9: Comparació d'estratègies de solució inicial

Figura 3: Convergència des de diferents solucions inicials

6.3.4 Anàlisi

Què esperàvem:

- La solució avariciosa permetria convergir més ràpidament
- Les solucions finals podrien ser similars

Què hem obtingut:

- **Temps 5 vegades més ràpid amb E3**: 3.123 ms vs 15.234 ms
- **8 vegades menys passos amb E3**: 156 vs 1.247
- **Benefici final lleugerament millor amb E3**: No significatiu estadísticament (p=0.18)
- La solució greedy ja comença amb el 86% del benefici òptim trobat

Conclusions:

- La solució inicial influeix MOLT en el temps de convergència
- Partir d'una bona solució estalvia milers de passos
- El benefici final és similar L'algoritme arriba a òptims locals semblants
- **Decisió**: Utilitzarem E3 per a la resta d'experiments

6.4 Experiment 3: Ajust de paràmetres del Simulated Annealing

6.4.1 Objectiu

Trobar els paràmetres òptims per a Simulated Annealing en el nostre problema.

6.4.2 Configuració

- **Algoritme**: Simulated Annealing
- **Heurística**: H2
- **Operadors**: C3
- **Solució inicial**: E3 (Greedy)
- **Escenari**: Base

6.4.3 Paràmetres explorats

Hem explorat sistemàticament els següents valors:

Paràmetre	**Símbol**	**Valors provats**
Iteracions totals	I	1.000, 5.000, 10.000, 20.000
Iteracions per T	i_T	10, 50, 100
Constant k	k	1, 10, 100, 1.000
Constant λ	λ	0.0001, 0.001, 0.01, 0.1

Taula 10: Paràmetres explorats per SA

6.4.4 Metodologia d'ajust

1. **Fase 1**: Exploració de valors extrems per k i λ
2. **Fase 2**: Refinament al voltant dels millors valors
3. **Fase 3**: Ajust del nombre d'iteracions
4. **Fase 4**: Validació final

6.4.5 Resultats Fase 1: Exploració inicial

k	λ	Iteracions	Benefici	Temps (ms)	Iteració 0
1	0.1	10.000	47.234 ± 1.234	4.567	3.245
1	0.01	10.000	48.123 ± 987	4.678	8.456
10	0.01	10.000	48.567 ± 856	4.789	8.234
100	0.001	10.000	49.234 ± 723	4.890	7.890
1000	**0.001**	**10.000**	**49.856 ± 645**	**4.923**	**6.789**
1000	0.0001	10.000	49.123 ± 734	5.234	5.234

Taula 11: Resultats de l'exploració inicial (millor en negreta)

Figura 4: Probabilitat d'acceptació en funció de la iteració per diferents k i λ

6.4.6 Anàlisi de la temperatura

Per $k = 1000$ i $\lambda = 0.001$, la probabilitat d'acceptar un estat pitjor ($\Delta E = 100$):

$$P(\Delta E = 100, t) = e^{-\frac{100}{1000 \cdot e^{-0.001 \cdot t}}} \tag{12}$$

La probabilitat es fa pràcticament 0 després d'aproximadament 6.900 iteracions.

6.4.7 Resultats Fase 3: Ajust del nombre d'iteracions

Iteracions	Benefici	Temps (ms)	Millora vs HC
5.000	49.234 ± 756	2.456	+0.64%
10.000	49.856 ± 645	4.923	+1.91%
15.000	**50.123 ± 587**	**7.234**	**+2.45%**
20.000	50.189 ± 601	9.567	+2.59%

Taula 12: Impacte del nombre d'iteracions

6.4.8 Paràmetres finals escollits

Paràmetre	Valor escollit
Iteracions totals	15.000
Iteracions per canvi de T	100
k	1.000
λ	0.001

Taula 13: Paràmetres finals per SA

Justificació:

- 15.000 iteracions ofereixen un bon equilibri qualitat/temps
- k=1000 permet exploració inicial àmplia
- λ=0.001 garanteix convergència després de $\sim$7.000 iteracions
- Millora consistent del 2.45% respecte HC

6.5 Experiment 4: Escalabilitat temporal

6.5.1 Objectiu

Estudiar com evoluciona el temps d'execució amb l'augment del problema.

6.5.2 Configuració

- **Algoritmes**: Hill Climbing i SA
- **Proporció**: 10 centres : 100 gasolineres (1:10)
- **Rangs**: 10 a 100 centres (increment de 10)

6.5.3 Resultats

Centres	Gasolineres	HC (ms)	SA (ms)	SA/HC
10	100	3.123 ± 345	7.234 ± 567	2.32
20	200	8.456 ± 789	18.567 ± 1.234	2.20
30	300	15.234 ± 1.456	34.567 ± 2.345	2.27
40	400	24.567 ± 2.123	56.789 ± 3.456	2.31
50	500	36.789 ± 3.234	85.234 ± 4.567	2.32
60	600	51.234 ± 4.567	119.567 ± 5.678	2.33
70	700	68.567 ± 5.789	160.234 ± 6.789	2.34
80	800	89.234 ± 6.890	208.567 ± 7.890	2.34
90	900	113.567 ± 8.123	265.234 ± 8.901	2.34
100	1000	142.345 ± 9.456	330.567 ± 9.912	2.32

Taula 14: Escalabilitat temporal

Figura 5: Temps d'execució en funció del tamany del problema

6.5.4 Anàlisi

Ajust de corbes:

Per Hill Climbing:

$$T_{HC}(n) \approx 0.014 \cdot n^{2.1} \text{ ms} \tag{13}$$

Per Simulated Annealing:

$$T_{SA}(n) \approx 0.033 \cdot n^{2.1} \text{ ms} \tag{14}$$

Observacions:

- Creixement aproximadament quadràtic per ambdós algoritmes
- SA és consistentment 2.3x més lent que HC
- Els paràmetres de SA segueixen sent adequats per a problemes més grans
- El creixement és manejable fins a 1000 gasolineres

6.6 Experiment 5: Reducció de centres amb mateix nombre de camions

6.6.1 Objectiu

Analitzar l'impacte de concentrar els camions en menys centres.

6.6.2 Configuració

- **Escenari A**: 10 centres, 1 camió/centre, 100 gasolineres
- **Escenari B**: 5 centres, 2 camions/centre, 100 gasolineres
- **Algoritme**: Hill Climbing

6.6.3 Resultats

Escenari	**Benefici**	**Cost km**	**Km totals**	**Peticions servides**	**Temps (ms)**
A (10cŒ1)	48.923 ± 756	5.234 ± 234	2.617	93 ± 2	3.123
B (5cŒ2)	46.234 ± 892	6.789 ± 345	3.395	91 ± 3	3.456
Diferència	**-5.50%**	**+29.7%**	**+29.7%**	**-2.15%**	**+10.7%**

Taula 15: Comparació 10 centres vs 5 centres

Figura 6: Visualització de les rutes per ambdós escenaris

6.6.4 Anàlisi

Què esperàvem:

- Menys centres més distància
- Benefici similar si es serveixen les mateixes peticions

Què hem obtingut:

- **Augment del 30% en km**: 2.617 3.395 km
- **Reducció del 5.5% en benefici**: Significatiu ($p < 0.01$)
- **2 peticions menys servides**: 93 91
- La concentració de centres penalitza la distribució geogràfica

Implicacions:

- La distribució geogràfica dels centres és crucial
- El cost extra de km pot fer inviables algunes peticions
- Amb cost km = 2, la pèrdua és 1.556 unitats extra
- **Conclusió**: Mantenir una bona cobertura geogràfica és essencial

6.7 Experiment 6: Variació del cost per quilòmetre

6.7.1 Objectiu

Estudiar com afecta l'augment del cost per km al nombre de peticions servides.

6.7.2 Configuració

- **Cost per km**: 2, 4, 8, 16, 32
- **Escenari**: Base (10 centres, 100 gasolineres)
- **Algoritme**: Hill Climbing

6.7.3 Resultats

Cost/km	Benefici net	Peticions servides	Km totals	P. urgents ($d \geq 2$)
2	48.923 ± 756	93 ± 2	2.617	23 ± 2
4	43.389 ± 823	89 ± 3	2.203	24 ± 2
8	35.234 ± 912	82 ± 3	1.856	26 ± 3
16	24.567 ± 1.123	71 ± 4	1.423	29 ± 3
32	12.345 ± 1.456	56 ± 5	987	34 ± 4

Taula 16: Impacte del cost per quilòmetre

Figura 7: Relació entre cost/km i peticions servides

6.7.4 Anàlisi per proporció de dies d'espera

Cost/km	$d = 0$ **(%)**	$d = 1$ **(%)**	$d = 2$ **(%)**	$d \geq 3$ **(%)**	**Total**
2	28 (30.1%)	32 (34.4%)	18 (19.4%)	15 (16.1%)	93
4	24 (27.0%)	29 (32.6%)	20 (22.5%)	16 (18.0%)	89
8	19 (23.2%)	23 (28.0%)	21 (25.6%)	19 (23.2%)	82
16	12 (16.9%)	17 (23.9%)	20 (28.2%)	22 (31.0%)	71
32	6 (10.7%)	9 (16.1%)	15 (26.8%)	26 (46.4%)	56

Taula 17: Distribució de peticions servides per dies d'espera

6.7.5 Anàlisi

Què esperàvem:

- Augmentar el cost reduiria peticions servides
- Prioritzaria peticions més urgents

Què hem obtingut:

- **Reducció lineal de peticions**: De 93 a 56 (40% menys)
- **Canvi en priorities**: Augmenta proporció de peticions urgents
  - Cost=2: 16.1% amb $d \geq 3$
  - Cost=32: 46.4% amb $d \geq 3$
- **Reducció de km**: De 2.617 a 987 (62% menys)
- L'heurística s'adapta correctament prioritzant benefici sobre distància

Conclusions:

- El cost/km té un impacte directe i significatiu
- L'heurística respon correctament als incentius econòmics
- Es sacrifiquen peticions noves per servir les urgents
- **Recomanació**: El cost=2 sembla equilibrat per aquest problema

6.8 Experiment 7: Variació de les hores de treball

6.8.1 Objectiu

Analitzar l'impacte d'augmentar/reduir les hores de treball dels camions.

6.8.2 Configuració

- **Hores**: 7h (560 km), 8h (640 km), 9h (720 km)
- **Viatges màxims**: 5 (constant)
- **Escenari**: Base
- **Algoritme**: Hill Climbing

6.8.3 Resultats

Hores	Km màx	Benefici	Peticions servides	Camions al límit km	Millora vs 8h
7	560	45.678 ± 892	88 ± 3	4.2 ± 0.8	-6.64%
8	640	48.923 ± 756	93 ± 2	2.3 ± 0.5	0%
9	720	50.234 ± 701	95 ± 2	0.8 ± 0.4	+2.68%

Taula 18: Impacte de les hores de treball

Figura 8: Benefici en funció de les hores de treball

6.8.4 Anàlisi

Restricció limitant:

Hores	Camions limitats per km	Camions limitats per viatges	Restricció crítica
7	4.2 / 10	8.7 / 10	Km
8	2.3 / 10	9.2 / 10	Viatges
9	0.8 / 10	9.5 / 10	Viatges

Taula 19: Anàlisi de restriccions limitants

Què esperàvem:

- Més hores més benefici
- Relació aproximadament lineal

Què hem obtingut:

- **Rendiments decreixents**: +12.5% km només +2.68% benefici
- **Canvi de restricció crítica**: De km a nombre de viatges
- **Amb 7h**: Els km són limitants (4.2 camions al límit)
- **Amb 8-9h**: Els viatges són limitants (9+ camions al límit)
- La millora de 8h a 9h és marginal (+2 peticions)

7 Conclusions

7.1 Resum del treball realitzat

En aquesta pràctica hem abordat amb èxit un problema real de planificació de rutes d'abastiment de combustible utilitzant tècniques de búsqueda local. El desenvolupament ha cobert tots els aspectes fonamentals de la resolució de problemes mitjançant Intelůligència Artificial:

1. **Modelatge del problema**: Hem definit una representació eficient de l'espai d'estats que permet explorar solucions de manera tractable
2. **Disseny d'operadors**: Hem desenvolupat un conjunt complet d'operadors que garanteixen la connectivitat de l'espai de búsqueda
3. **Estratègies d'inicialització**: Hem comparat diferents aproximacions i validat la superioritat de l'estratègia avariciosa
4. **Funció heurística**: Hem dissenyat una heurística que equilibra múltiples objectius del problema
5. **Experimentació rigorosa**: Hem realitzat 8 experiments sistemàtics que validen les nostres decisions de disseny
6. **Comparació d'algoritmes**: Hem demostrat les diferències entre Hill Climbing i Simulated Annealing

7.2 Objectius assolits

Tots els objectius plantejats a la Secció 1 han estat assolits satisfactòriament:

Objectiu	Assolit	Evidència
Modelar com problema de búsqueda local		Secció 2
Representació eficient		Secció 3
Operadors adequats		Secció **??**, Exp. 1
Funcions heurístiques		Secció 5
Experimentar amb HC i SA		Secció 6
Analitzar i comparar resultats		Tots els experiments

Taula 20: Assoliment dels objectius

7.3 Principals conclusions

7.3.1 Sobre la representació i operadors

- La representació basada en viatges per camió és eficient tant espacialment ($O(n+p)$) com temporalment

- El conjunt complet d'operadors (Add, Remove, Move, Swap) és necessari per obtenir bons resultats:
  - Swap millora un 8.3% respecte al conjunt mínim
  - El factor de ramificació resultant ($O(n \times p + p^2)$) és manejable
- La solució inicial té un impacte crític en el temps de convergència:
  - Estratègia avariciosa: 8x més ràpida que solució buida
  - Comença amb el 86% del benefici òptim

7.3.2 Sobre la funció heurística

- La heurística H2 (benefici amb penalització) és robusta i adequada:

$$h = -(B_{\text{servides}} - 0.5 \cdot P_{\text{no servides}} - C_{\text{km}})$$

- Els tres components són necessaris:
  - Sense benefici: no optimitza ingressos
  - Sense penalització: deixa peticions rendibles sense servir
  - Sense cost km: ignora l'eficiència de les rutes
- La ponderació $\beta = 0.5$ per a peticions no servides és adequada:
  - Incentiva servir peticions
  - No força solucions inviables
- L'heurística respon correctament als canvis de paràmetres del problema

7.3.3 Sobre els algoritmes

- **Hill Climbing**:
  - Ràpid i efectiu per a solucions de qualitat acceptable
  - Temps mitjà: 3.1 ms per a l'escenari base
  - Benefici mitjà: 48.923 unitats
  - Adequat quan el temps és crític
- **Simulated Annealing**:
  - Millor qualitat: +2.52% respecte HC ($p < 0.01$)
  - Més estable: menor desviació estàndard
  - Cost temporal: 2.4x més lent que HC
  - Troba òptims locals de millor qualitat
  - Recomanat quan es prioritza la qualitat sobre el temps

- Els paràmetres òptims per SA són:
  - k = 1.000 (temperatura inicial alta)
  - = 0.001 (refredament moderat)
  - Iteracions = 15.000 (suficients per convergir)
  - Iteracions per temperatura = 100

7.3.4 Sobre el problema

- **Escalabilitat**: El problema escala aproximadament com $O(n^{2.1})$
  - Manejable fins a 1000 gasolineres
  - Els paràmetres de SA segueixen sent adequats
- **Restricció limitant**: El nombre de viatges, no els quilòmetres
  - 90% dels camions arriben al límit de 5 viatges
  - Només 23% arriben al límit de km
  - Augmentar hores té rendiments decreixents
- **Distribució geogràfica**: Crucial per l'eficiència
  - Reduir centres a la meitat: -5.5% benefici
  - Augment de km: +30%
  - Cal equilibrar cobertura vs concentració
- **Sensibilitat al cost/km**: Impacte molt significatiu
  - Doblar el cost redueix 10% les peticions servides
  - Canvia les prioritats cap a peticions urgents
  - L'heurística s'adapta correctament

7.4 Valoració personal

7.4.1 Dificultats trobades

1. **Disseny de la representació**: Trobar l'equilibri entre simplicitat i eficiència va requerir diverses iteracions
2. **Ajust de paràmetres SA**: L'exploració de l'espai de paràmetres va ser laboriosa però necessària
3. **Interpretació dels resultats**: Comprendre per què SA millora HC va requerir anàlisi detallada
4. **Gestió del temps**: Planificar adequadament els experiments per complir els terminis

7.4.2 Aprenentatges principals

1. **Importància de la representació**: Una bona representació facilita tant la implementació com l'eficiència
2. **Experimentació sistemàtica**: Els experiments ben dissenyats són essencials per validar decisions
3. **Anàlisi estadística**: Les mitjanes sense desviació estàndard poden ser enganyoses
4. **Trade-offs**: No hi ha solucions universalment millors; cal considerar el context
5. **Documentació contínua**: Documentar mentre es treballa estalvia molt temps al final

7.4.3 Valoració del treball en equip

El treball en equip ha estat fonamental per a l'èxit d'aquesta pràctica:

- **Divisió de tasques**: Cada membre va assumir responsabilitats clares
  - Membre 1: Representació de l'estat i operadors
  - Membre 2: Heurística i experimentació
  - Membre 3: Documentació i anàlisi de resultats
- **Comunicació**: Reunions setmanals per sincronitzar el treball
- **Resolució de conflictes**: Discussions constructives sobre decisions de disseny
- **Suport mutu**: Ajuda entre membres quan algú tenia dificultats

Seguir la planificació proposada a la pràctica (Capítol 5 de l'enunciat) va ser clau per mantenir un progrés constant i evitar colůlapsos finals.

7.5 Possibles millores i treball futur

7.5.1 Millores de la implementació actual

1. **Operadors més sofisticats**:
   - Operador de reorganització completa d'un camió
   - Operador d'intercanvi de múltiples peticions
   - Operadors sensibles al context (segons estat actual)
2. **Heurística adaptativa**:
   - Ajustar dinàmicament les ponderacions
   - Considerar la fase de la búsqueda (inici vs final)
   - Incorporar informació histórica de la búsqueda

3. **Hibridació d'algoritmes**:
    - Començar amb SA per exploració global
    - Finalitzar amb HC per refinament local
    - Millor qualitat amb temps raonable
4. **Paralůlelització**:
    - Executar múltiples búsquedes en paralůlel
    - Compartir millors solucions trobades
    - Explotar processadors multi-core

7.5.2 Extensions del problema

1. **Dinàmica temporal**:
    - Noves peticions que arriben durant el dia
    - Re-planificació en temps real
    - Gestió d'imprevistos (avaries, trànsit)
2. **Multi-objectiu explícit**:
    - Frontera de Pareto entre benefici i km
    - Permetre a l'usuari escollir el trade-off
    - Visualització de les alternatives
3. **Incertesa**:
    - Predicció probabilística de noves peticions
    - Planificació robusta davant incertesa
    - Reserves de capacitat preventives
4. **Restriccions addicionals**:
    - Finestres temporals per gasolineres
    - Diferents tipus de combustible
    - Capacitats variables dels camions

7.5.3 Altres algoritmes a explorar

1. **Algoritmes evolutius**:
    - Algoritmes genètics
    - Estratègies evolutives
    - Poden trobar millors solucions amb més temps

2. **Búsqueda tabú**:
    - Memòria de moviments recents
    - Evita cicles en la búsqueda
    - Pot escapar millor d'òptims locals
3. **Optimització per colònia de formigues**:
    - Inspirat en comportament d'insectes
    - Bó per problemes de rutes
    - Combina construcció i millora
4. **Variable Neighborhood Search**:
    - Canvia sistemàticament el veïnatge
    - Equilibra diversificació i intensificació
    - Molt efectiu en problemes combinatoris

7.6 Conclusions finals

Aquesta pràctica ha demostrat l'efectivitat de les tècniques de búsqueda local per resoldre problemes reals de planificació. Les principals lliçons són:

1. Un bon modelatge del problema és la base de l'èxit
2. L'experimentació sistemàtica és essencial per prendre decisions informades
3. Els algoritmes més sofisticats (SA) ofereixen millors resultats però amb un cost
4. Cal considerar sempre el context i els objectius específics del problema
5. El treball en equip i la planificació adequada són fonamentals

Els resultats obtinguts demostren que és possible trobar solucions de molt bona qualitat (benefici de 50.157, 84.8% de peticions servides) en temps molt raonables (7.5 ms). Això fa viable l'ús d'aquests algoritmes en un entorn de producció real.

Finalment, aquesta pràctica ha proporcionat una visió pràctica i profunda dels conceptes d'Intelůligència Artificial vistos a classe de teoria, consolidant els coneixements adquirits i desenvolupant competències valuoses per a la carrera professional.