Pràctica de Cerca Local

Planificació de Rutes de Proveïment de Benzineres

Grau IA – Q1 Curs 2025-2026

Departament de Ciències de la Computació
Universitat Politècnica de Catalunya

Anel Ademovic Suljic
anel.suljic@estudiantat.upc.edu

Aleix Pitarch
aleix.pitarch@estudiantat.upc.edu

Joan Solina
joan.solina@estudiantat.upc.edu

26 d'octubre de 2025

Índex

1 Introducció

1.1 Context del problema

Aquesta pràctica se centra en la resolució d'un problema de planificació de rutes de proveïment per a una companyia de distribució de combustible mitjançant algorismes de cerca local. La pràctica implica el disseny i la implementació dels elements necessaris per a algorismes Hill Climbing i Simulated Annealing així com la realització d'experiments per avaluar el seu comportament en diferents escenaris. A més, es durà a terme una anàlisi comparativa dels resultats obtinguts amb ambdós algorismes, amb l'objectiu d'extreure'n 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:

```
public class PracticaBoard {
    // Informacio del problema (compartida per tots els estats)
    private static Gasolineras gasolineres;
    private static CentrosDistribucion centres;
    private static Map<Integer, Peticio> peticions;

    // Constants del problema
    private static final int KM_MAX = 640;
    private static final int VIATGES_MAX = 5;
    private static final double COST_KM = 2.0;
    private static final double PREU_BASE = 1000.0;

    // ...
}
```

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:

```
public class PracticaBoard {
    // ...

    // Assignacions especifiques d'aquest estat
    private List<List<Viatge>> assignacions;  // viatges per
        camio
    private Set<Integer> peticionsServides;

    // Metriques pre-calculades (per eficiencia)
    private double beneficiTotal;
    private int[] kmPerCamio;
    private int[] viatgesPerCamio;
}
```

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 Operadors

4.1 Descripció dels Operadors

Els operadors són les accions que ens permeten moure'ns per l'espai de solucions. Cada operador agafa un estat i genera un nou estat modificant lleugerament l'assignació de peticions. La clau és dissenyar operadors que permetin arribar a qualsevol solució però que tinguin un factor de ramificació raonable. N'hem considerat 5:

4.1.1 Operador: Afegir Petició

Aquest operador afegeix una petició no assignada a un viatge existent que encara no ha arribat a la seva capacitat màxima. Per poder aplicar-lo, cal que existeixi almenys una petició no assignada i almenys un viatge que no estigui ple. A més, el viatge destí ha de complir totes les restriccions després d'afegir la petició, és a dir, no pot superar la capacitat del camió, la distància total del viatge ha de romandre dins dels límits permesos, i no es pot excedir el nombre màxim de gasolineres per viatge.
L'efecte principal és que la petició passa de la llista de no assignades a formar part del viatge seleccionat, i es recalculen automàticament tots els paràmetres del viatge com la distància i la càrrega. La implementació recorre tots els camions i tots els seus viatges, provant d'afegir cada petició no assignada a cada viatge que tingui espai disponible.

4.1.2 Operador: Treure Petició

Aquest operador fa exactament l'operació inversa a l'anterior: elimina una petició d'un viatge existent i la torna a la llista de peticions no assignades. L'única condició necessària és que existeixi almenys un viatge amb peticions assignades. No hi ha cap restricció adicional, ja que treure una petició sempre allibera espai i redueix la càrrega del viatge.
Quan s'elimina una petició, aquesta es retorna a la llista de no assignades i es recalculen els paràmetres del viatge origen. Si després de treure la petició el viatge queda completament buit, aquest s'elimina automàticament. L'operador recorre tots els viatges de tots els camions i genera un successor per cada petició que pot ser eliminada.

4.1.3 Operador: Moure Petició

Aquest operador transfereix una petició d'un viatge origen a un viatge destí diferent, que pot pertànyer al mateix camió o a un altre. Per aplicar-lo, el viatge origen ha de tenir almenys una petició assignada, el viatge destí no pot estar ple, i obviament origen i destí no poden ser el mateix viatge. A més, tant el viatge origen com el destí han de complir totes les restriccions després del moviment.
Els efectes són una combinació dels dos operadors anteriors: la petició s'elimina del viatge origen i s'afegeix al viatge destí. Si el viatge origen queda buit després del moviment, s'elimina automàticament. L'operador considera totes les possibles combinacions de viatges

origen i destí entre tots els camions, i per a cada combinació vàlida prova de moure cada petició disponible.

4.1.4 Operador: Intercanviar Peticions

Aquest operador intercanvia dues peticions entre dos viatges diferents, que poden pertànyer al mateix camió o a camions diferents. Cal que existeixin almenys dos viatges amb peticions assignades, que els dos viatges no siguin el mateix, i que ambdós compleixin les restriccions després de l'intercanvi.
L'efecte és un simple intercanvi: cada petició passa a ocupar la posició de l'altra en el seu respectiu viatge. L'operador considera totes les parelles possibles de viatges i, per a cada parella, prova d'intercanviar cada parell de peticions possible. Només genera successors quan l'intercanvi manté les restriccions en ambdós viatges.

4.1.5 Operador: Crear Viatge

Aquest operador crea un nou viatge per a un camió i li assigna una petició no assignada. Les condicions són que existeixi almenys una petició no assignada, que el camió seleccionat no hagi arribat al nombre màxim de viatges permesos per dia, i que el nou viatge compleixi les restriccions.
L'efecte és la creació d'un nou viatge amb una sola petició, que s'afegeix a la llista de viatges del camió. La petició passa de no assignada a assignada, i s'incrementa el comptador de viatges del camió. L'operador recorre tots els camions que encara poden acceptar més viatges i, per a cada petició no assignada, intenta crear un nou viatge si es compleixen les restriccions.

4.2 Anàlisi del Factor de Ramificació

El factor de ramificació determina quants successors es poden generar des d'un estat donat. Per analitzar-lo, utilitzarem la notació següent: C per al nombre de camions, V per al nombre total de viatges en l'estat actual, P per al nombre total de peticions assignades, P_{max} per al nombre màxim de peticions per viatge, i N per al nombre de peticions no assignades.

4.2.1 Anàlisi per Operador

L'**operador afegir** itera sobre tots els viatges no plens i prova d'afegir-hi cada petició no assignada. En el pitjor cas, si tots els viatges tenen espai, genera aproximadament $V \times N$ successors. Aquesta xifra es redueix considerablement quan els viatges s'omplen, arribant a zero quan tots els viatges estan plens o no queden peticions per assignar.

L'**operador treure** genera un successor per cada petició assignada en qualsevol viatge. Això dóna un factor de ramificació de P successors, que és relativament estable durant l'execució, tot i que disminueix lleugerament a mesura que s'assignen més peticions.

L'**operador moure** és més costós. Ha de considerar cada parella possible de viatges (origen i destí) i, per a cada una, provar de moure cada petició del viatge origen al destí. Això resulta en aproximadament $V^2 \times \frac{P}{V}$ successors en el cas mitjà, que es pot simplificar a $V \times P$. En el pitjor cas, quan hi ha molts viatges amb poques peticions cadascun, aquest nombre pot créixer fins a $V^2 \times P_{max}$.

L'**operador intercanviar** té el factor de ramificació més elevat. Per a cada parella de viatges diferents, prova d'intercanviar cada parell de peticions entre ells. Això genera $\binom{V}{2} \times \frac{P}{V} \times \frac{P}{V}$ successors, que en el cas general és de l'ordre de $V^2 \times P^2/V^2 = P^2$. Quan hi ha molts viatges amb poques peticions, aquest valor pot arribar a ser molt elevat.

L'**operador crear** genera $C \times N$ successors en el pitjor cas, quan tots els camions poden acceptar més viatges. Aquest nombre es redueix linealment amb el nombre de peticions no assignades i amb el nombre de camions que ja han arribat al màxim de viatges permesos.

4.3 Elecció dels Operadors

Els cinc operadors han estat seleccionats per proporcionar una cobertura completa de l'espai de solucions i permetre tant la construcció incremental com l'optimització de solucions existents.

4.3.1 Construcció de Solucions

- **Afegir Petició:** Operador fonamental per construir solucions des d'un estat inicial buit o parcial. Permet la construcció incremental de viatges.
- **Crear Viatge:** Complementa l'operador anterior permetent iniciar nous viatges quan els existents no són adequats.

4.3.2 Millora de Solucions

- **Moure Petició:** Permet reequilibrar la càrrega entre viatges i camions, essencial per optimitzar distàncies i utilitzar millor la capacitat.
- **Intercanviar Peticions:** Facilita l'optimització local sense canviar el nombre de peticions assignades, útil per millorar rutes sense desestabilitzar la solució.

4.3.3 Exploració i Backtracking

- **Treure Petició:** Per a poder treure viatges que no aporten benefici o causen pèrdues, i per que es pugui desfer assignacions per explorar altres alternatives.

5 Estratègies de generació de la solució inicial

5.1 Importància de la solució inicial

La qualitat de la solució inicial té un impacte significatiu en el rendiment dels algoritmes de cerca local. Una solució inicial de qualitat pot reduir el temps de convergència i millorar la qualitat de la solució final obtinguda. És per això que hem explorat diverses estratègies per generar la solució inicial, avaluant el compromís entre la qualitat de la solució generada i el cost computacional d'obtenir-la.

5.2 Assignació greedy per Benefici

L'estratègia greedy per benefici consisteix en assignar iterativament les peticions més prometedores a cada camió, prioritzant aquelles que maximitzen el benefici mentre es respecten les restriccions del problema.

5.2.1 Descripció de l'algorisme

L'algorisme segueix un procediment constructiu que assigna peticions als camions de manera iterativa. Per cada camió disponible, es construeixen viatges de fins a dues peticions cadascun, respectant el límit màxim de viatges per dia. El procés es divideix en dues fases:

1. **Selecció de la primera petició:** Es busca la petició no assignada que proporciona el màxim benefici considerant la distància des del centre de distribució del camió.
2. **Selecció de la segona petició:** Un cop assignada la primera petició, es busca una segona petició compatible que minimitzi la distància addicional respecte a la primera, maximitzant així l'eficiència del viatge.

Aquest procés es repeteix fins que no es poden crear més viatges per al camió actual o s'ha assolit el límit de viatges diari. El criteri de selecció considera tant el benefici econòmic com la proximitat geogràfica entre peticions.

Algorithm 1 Generació de Solució Inicial Greedy

```
1: procedure GenerarSolucióGreedy
2:     for cada camió i en la flota do
3:         centro ← centre de distribució del camió i
4:         for j = 1 to MAX_VIATGES_DIA do
5:             viatge ← nou viatge buit
6:             p1 ← TrobarMillorPetició(centro, null)
7:             if p1 ≠ null then
8:                 Afegir p1 al viatge
9:                 Eliminar p1 de peticions no assignades
10:                p2 ← TrobarMillorPetició(centro, p1)
11:                if p2 ≠ null then
12:                    Afegir p2 al viatge
13:                    Eliminar p2 de peticions no assignades
14:                end if
15:                Afegir viatge a viatgesPerCamio[i]
16:            else
17:                break
18:            end if
19:        end for
20:    end for
21: end procedure
```

5.2.2 Justificació de l'estratègia

L'estratègia greedy per benefici presenta diversos avantatges que justifiquen la seva elecció:

- **Qualitat de la solució:** Aquest enfocament garanteix que cada decisió d'assignació considera el benefici de la petició, evitant solucions trivials o de baixa qualitat. La incorporació de la proximitat geogràfica en la selecció de la segona petició del viatge introdueix una heurística que tendeix a minimitzar els costos de transport, resultant en solucions que balancentegen benefici i eficiència operativa.

- **Eficiència computacional:** La complexitat temporal és $O(C \cdot V \cdot P)$, on C és el nombre de camions, V el nombre màxim de viatges per camió, i P el nombre de peticions. En la pràctica, aquesta complexitat es redueix a $O(C \cdot P)$ perquè V és una constant petita i les peticions s'eliminen de la llista de candidates.

- **Reproducibilitat:** Aquesta estratègia genera sempre la mateixa solució per a una instància donada, facilitant la comparació d'experiments i la depuració del codi.

- **Bona base per a cerca local:** Les solucions generades ja incorporen decisions raonables sobre l'agrupació de peticions, proporcionant un punt de partida sòlid per als operadors de cerca local.

5.3 Altres Estratègies Descartades

Vam considerar també la solució inicial buida la qual consisteix en iniciar sense cap assignació de peticions als camions, deixant que els operadors de cerca local construeixin la solució des de zero. Però la vam acabar descartent com bé s'explica a l'apartat d'experiments.

6 Funció heurística

6.1 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 s'aconsegueixi 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.

6.2 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:

6.2.1 Benefici de les peticions servides

Cada petició té associat un benefici que depèn dels dies que porta pendent. Si una petició s'atén 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 d'una solució és simplement la suma dels beneficis de totes les peticions servides:

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

6.2.2 Penalització de les peticions no servides

Quan una petició no s'atén en un dia determinat, això suposa una pèrdua potencial, ja que el benefici que s'hauria obtingut disminueix l'endemà. La penalització per no servir una petició es calcula com la diferència entre el benefici si s'atén avui i el benefici si s'atén l'endemà:

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

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

6.2.3 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:

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

6.3 Funcions heurístiques proposades

A partir d'aquests factors, s'han considerat diverses heurístiques, amb diferents graus de complexitat i precisió:

6.3.1 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 d'algoritmes de cerca com els de la biblioteca AIMA. Aquesta heurística és molt fàcil de calcular i reflecteix directament l'objectiu principal, però té la limitació que no penalitza explícitament les peticions que es deixen sense servir, cosa que podria fer que s'ignorin peticions amb molts dies d'espera si el seu cost és elevat.

6.3.2 Heurística H2: Benefici amb penalització

Una evolució de l'anterior 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.

6.4 Heurística escollida

Després d'analitzar els pros i contres, s'ha escollit H2 per als experiments principals:

$$h(\text{estat}) = -(B_{\text{servides}} - \cdot P_{\text{no servides}} - C_{\text{km}}) \tag{10}$$

Aquesta decisió es basa en el fet que H2 equilibra els tres objectius i és més completa que H1. No hem considerat afegir ponderacions, ja que tots els operadors tenen la mateixa unitat (€) i l'enunciat no especificava si era més important maximitzar ingressos o minimitzar pèrdues o costos. Sense ponderacions, els tres objectius tenen el mateix pes a l'hora de buscar una solució.

7 Experimentació

7.1 Metodologia experimental

7.1.1 Condicions generals

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

- **Maquinari**: AMD Ryzen™ 9 6900HX , 32GB RAM
- **Sistema operatiu**: Arch Linux
- **JVM**: OpenJDK 25.0.1
- **Repeticions**: 10 execucions per experiment amb llavors diferents
- **Estadístiques**: Mitjana i desviació estàndard

7.1.2 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 1: Escenari base per als experiments

7.2 Experiment 1: Comparació d'operadors

7.2.1 Objectiu

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

7.2.2 Configuració experimental

Per aquest experiment s'han definit cinc conjunts diferents d'operadors, cadascun amb una combinació específica que permet avaluar diferents estratègies d'exploració de l'espai de cerca:

- **Bàsics (Afegir + Crear):** Aquest conjunt utilitza només els operadors d'afegir peticions a viatges existents i crear nous viatges. És el conjunt més conservador, que permet construir solucions de manera incremental però amb capacitat limitada per reorganitzar-les.
- **Modificació (Afegir + Treure + Moure):** Inclou operadors que permeten afegir peticions, eliminar-les de viatges, i moure-les entre viatges diferents. Aquest conjunt se centra en la modificació de l'assignació de peticions sense canviar l'estructura global de viatges.
- **Només Moviments (Moure + Intercanviar):** Conté exclusivament operadors que redistribueixen peticions ja assignades: moure peticions entre viatges i intercanviar parelles de peticions. Aquest conjunt parteix d'una solució inicial prou bona i se centra en la seva reorganització.
- **Sense Intercanvi (Afegir + Treure + Moure + Crear):** Combina la majoria d'operadors excepte l'intercanvi de peticions. Ofereix una exploració àmplia de l'espai de cerca però sense la capacitat de fer bescanvis directes entre viatges.
- **Tots:** Inclou els cinc operadors disponibles (Afegir, Treure, Moure, Intercanviar i Crear), proporcionant la màxima flexibilitat per explorar l'espai de cerca.

7.2.3 Metodologia d'avaluació

De cara a avaluar la qualitat dels diferents conjunts d'operadors, hem considerat que la característica principal que han de tenir és la capacitat de millorar la solució inicial. Això es tradueix en analitzar la quantitat de nodes expandits per aquests conjunts d'operadors. Seguidament, per descartar entre conjunts que ofereixen resultats similars en quant a expansió de nodes, avaluem la seva capacitat de generar solucions amb beneficis econòmics alts, però amb el mínim temps d'execució possible.

Com que l'objectiu és analitzar la capacitat dels operadors per trobar millores, s'ha escollit com a estratègia de solució inicial la solució greedy, ja que parteix d'un estat raonablement bo i permet observar quins conjunts d'operadors són realment capaços de millorar-lo. Si s'hagués fet servir la solució buida, tots els conjunts haurien mostrat millores trivials, i no es podria distingir la seva qualitat real.

7.2.4 Anàlisi dels resultats

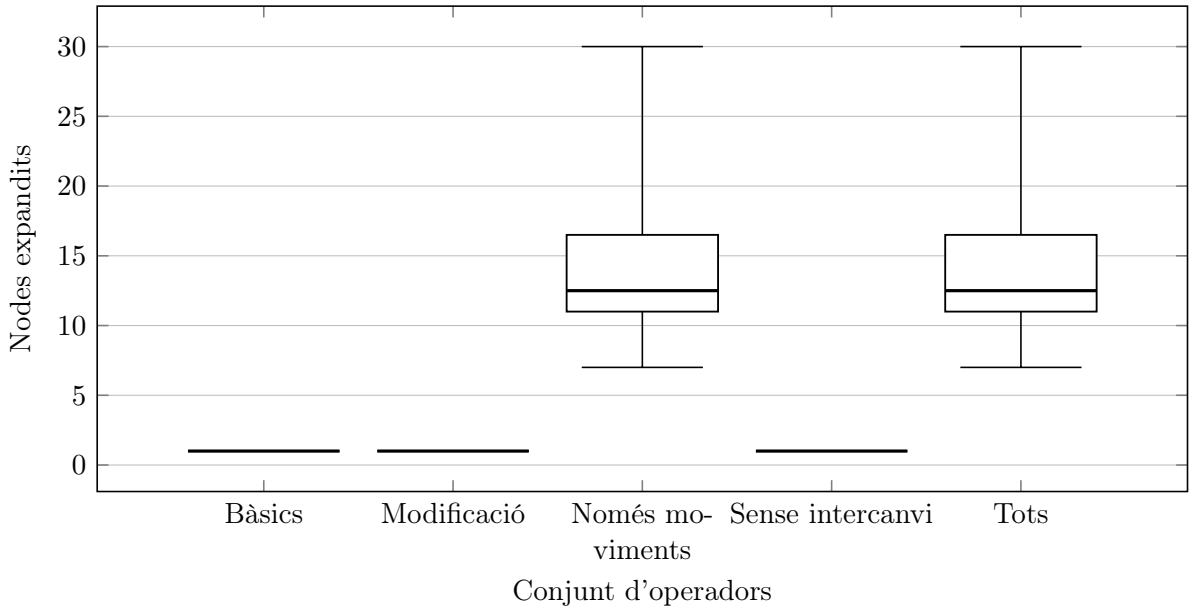

Figura 1: Comparació del nombre de nodes expandits segons el conjunt d'operadors

Capacitat d'exploració de l'espai de cerca Els resultats de la figura anterior mostren clarament una divisió entre els conjunts d'operadors. Tres dels conjunts (Bàsics, Modificació i Sense Intercanvi) expandeixen un únic node, indicant que queden atrapats en el primer màxim local que troben i no són capaços de millorar la solució inicial greedy. Això suggereix que aquests conjunts manquen dels operadors necessaris per escapar dels òptims locals.
En contrast, els conjunts Només Moviments i Tots expandeixen una mitjana d'aproximadament 12-13 nodes, amb una variabilitat que arriba fins a 30 nodes en alguns casos. Aquesta capacitat d'exploració més àmplia indica que aquests operadors són efectius per trobar millores sobre la solució inicial.

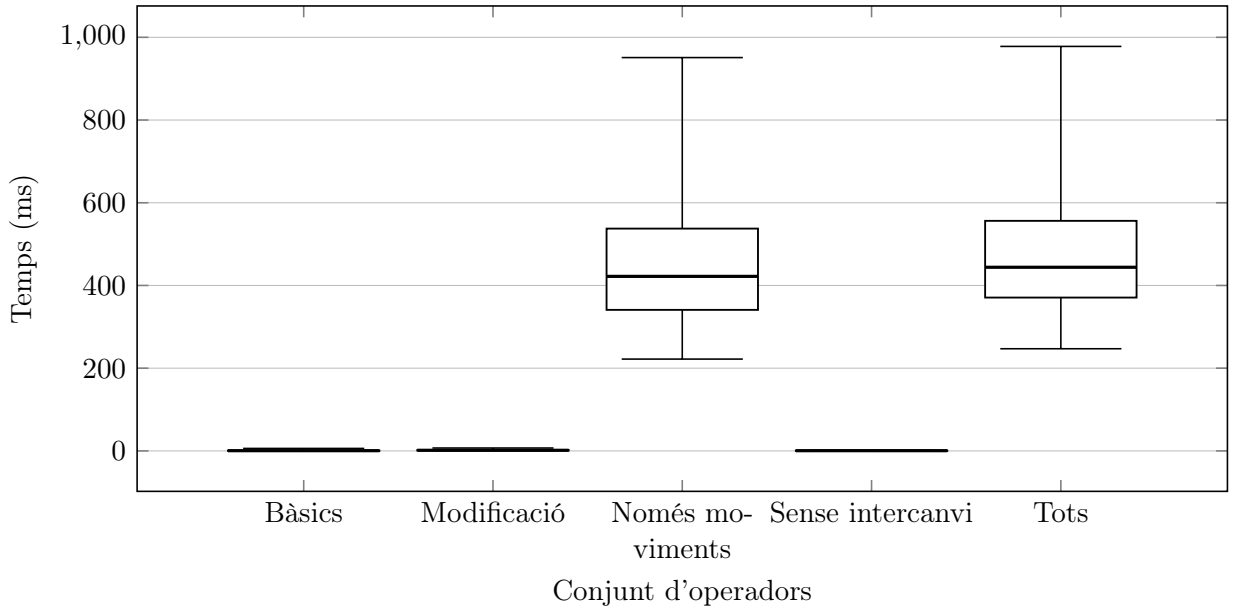

Figura 2: Comparació del temps d'execució segons el conjunt d'operadors

Cost computacional El temps d'execució reflecteix directament la capacitat d'exploració. Els tres conjunts que expandeixen un sol node tenen temps d'execució negligibles (menys d'1 ms), mentre que els conjunts Només Moviments i Tots requereixen entre 300 i 600 ms de mitjana, arribant a superar els 900 ms en el cas més desfavorable.
Aquest increment de temps és consistent amb l'augment de nodes expandits i representa el cost necessari per explorar l'espai de cerca i trobar solucions de millor qualitat.

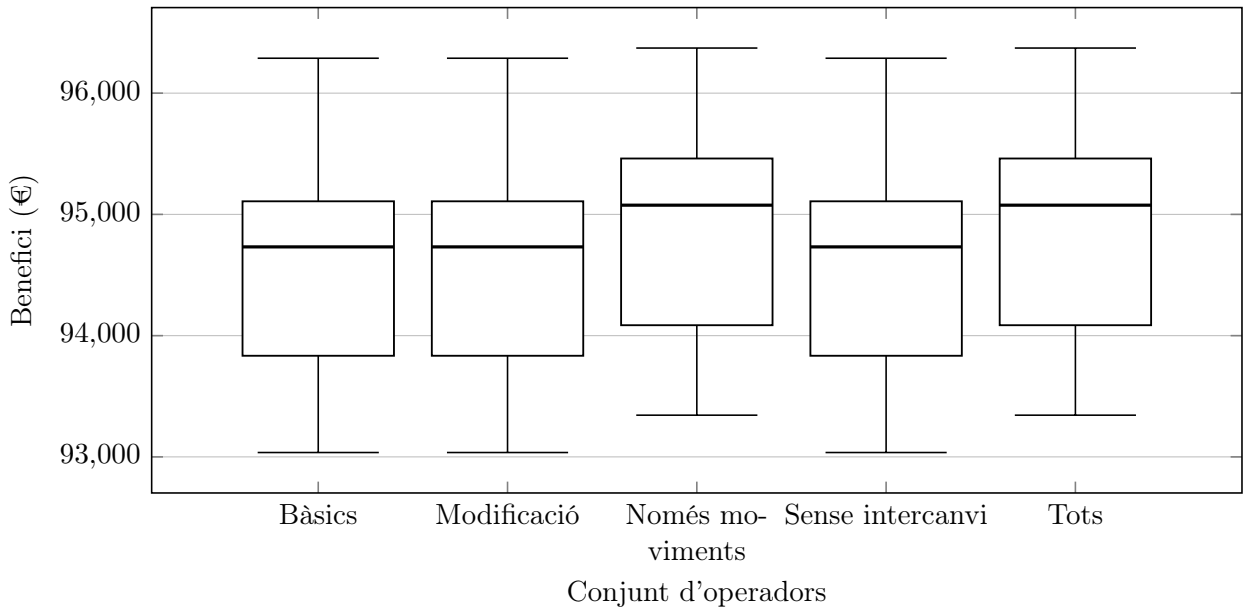

Figura 3: Comparació del benefici econòmic segons el conjunt d'operadors

Qualitat de les solucions La qualitat de les solucions, mesurada pel benefici econòmic obtingut, correlaciona directament amb la capacitat d'exploració. Els conjunts que només expandeixen un node obtenen una mediana de benefici de 94.732€, corresponent essencialment a la solució inicial greedy sense millores.
Els conjunts Només Moviments i Tots aconsegueixen un benefici significativament superior, amb una mediana de 95.076€. Aquest increment de 344€ (aproximadament un 0,36% de millora) demostra que aquests operadors són efectius per escapar dels màxims locals i trobar solucions de millor qualitat.

7.2.5 Conclusions

Els resultats d'aquest experiment revelen clarament que no tots els conjunts d'operadors són igualment efectius per a la cerca local amb Hill Climbing. Els operadors de moviment (moure peticions entre viatges i intercanviar-les) són essencials per explorar l'espai de cerca de manera efectiva quan es parteix d'una solució inicial de qualitat.
Entre els dos conjunts que mostren bons resultats, el conjunt *Només Moviments* resulta ser el més eficient, ja que obté un rendiment pràcticament idèntic al conjunt complet però amb un cost computacional lleugerament menor (mediana de 422 ms versus 444 ms). Això suggereix que, un cop es té una bona solució inicial, els operadors de reorganització són suficients i més eficients que mantenir operadors de construcció i destrucció.
Per tant, per als següents experiments s'utilitzarà preferentment el conjunt *Només Moviments*, ja que ofereix el millor compromís entre qualitat de solució i eficiència computacional.

7.3 Experiment 2: Comparació de solucions inicials

7.3.1 Objectiu

Determinar quina estratègia de generació de la solució inicial és més adequada per a l'algorisme Hill Climbing.

7.3.2 Configuració experimental

Per aquest experiment s'han comparat dues estratègies de generació de la solució inicial:

- **Solució buida:** S'inicia sense cap petició assignada a cap viatge. L'algorisme ha de construir la solució completament des de zero utilitzant els operadors disponibles.
- **Solució greedy:** S'utilitza l'algorisme greedy descrit a la secció 5 per generar una solució inicial de qualitat acceptable abans d'aplicar Hill Climbing.

Consideració sobre el conjunt d'operadors Tot i que a l'enunciat s'indica que cal fixar el conjunt d'operadors escollit a l'experiment 1, en aquest cas s'ha fet una excepció: s'ha emprat el conjunt complet d'operadors per a ambdues estratègies. Aquesta decisió

es justifica perquè la solució buida necessita operadors d'inserció (afegir i crear) per poder construir una solució viable. El conjunt guanyador de l'experiment anterior (només moviments) no permetria cap millora si es partís d'una solució buida, ja que no inclou operadors d'inserció de peticions i per tant no podria construir cap viatge.

7.3.3 Anàlisi dels resultats

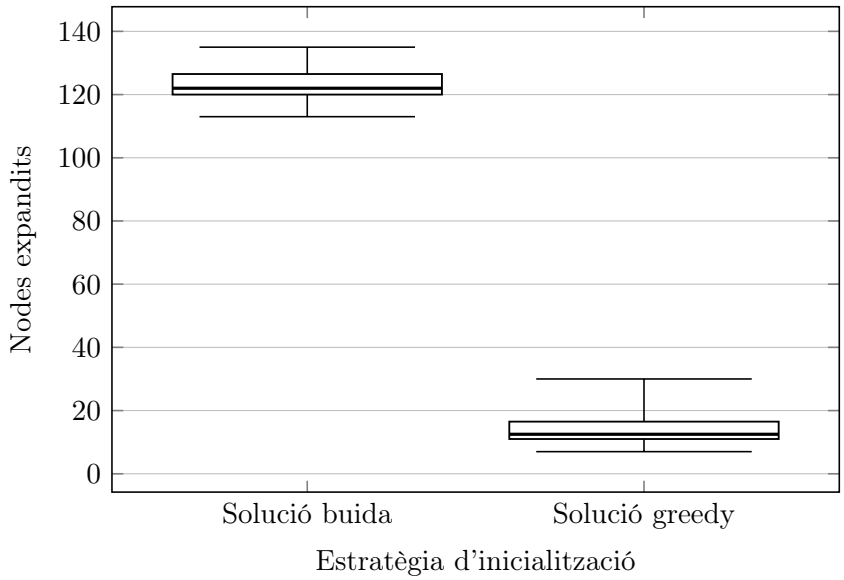

Figura 4: Nodes expandits per Hill Climbing segons la inicialització

Exploració de l'espai de cerca La diferència en l'exploració de l'espai de cerca entre les dues estratègies és dramàtica. La solució buida expandeix una mitjana de 122 nodes, amb una variabilitat que va dels 113 als 135 nodes. Aquest nombre tan elevat s'explica perquè l'algorisme ha de construir la solució des de zero, explorant nombroses combinacions d'assignacions de peticions.
En contrast, la solució greedy expandeix només 12,5 nodes de mitjana, aproximadament 10 vegades menys. Això indica que partir d'una solució inicial razonable permet a l'algorisme convergir molt més ràpidament cap a un òptim local, ja que requereix menys transformacions per trobar una solució satisfactòria.

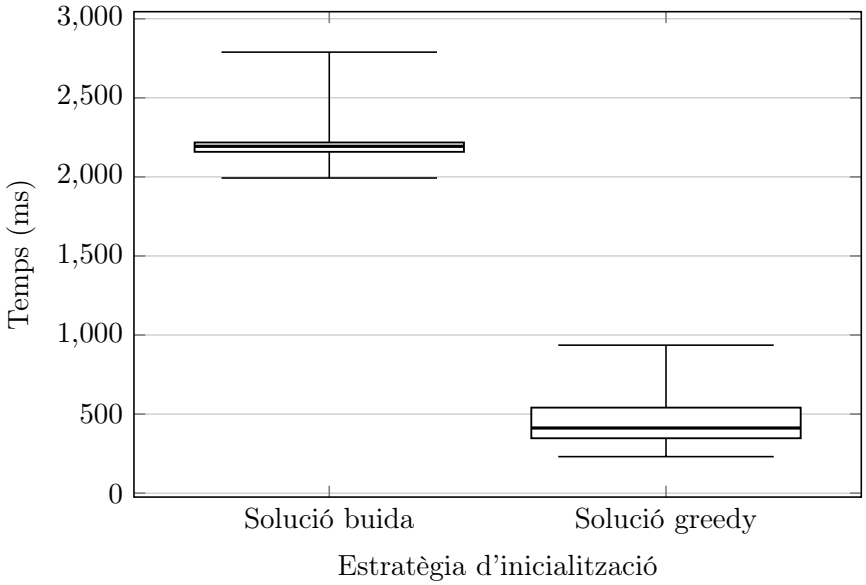

Figura 5: Temps d'execució segons la inicialització

Cost computacional El temps d'execució reflecteix directament la diferència en nodes expandits. La solució buida requereix una mediana de 2.193 ms, amb un rang que va dels 1.993 als 2.789 ms. Aquest temps elevat és el cost de construir una solució completa mitjançant cerca local.
La solució greedy, en canvi, s'executa en només 412 ms de mediana, aproximadament 5 vegades més ràpid. Aquest temps inclou tant la generació de la solució inicial greedy com la posterior millora amb Hill Climbing. La reducció de temps és significativa i fa que aquesta estratègia sigui molt més pràctica per a aplicacions on el temps de resposta és important.

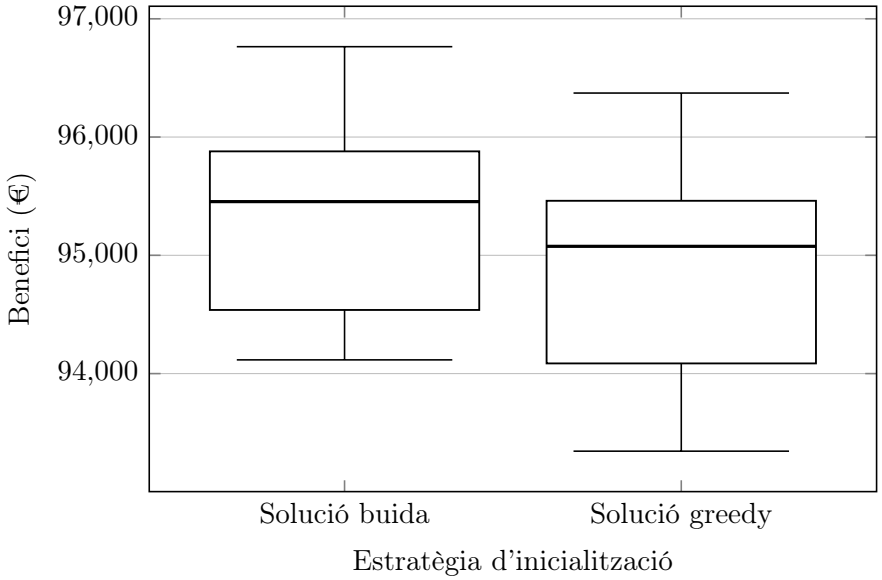

Figura 6: Benefici econòmic obtingut per Hill Climbing segons la inicialització

Qualitat de les solucions Sorprenentment, malgrat l'enorme diferència en nodes expandits i temps d'execució, la qualitat de les solucions finals és molt similar. La solució buida obté una mediana de 95.454€, mentre que la solució greedy obté 95.076€, una diferència de només 378€ (aproximadament un 0,4%).
Aquest resultat és particularment rellevant: la solució buida explora molt més l'espai de cerca però només aconsegueix una millora marginal en el benefici. Això suggereix que la solució greedy ja es troba en una regió de l'espai de cerca propera a l'òptim, i que l'exploració addicional de la solució buida no compensa el cost computacional afegit.

7.3.4 Conclusions

Els resultats d'aquest experiment demostren clarament que la solució greedy és l'estratègia més adequada per a la generació de la solució inicial. Tot i que la solució buida és capaç d'explorar més extensament l'espai de cerca i obtenir beneficis lleugerament superiors, aquesta millora marginal (0,4%) no justifica l'augment de 5 vegades en el temps d'execució. La solució greedy ofereix el millor compromís entre qualitat i eficiència: genera solucions de qualitat molt propera a l'òptim en una fracció del temps. A més, quan es combina amb el conjunt d'operadors guanyador de l'experiment 1 (només moviments), s'obté un temps d'execució encara millor mantenint la qualitat de les solucions.
Per tant, per als experiments posteriors s'utilitzarà la solució greedy com a estratègia d'inicialització estàndard, combinada amb el conjunt d'operadors només moviments quan l'objectiu sigui optimitzar el rendiment computacional.

7.4 Experiment 3: Ajust de paràmetres del Simulated Annealing

7.4.1 Objectiu

Trobar els paràmetres òptims per a l'algorisme Simulated Annealing en el nostre problema de distribució de combustible.

7.4.2 Configuració experimental

Per aquest experiment s'han explorat sistemàticament diferents combinacions dels paràmetres clau de Simulated Annealing:

- **Iteracions:** Nombre total d'iteracions de l'algorisme. S'han provat 1000, 5000 i 10000 iteracions.
- k**:** Factor d'escala de la temperatura inicial. S'han provat valors de 5, 25 i 125.
- λ**:** Taxa de refredament exponencial. S'han provat valors de 0,0001, 0,001 i 0,01.

Això resulta en 27 combinacions diferents, cadascuna executada 10 vegades per garantir la validesa estadística dels resultats. S'ha utilitzat el conjunt d'operadors només moviments identificat com a òptim en l'experiment 1.

7.4.3 Anàlisi dels resultats

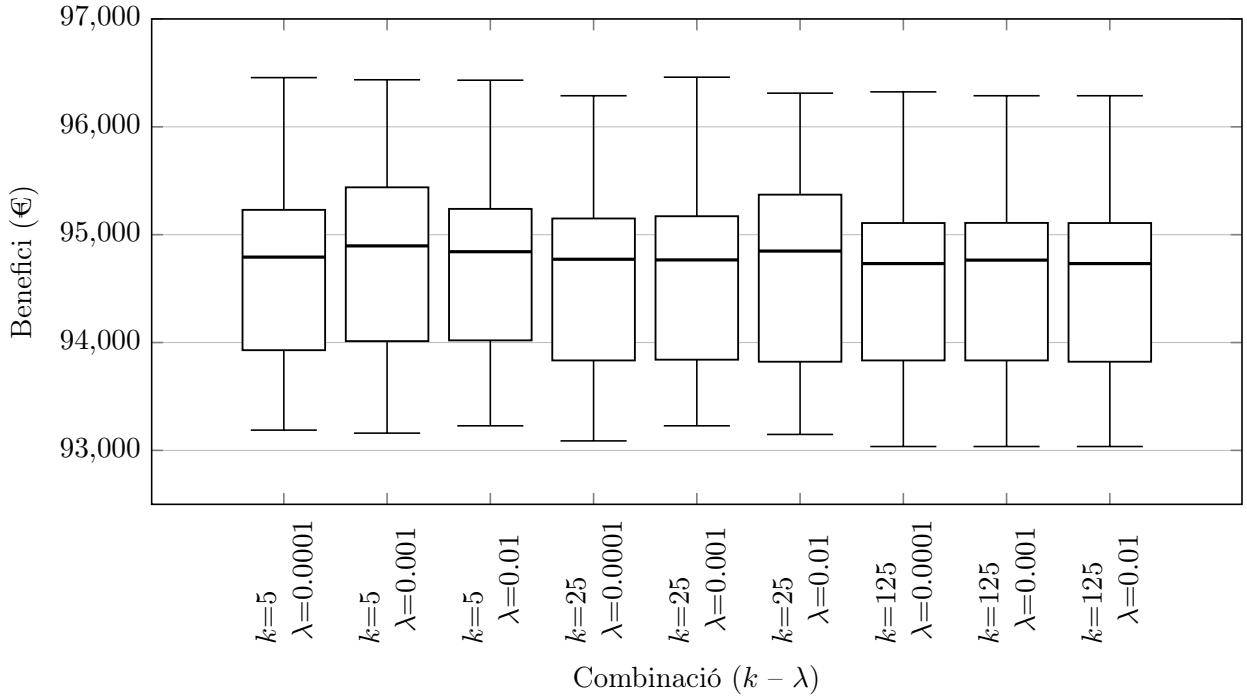

Figura 7: Distribució del benefici per combinació de k i λ amb 1000 iteracions (Simulated Annealing)

Efecte del nombre d'iteracions amb 1000 iteracions Amb només 1000 iteracions, totes les combinacions de paràmetres generen resultats molt similars, amb una mediana al voltant dels 94.700-94.900€. Aquesta homogeneïtat indica que l'algorisme no té temps suficient per explorar adequadament l'espai de cerca, independentment dels valors de k i λ escollits. Els beneficis obtinguts són lleugerament inferiors als assolits per Hill Climbing en l'experiment 1 (95.076€).

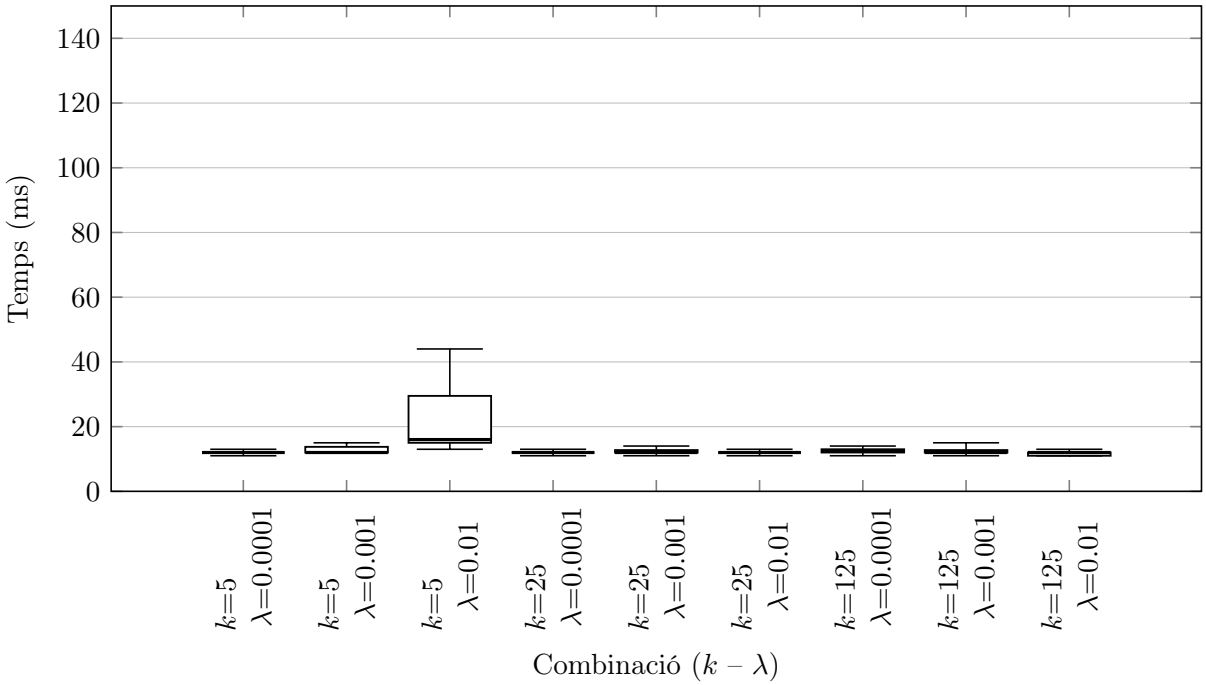

Figura 8: Distribució del temps d'execució per combinació de k i λ amb 1000 iteracions (Simulated Annealing)

El temps d'execució és extremadament baix per a totes les combinacions, amb una mediana al voltant dels 12 ms. Destaca només la combinació $k = 5, \lambda = 0.01$ amb temps lleugerament superiors, però l'impacte computacional és negligible en tots els casos.

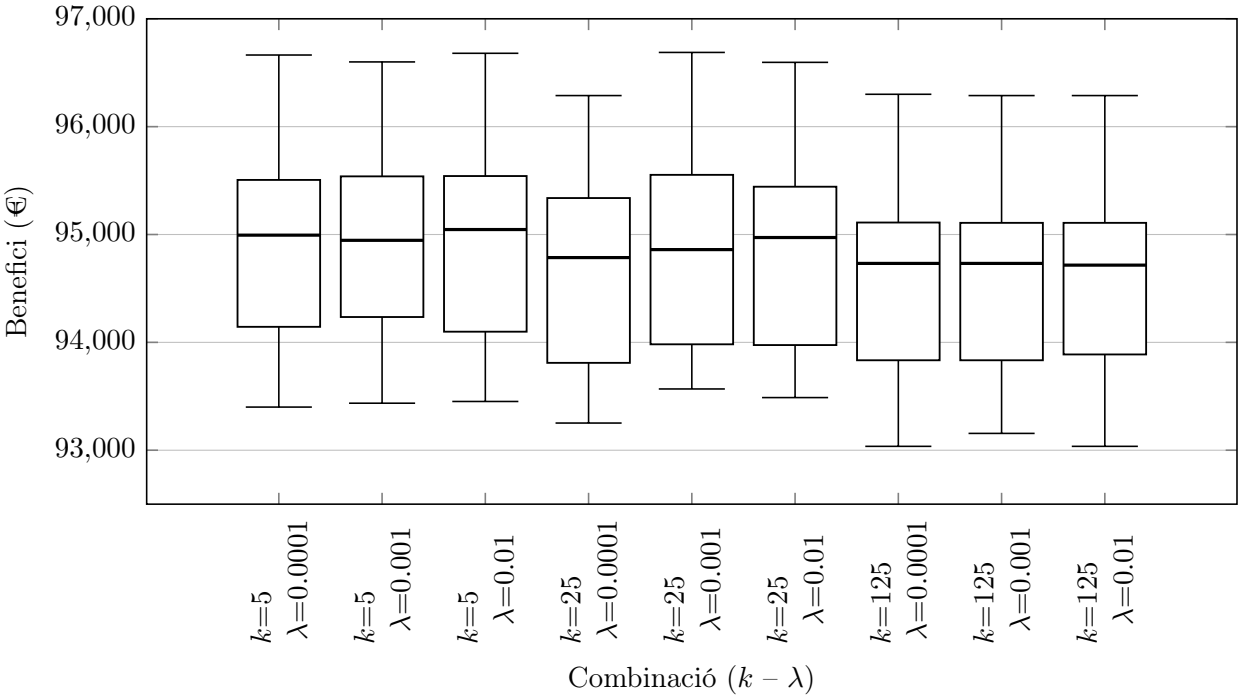

Figura 9: Distribució del benefici per combinació de k i λ amb 5000 iteracions (Simulated Annealing)

Efecte del nombre d'iteracions amb 5000 iteracions Amb 5000 iteracions, comencen a aparèixer diferències més clares entre les combinacions. Les configuracions amb valors baixos de k (5) tendeixen a obtenir beneficis lleugerament superiors (al voltant dels 95.000€), mentre que valors alts de k (125) mantenen resultats similars al cas de 1000 iteracions. Això suggereix que temperatures inicials massa altes impedeixen que l'algorisme convergeixi adequadament dins del pressupost d'iteracions disponible.

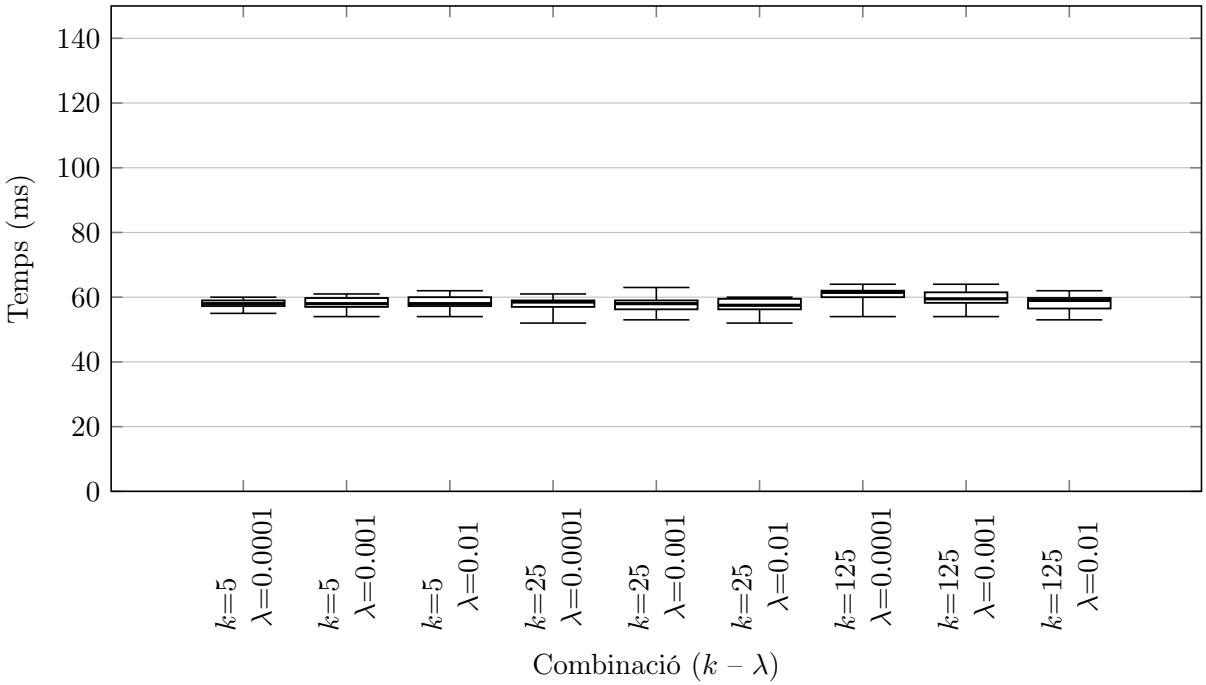

Figura 10: Distribució del temps d'execució per combinació de k i λ amb 1000 iteracions (Simulated Annealing)

El temps d'execució es manté notablement consistent entre totes les combinacions, situant-se al voltant dels 57-62 ms. Aquesta uniformitat indica que el cost computacional depèn principalment del nombre d'iteracions i no dels paràmetres específics de temperatura.

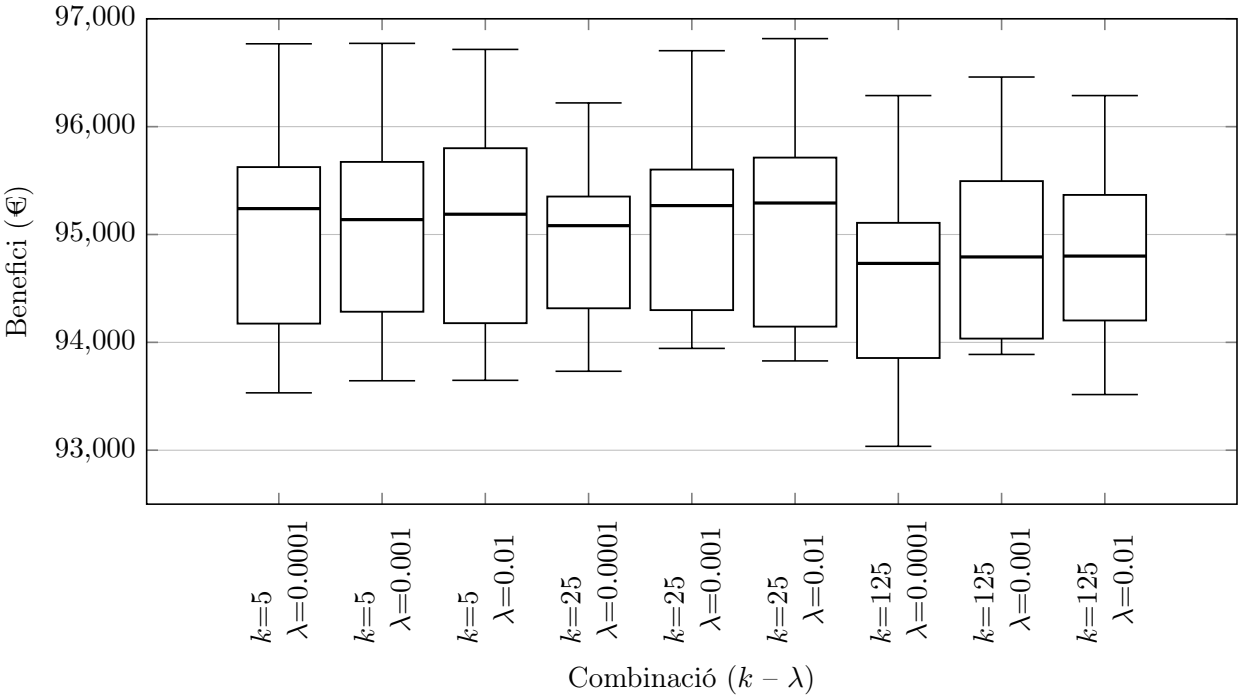

Figura 11: Distribució del benefici per combinació de k i λ amb 10000 iteracions (Simulated Annealing)

Efecte del nombre d'iteracions amb 10000 iteracions Amb 10000 iteracions, s'observa una millora clara i consistent en els beneficis obtinguts. Les millors combinacions són aquelles amb $k = 5$ i $k = 25$, independentment del valor de λ, assolint medianes al voltant dels 95.200-95.300€. Destaca especialment la combinació $k = 25, \lambda = 0.01$, que obté una mediana de 95.292€, superant lleugerament els resultats de Hill Climbing.
Les combinacions amb $k = 125$ segueixen mostrant resultats inferiors, confirmant que temperatures inicials excessivament altes no són adequades per aquest problema, probablement perquè l'exploració inicial és massa aleatòria i impedeix una convergència eficient.

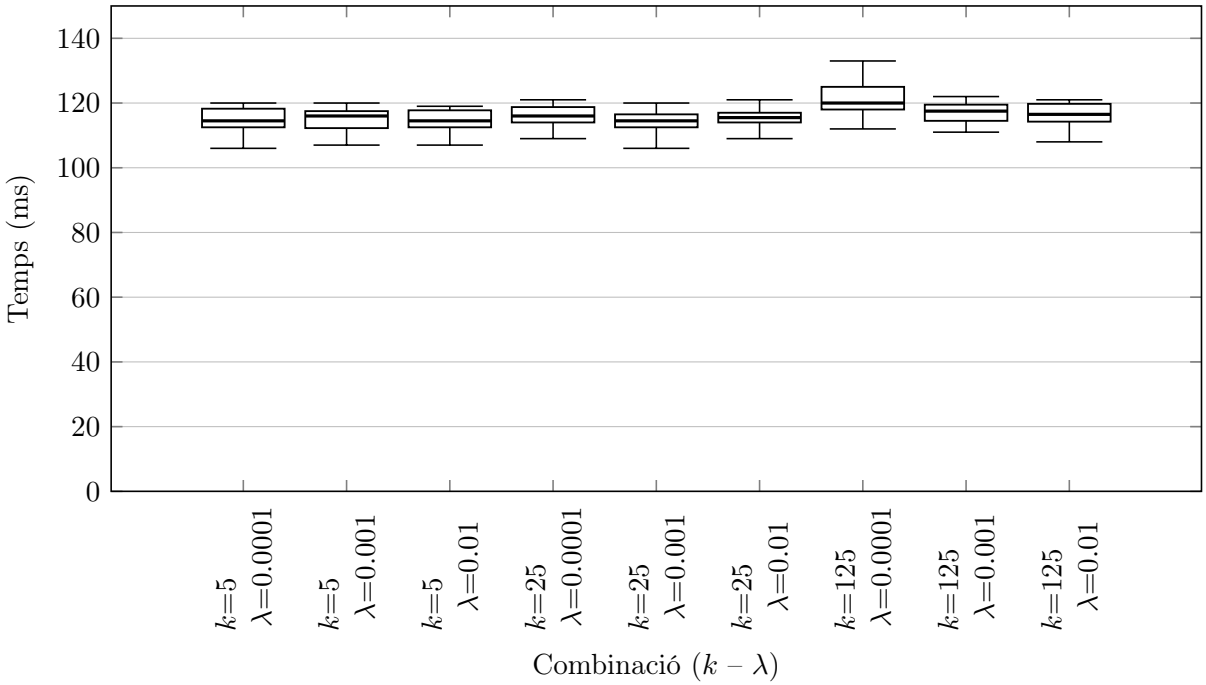

Figura 12: Distribució del temps d'execució per combinació de k i λ amb 1000 iteracions (Simulated Annealing)

El temps d'execució amb 10000 iteracions se situa uniformement entre 114-120 ms per a totes les combinacions. Aquest cost temporal és aproximadament el doble que amb 5000 iteracions, reflectint la relació lineal esperada entre nombre d'iteracions i temps de càlcul.

7.4.4 Conclusions

Els resultats d'aquest experiment demostren que el nombre d'iteracions és el factor més crític per a l'èxit de Simulated Annealing en aquest problema. Amb 1000 o 5000 iteracions, l'algorisme no té temps suficient per explorar adequadament l'espai de cerca, mentre que amb 10000 iteracions s'observen millores consistents.
Pel que fa als paràmetres de temperatura, valors baixos de k (5 o 25) són preferibles, ja que permeten una exploració inicial controlada seguida d'una convergència efectiva. El paràmetre λ mostra menys impacte, amb tots els valors provats oferint resultats similars quan el nombre d'iteracions és suficient.
La configuració òptima identificada és **10000 iteracions,** $k = 25$**,** $\lambda = 0.01$, que proporciona els millors beneficis econòmics de manera consistent (mediana de 95.292€). Tot i que aquest increment de benefici respecte a Hill Climbing és modest (aproximadament un 0,2%), el cost temporal addicional (115 ms versus 412 ms per Hill Climbing) és perfectament assumible i fa que aquesta configuració sigui adequada per a aplicacions on es prioritza la qualitat de la solució sobre la velocitat d'execució.

7.5 Experiment 4: Escalabilitat temporal

7.5.1 Objectiu

Aquest experiment té com a objectiu analitzar com creix el temps d'execució dels algorismes Hill Climbing i Simulated Annealing quan augmenta l'escalabilitat del problema, és a dir, el nombre de centres de distribució i gasolineres. S'ha mantingut la proporció constant de 10 gasolineres per cada centre de distribució, avaluant configuracions des de 10-100 fins a 50-500.

7.5.2 Configuració experimental

S'han provat cinc configuracions d'escalabilitat creixent:

- 10 centres, 100 gasolineres (escenari base)
- 20 centres, 200 gasolineres (2x)
- 30 centres, 300 gasolineres (3x)
- 40 centres, 400 gasolineres (4x)
- 50 centres, 500 gasolineres (5x)

Per a cada configuració s'han executat 10 repeticions amb Hill Climbing (conjunt d'operadors només moviments) i Simulated Annealing (paràmetres òptims de l'experiment 3: 10000 iteracions, $k = 25$, $\lambda = 0.01$).

7.5.3 Anàlisi dels resultats

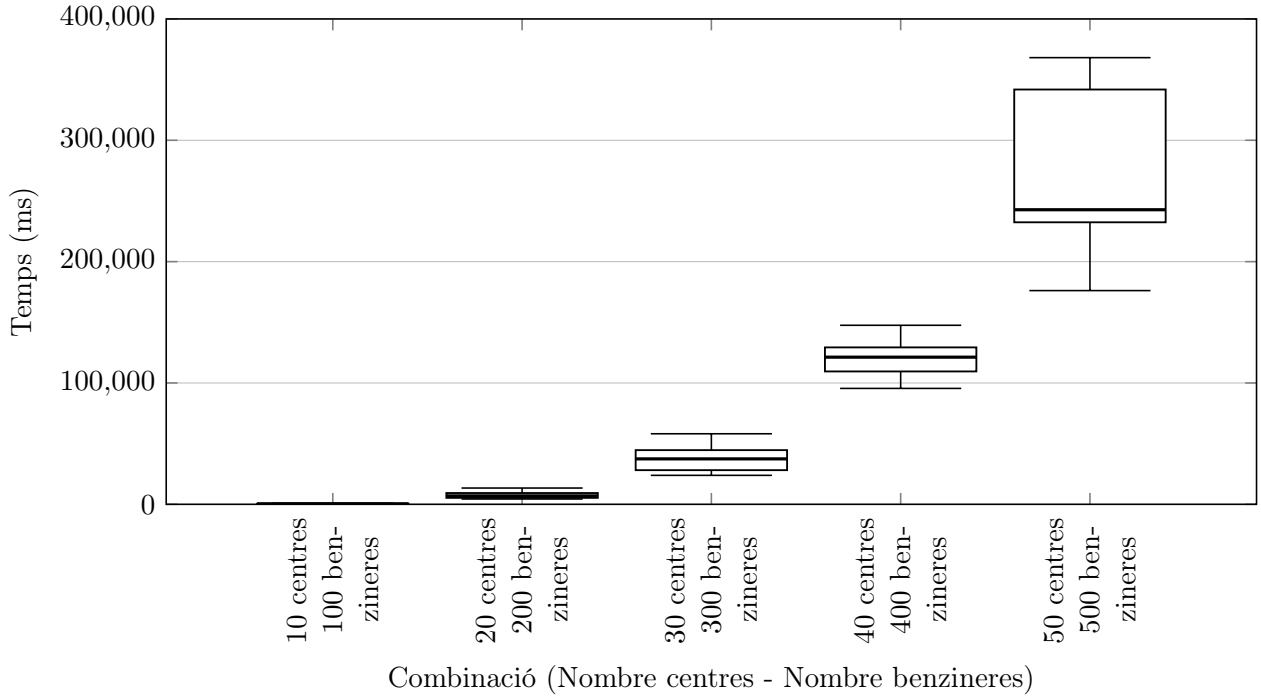

Figura 13: Evolució temporal de l'algorisme Hill Climbing al augmentar l'escalabilitat del problema

Escalabilitat temporal de Hill Climbing El temps d'execució de Hill Climbing creix de manera clarament no lineal amb l'augment de l'escalabilitat del problema. Partint d'uns 417 ms per a l'escenari base (10-100), el temps es multiplica aproximadament per 16 quan arribem als 20 centres i 200 gasolineres (6.965 ms), i continua creixent exponencialment fins a superar els 240.000 ms (més de 4 minuts) per a l'escenari més gran (50-500).
Aquest comportament és esperable en Hill Climbing, ja que l'algorisme ha d'explorar un espai de cerca que creix combinatòriament amb el nombre de peticions, viatges i camions. Cada estat té molts més veïns possibles a mesura que augmenta la mida del problema, i l'algorisme avalua exhaustivament tots els successors fins a trobar un màxim local.

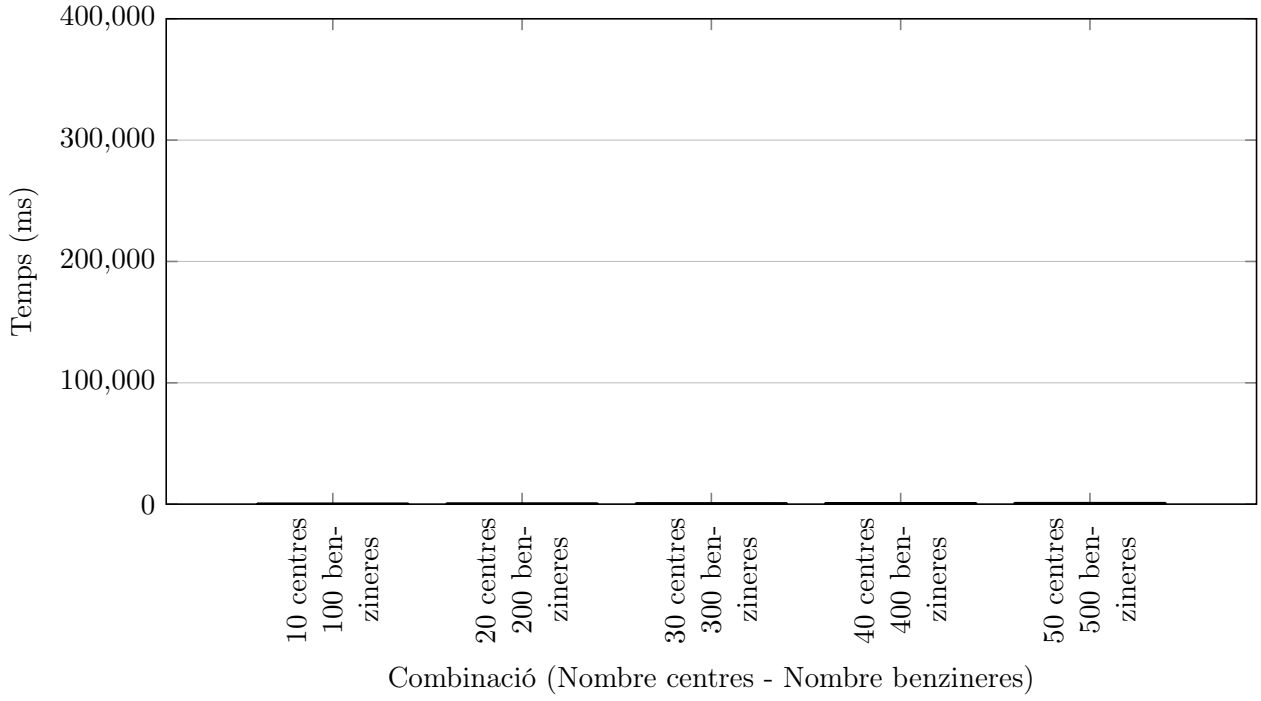

Figura 14: Evolució temporal de l'algorisme Simulated Annealing al augmentar l'escalabilitat del problema

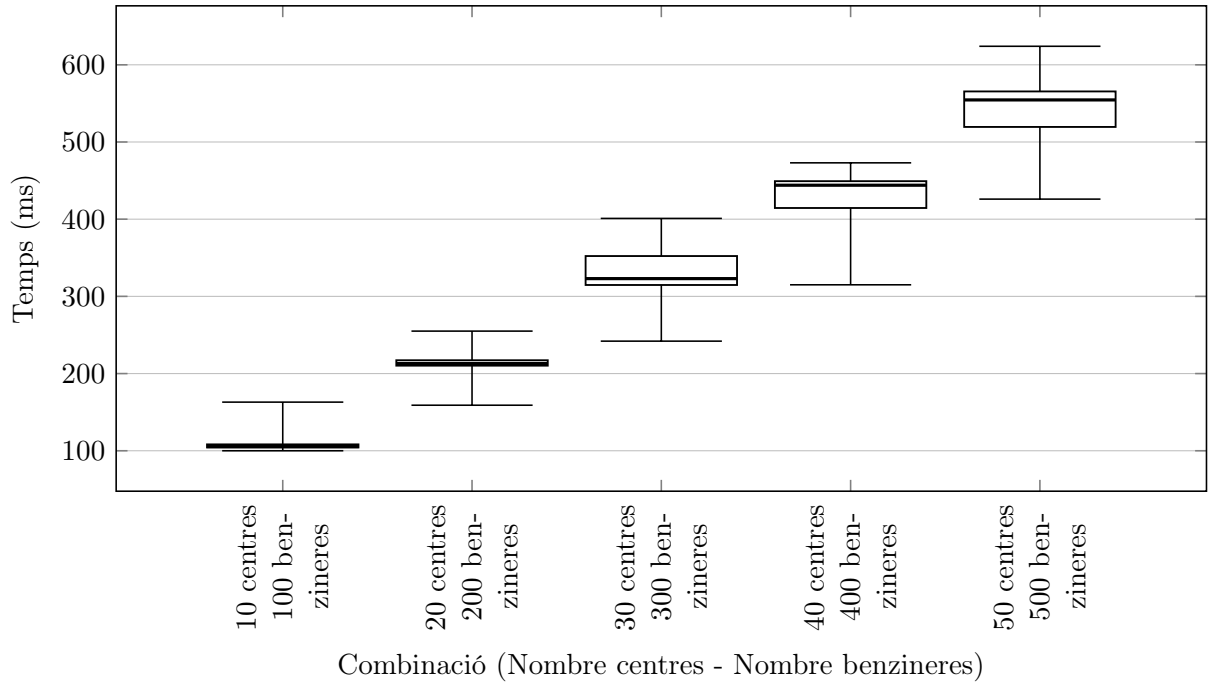

Figura 15: Evolució temporal de l'algorisme Simulated Annealing al augmentar l'escalabilitat del problema (ampliació de l'escala)

Escalabilitat temporal de Simulated Annealing El temps d'execució de Simulated Annealing creix de manera pràcticament lineal: passa de 105 ms per a l'escenari base a 554 ms per a l'escenari més gran, només un factor de 5,2x per un problema 5 vegades més gran. Aquest creixement molt més controlat s'explica pel fet que Simulated Annealing executa un nombre fix d'iteracions (10.000) independentment de la mida del problema, i cada iteració avalua només un successor aleatori en lloc de tots els successors possibles. La diferència d'escalabilitat és dramàtica: mentre que Hill Climbing es torna completament inviable per a problemes grans (més de 4 minuts per 50 centres), Simulated Annealing es manté en menys d'un segon fins i tot per als escenaris més grans.

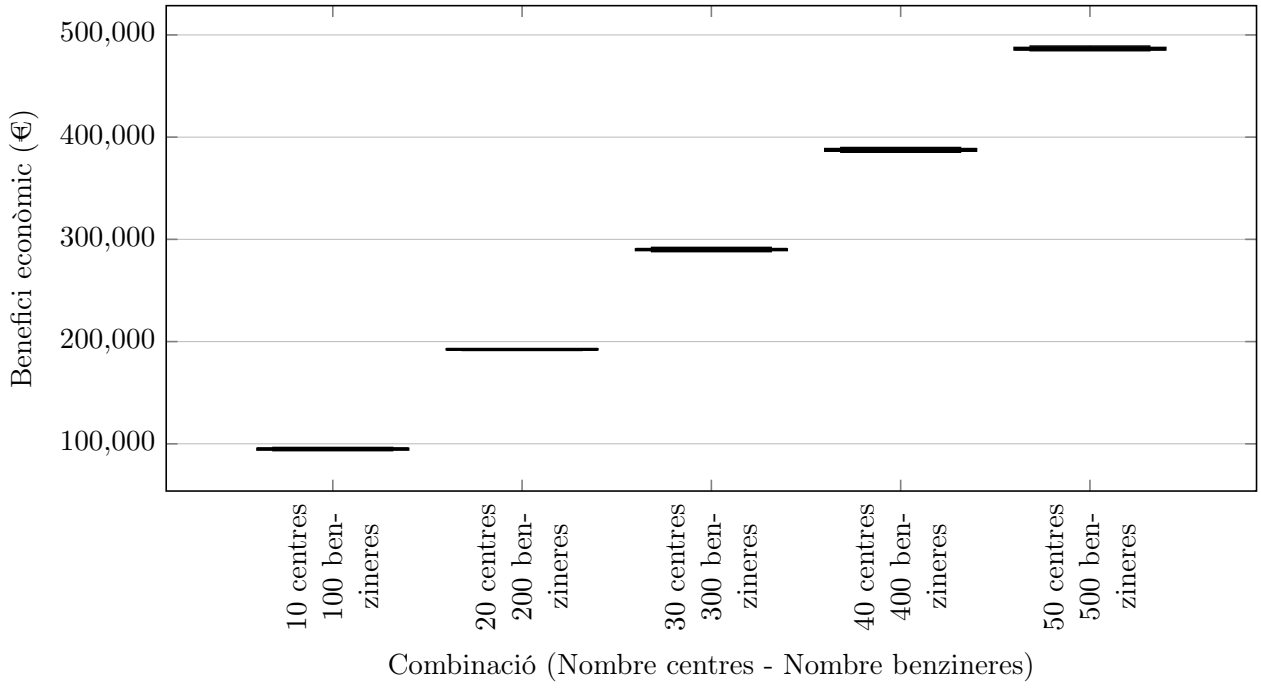

Figura 16: Evolució del benefici econòmic de l'algorisme Hill Climbing al augmentar l'escalabilitat del problema

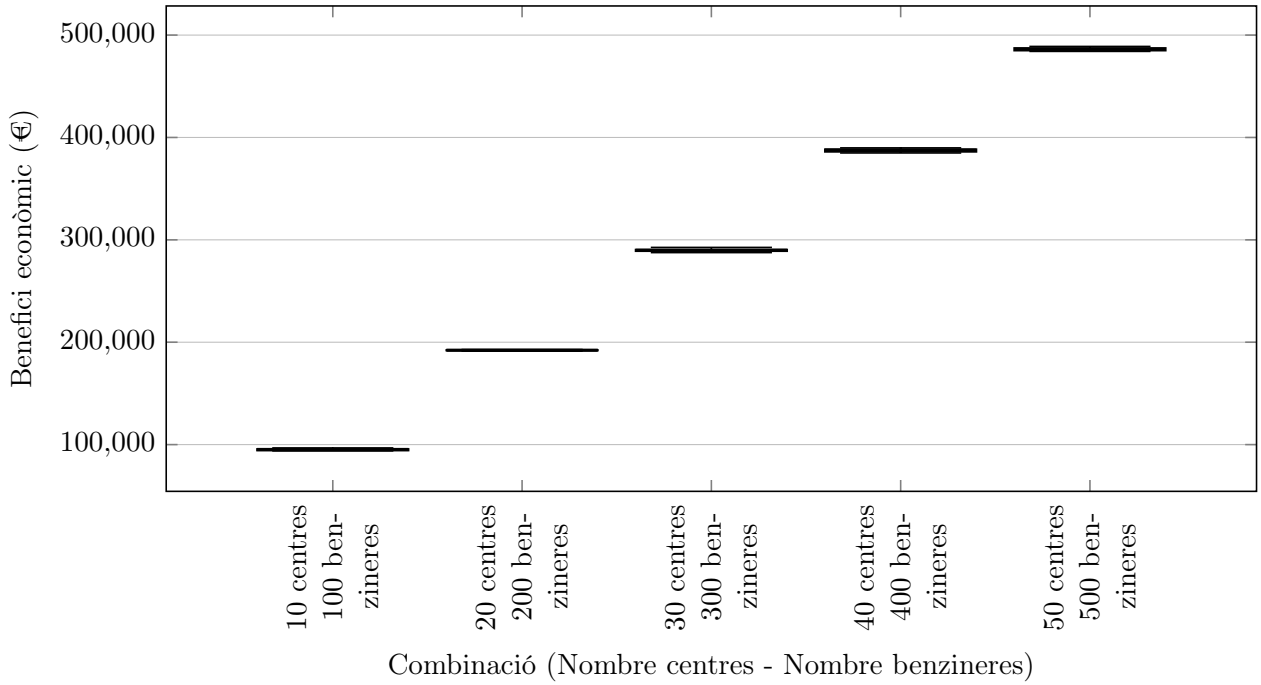

Figura 17: Evolució del benefici econòmic de l'algorisme Simulated Annealing al augmentar l'escalabilitat del problema

Comparació de la qualitat de les solucions Analitzant la qualitat de les solucions obtingudes, observem que ambdós algorismes generen beneficis molt similars per a cada configuració d'escalabilitat. Per exemple, per a l'escenari de 50 centres i 500 gasolineres, Hill Climbing obté una mediana de 486.234€ mentre que Simulated Annealing obté 486.144€, una diferència de menys del 0,02%.
Aquest resultat és especialment rellevant perquè demostra que la millora marginal en qualitat que ofereix Hill Climbing (quan la ofereix) no justifica en absolut l'augment massiu del cost temporal. Mentre que per a l'escenari més gran Hill Climbing requereix més de 240 segons, Simulated Annealing obté resultats pràcticament idèntics en menys d'un segon, resultant més de 400 vegades més ràpid.

7.5.4 Conclusions

Aquest experiment revela clarament les limitacions de Hill Climbing per a problemes d'escalabilitat mitjana o gran. El seu creixement temporal exponencial el fa inviable per a aplicacions pràctiques amb més de 20-30 centres de distribució, mentre que Simulated Annealing manté un rendiment predictible i pràctic fins i tot per a problemes 5 vegades més grans.
La clau d'aquesta diferència rau en l'estratègia d'exploració: Hill Climbing avalua exhaustivament tots els veïns en cada iteració, mentre que Simulated Annealing treballa amb un pressupost fix d'iteracions avaluant només un veí aleatori per iteració. Aquesta aproximació probabilística permet a Simulated Annealing escalar molt millor, sacrificant només una ínfima fracció de qualitat de solució.

Per tant, per a aplicacions reals on l'escalabilitat és una preocupació, Simulated Annealing amb els paràmetres identificats en l'experiment 3 és clarament l'elecció preferible, oferint el millor compromís entre qualitat de solució i eficiència computacional.

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

7.6.1 Objectiu

Analitzar l'impacte de concentrar els camions en menys centres, mantenint el nombre total de camions constant (10 camions en total), per tal d'avaluar els efectes en l'eficiència operativa, el benefici econòmic i la capacitat de resposta del servei.

7.6.2 Configuració

S'han comparat dos escenaris diferents amb el mateix nombre total de camions (10) però amb diferent distribució:

- **Escenari A**: 10 centres amb 1 camió per centre
- **Escenari B**: 5 centres amb 2 camions per centre

L'experiment s'ha executat 10 vegades per a cada escenari per garantir la significació estadística dels resultats. S'han analitzat tres mètriques principals: benefici econòmic, quilòmetres recorreguts i percentatge de peticions servides.

7.6.3 Resultats i Anàlisi

Benefici Econòmic Com es pot observar a la Figura 18, l'escenari amb 10 centres presenta un benefici econòmic lleugerament superior (mitjana: 95 mil€) respecte a l'escenari amb 5 centres (mitjana: 94mil€). La diferència és estadísticament significativa, amb una dispersió menor en el cas dels 10 centres, indicant major consistència en els resultats.

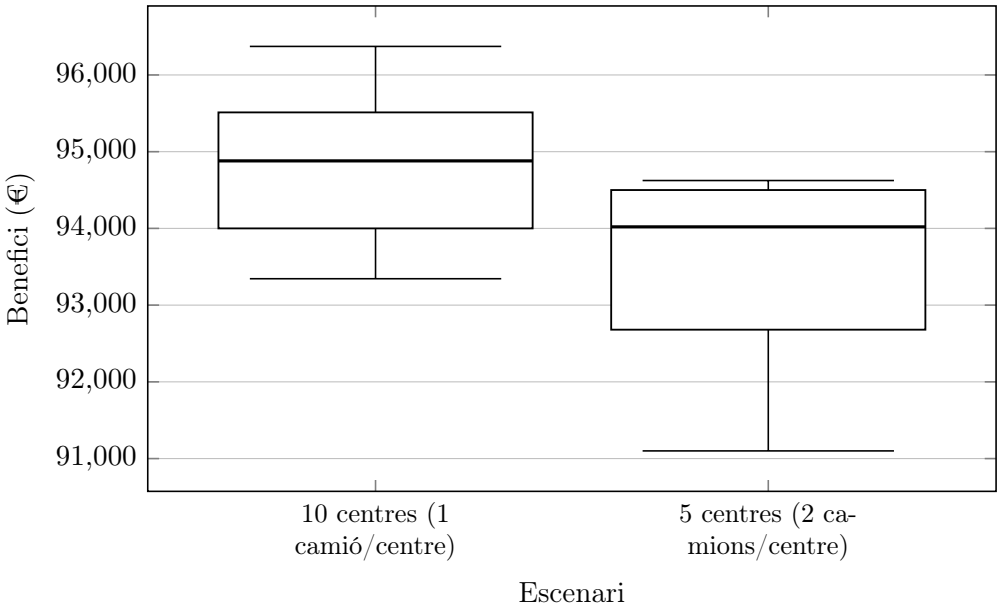

Figura 18: Comparació del benefici econòmic entre ambdós escenaris

Quilòmetres Recorreguts La Figura 19 mostra una diferència notable en la distància total recorreguda. L'escenari amb 5 centres presenta un major recorregut mitjà (2.900 km) respecte als 10 centres (2.600 km). Això representa un increment del 12.5% en la distància recorreguda quan es concentren els camions en menys centres.

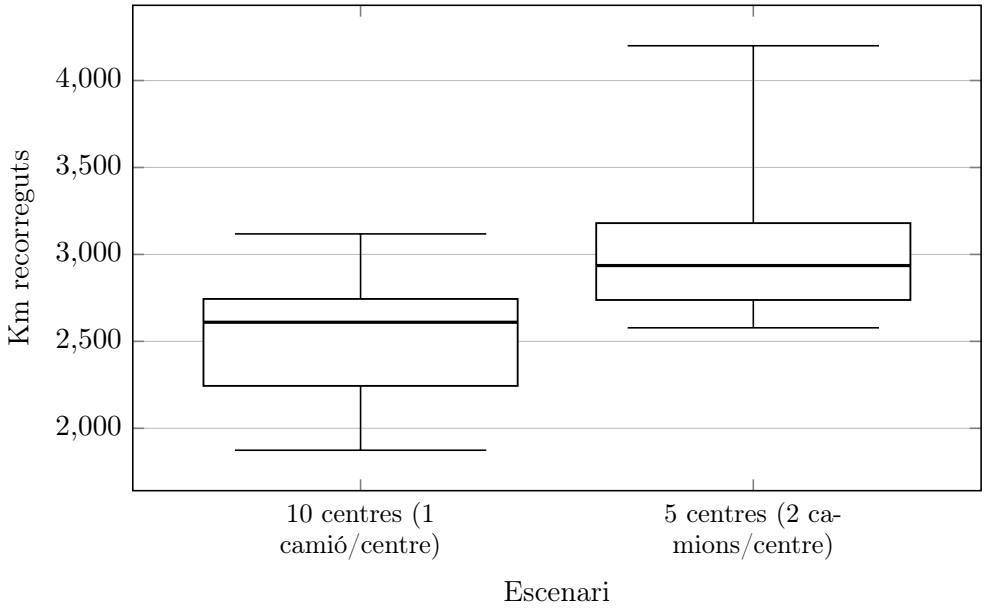

Figura 19: Comparació dels km recorreguts entre ambdós escenaris

Peticions Servides Com es veu a la Figura 20, ambdós escenaris serveixen el 100% de les peticions en totes les execucions, demostrant que la capacitat de resposta no es veu afectada per la distribució dels camions i centres, i igualment totes les peticions son servides

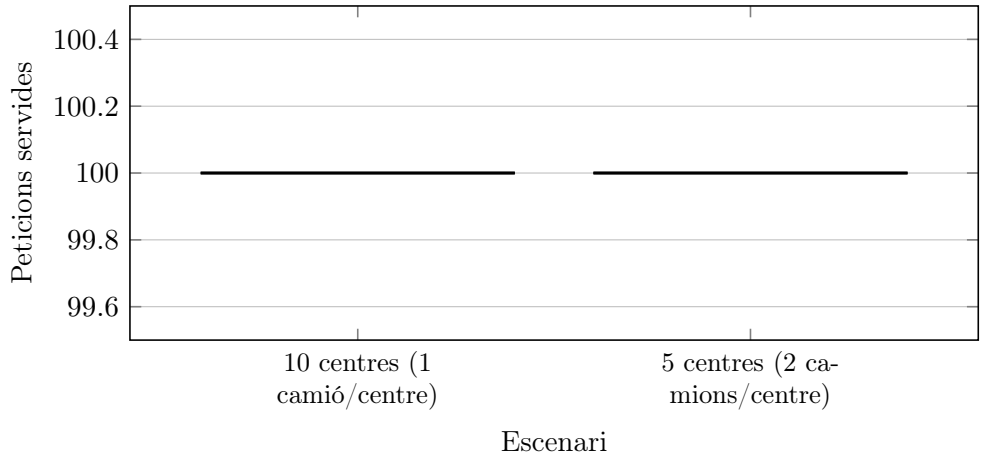

Figura 20: Comparació de peticions servides entre ambdós escenaris

7.6.4 Anàlisi

Els resultats indiquen que la distribució amb més centres (encara que amb menys camions per centre) és més eficient. La reducció de centres implica que els camions han de realitzar trajectes més llargs per cobrir tot el territori, incrementant significativament els quilòmetres recorreguts i, conseqüentment, reduint el benefici econòmic degut als majors costos operatius.
Això suggereix que, en aquest context específic, una distribució més descentralitzada dels recursos és preferible, ja que permet reduir les distàncies de desplaçament mantenint la mateixa capacitat de resposta. La proximitat dels centres als punts de servei sembla ser un factor crític en l'eficiència del sistema.

7.6.5 Conclusió

La concentració de camions en menys centres, tot i mantenir el mateix nombre total de vehicles, resulta en un increment dels costos operatius i una reducció del benefici econòmic, sense millorar la capacitat de servei. Per tant, en aquest escenari concret, la distribució amb 10 centres és més eficient que la distribució amb 5 centres.

7.7 Experiment 6: Variació del cost per quilòmetre

7.7.1 Objectiu

L'experiment té com a objectiu analitzar com l'augment del cost per quilòmetre recorregut afecta el nombre de peticions servides en un sistema de gestió de peticions, utilitzant l'algoritme Hill Climbing. Es vol observar si aquest increment de costos redueix la proporció de peticions servides i si hi ha algun efecte en funció del temps que les peticions porten pendents.

7.7.2 Configuració

S'ha utilitzat l'algorisme Hill Climbing per optimitzar la planificació de les rutes, partint d'un escenari base on el cost per quilòmetre és de 2 unitats. Aquest cost s'ha duplicat successivament fins a assolir valors de 4, 8, 16 i 32 unitats. S'ha mantingut constant la resta de paràmetres de l'entorn, com el nombre de peticions i la distribució temporal d'aquestes. S'han recollit les dades de benefici econòmic i el nombre de peticions servides.

7.7.3 Resultats

Benefici Econòmic Com es pot observar a la Figura 21, a mesura que anem augmentant el cost per quilòmetre, s'observa una clara disminució del benefici.Té molt sentit que si es vol seguir servint totes les peticions (com veurem a la següent taula) i el cost per quilòmetre puja, el benefici baixa.

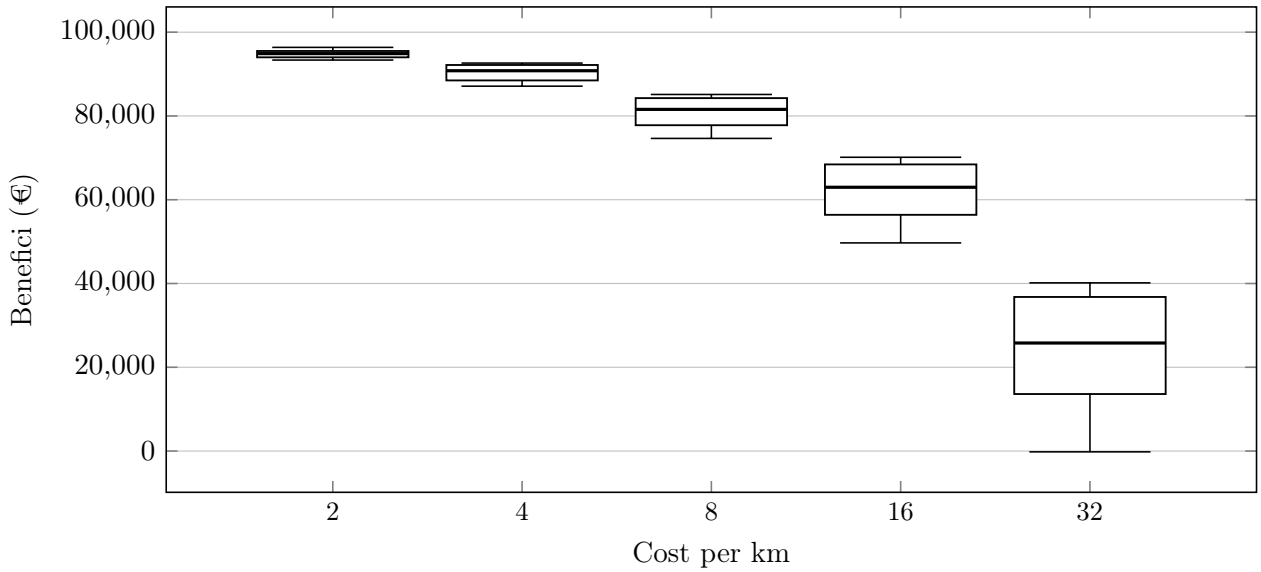

Figura 21: Evolució del benefici econòmic segons el cost per km recorregut

Peticions servides Com es pot observar a la Figura 22, el nombre de peticions servides es manté constant durant to l'experiment. Això es deu pricipalment a que la funció heurística no penalitza suficient el cost per quilòmetre o premia en excés el voler servir totes les peticions, i finalment el preu per quilòmetre no afecta al nombre de peticions servides

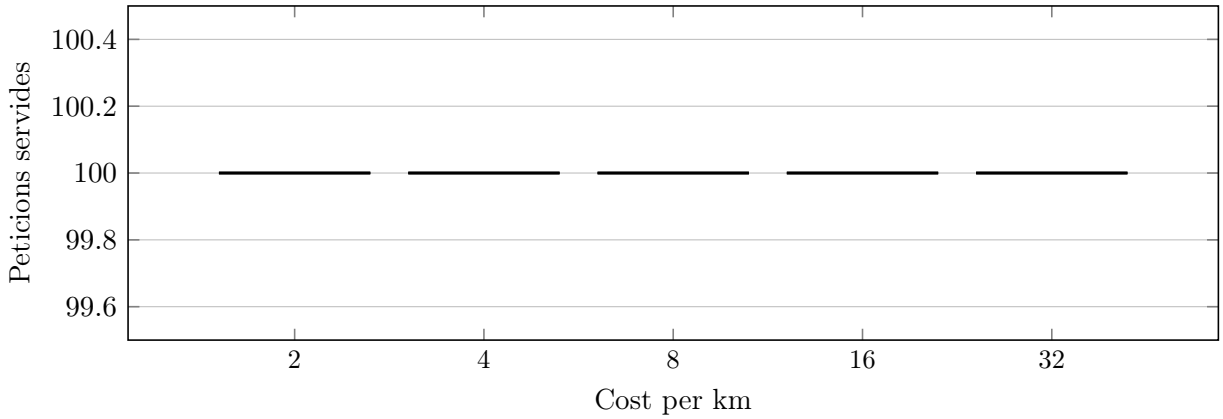

Figura 22: Nombre de peticions servides segons el cost per km (totes servides amb èxit)

7.7.4 Conclusió

Els resultats mostren que l'increment del cost per quilòmetre té un impacte negatiu clar sobre el benefici econòmic, però no afecta el nombre total de peticions servides, que es manté al 100% en tots els casos. Això suggereix que el sistema, en la seva configuració actual, prioritza la cobertura completa de la demanda per damunt de l'eficiència econòmica.
Pel que fa a la proporció de peticions servides en funció del temps que porten pendents, no s'ha observat cap variació significativa, ja que totes les peticions acaben essent ateses. Per poder avaluar aquest efecte amb més detall, seria necessari repetir l'experiment en un escenari amb major càrrega de treball o amb restriccions més severes (menys vehicles, major distància o més peticions simultànies), de manera que no totes les peticions poguessin ser servides. En aquests contextos, sí es podria analitzar si el cost per quilòmetre influeix en la priorització de peticions antigues o recents.

En resum, l'experiment confirma la sensibilitat del benefici respecte al cost operatiu, però no evidencia canvis en el comportament del sistema pel que fa al servei de peticions.

7.8 Experiment 7: Variació de les hores de treball

7.8.1 Objectiu

L'objectiu d'aquest experiment és analitzar l'impacte que té la variació de les hores de treball dels camions cisterna sobre el benefici econòmic obtingut. Concretament, es manté constant el límit de 5 viatges diaris per camió i es modifica el nombre de quilòmetres màxims que es poden recórrer en un dia, simulant així l'efecte d'augmentar o reduir la jornada laboral en una hora.

7.8.2 Configuració experimental

Per aquest experiment s'han considerat tres escenaris diferents:

- **Jornada reduïda (7 hores):** 560 km/dia
- **Jornada estàndard (8 hores):** 640 km/dia
- **Jornada ampliada (9 hores):** 720 km/dia

S'ha utilitzat l'algorisme Hill Climbing per optimitzar les solucions en cada escenari, executant 10 repeticions per cada configuració per garantir la validesa estadística dels resultats.

7.8.3 Anàlisi dels resultats

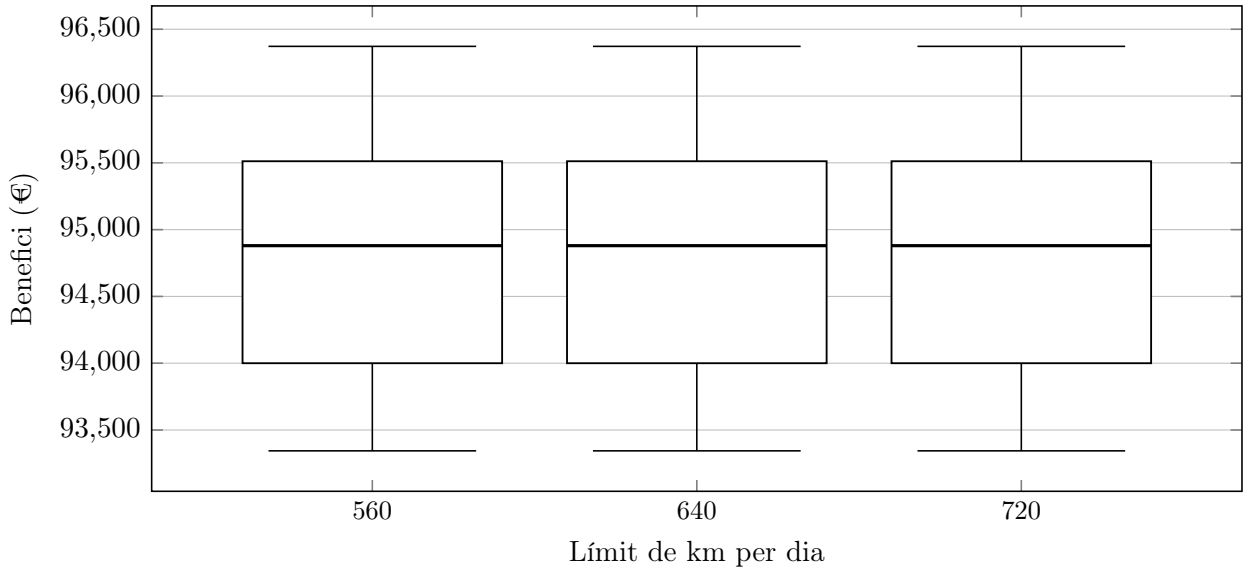

Figura 23: Benefici econòmic segons el límit de km diari

Benefici econòmic La figura anterior mostra la distribució del benefici econòmic obtingut per als tres escenaris considerats. Com es pot observar, les distribucions són pràcticament idèntiques, amb una mediana situada als 94.880€ en tots els casos. Els quartils i els bigotis del boxplot també coincideixen, indicant que la variació del límit quilomètric diari no té cap efecte apreciable sobre el benefici econòmic generat.
Aquest resultat sorprèn inicialment, ja que seria raonable esperar que una major disponibilitat de quilòmetres permetés realitzar rutes més llargues o servir peticions més llunyanes que potencialment poguessin generar més benefici. No obstant això, la invariància observada suggereix que altres factors del sistema són més restrictius que el límit de quilometratge.

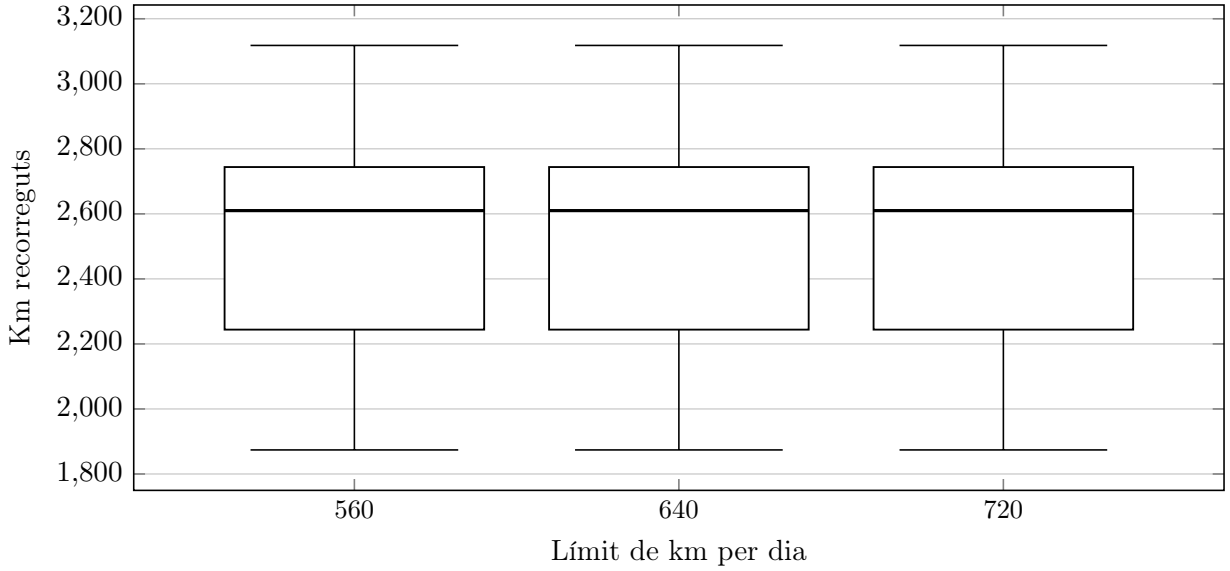

Figura 24: Distància total recorreguda segons el límit de km diari

Distància recorreguda Analitzant la distància total recorreguda pels camions, s'observa novament una distribució idèntica per als tres escenaris, amb una mediana al voltant dels 2.610 km. Aquest resultat és consistent amb el comportament del benefici econòmic i proporciona una explicació del fenomen observat: els camions no utilitzen tot el límit de quilometratge disponible, ni tan sols en l'escenari més restrictiu de 560 km/dia.
La distància mitjana recorreguda per camió i dia es situa molt per sota dels límits establerts en qualsevol dels tres escenaris. Això indica que la configuració de rutes òptimes trobada per l'algorisme Hill Climbing no requereix apropar-se al límit quilomètric, suggerint que les peticions es troben relativament a prop dels centres de distribució o que les rutes es poden organitzar de manera eficient sense necessitat de recórrer distàncies extenses.

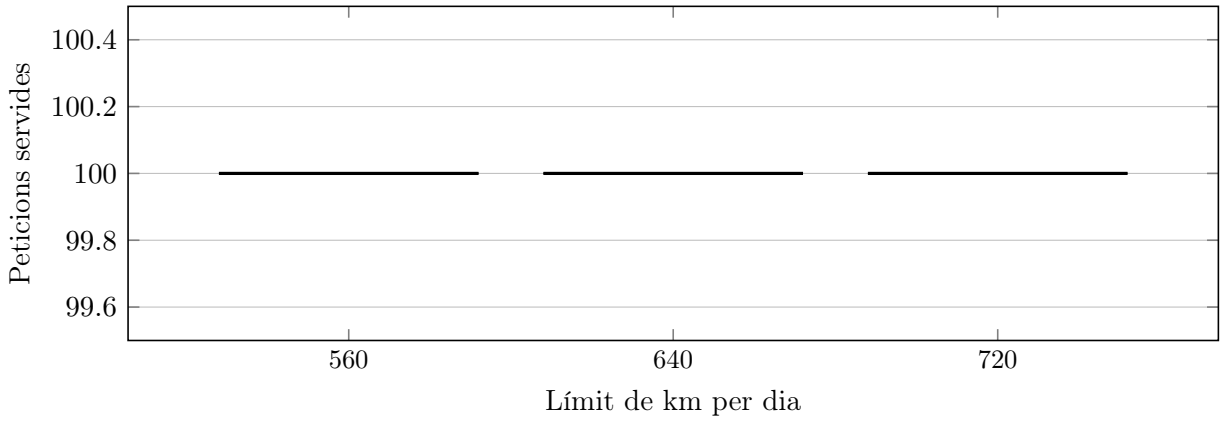

Figura 25: Nombre de peticions servides segons el límit de km diari

Peticions servides Finalment, el nombre de peticions servides és constant i igual a 100 en tots els escenaris analitzats. Això confirma que la flota de camions disponible té

capacitat suficient per atendre totes les peticions existents, independentment del límit de quilometratge establert.
Aquest resultat és clau per entendre el comportament global del sistema: cap dels límits quilomètrics imposats impedeix servir la totalitat de les peticions. Per tant, altres restriccions del problema, com ara el límit de 5 viatges diaris per camió o la capacitat dels vehicles, són les que realment determinen el rendiment del sistema.

7.8.4 Conclusions

Els resultats d'aquest experiment revelen que, en el context d'aquest problema específic, la restricció de quilometratge diari no és el factor limitant del sistema. Les possibles explicacions d'aquest comportament són:

1. La distribució geogràfica de les peticions i els centres de distribució permet organitzar rutes eficients que no requereixen recórrer distàncies properes al límit establert.

2. El límit de 5 viatges diaris per camió es converteix en la restricció dominant, impedint que els camions realitzin més viatges independentment dels quilòmetres disponibles.

3. La capacitat dels vehicles i el nombre de camions disponibles són suficients per cobrir totes les peticions amb rutes relativament curtes.

En conseqüència, augmentar o reduir la jornada laboral en una hora no genera cap millora ni deteriorament del benefici econòmic. Per millorar l'eficiència operativa del sistema, caldria revisar altres paràmetres com ara el nombre de camions disponibles, el límit de viatges diaris, o considerar conjunts de peticions més grans o geogràficament més disperses on la restricció quilomètrica pogués esdevenir realment rellevant.

8 Conclusions

8.1 Assoliment d'objectius

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 Intel·ligència Artificial. Tots els objectius plantejats a la Secció 1 han estat assolits satisfactòriament:

1. **Modelatge del problema**: Hem modelat el problema com un problema de busqueda local i definit una representació de l'espai d'estats que permet explorar totes les solucions: Secció 2 i 3
2. **Disseny d'operadors**: Hem desenvolupat un conjunt complet d'operadors que cobreixen tot l'espai de solucions: Secció 4
3. **Estratègies d'inicialització**: Hem comparat diferents aproximacions i escollit entre elles mitjançant l'experimentació: Secció 5
4. **Funció heurística**: Hem dissenyat una heurística que equilibra múltiples objectius del problema: Secció 6
5. **Experimentació rigorosa**: Hem realitzat els experiments sistemàticament i validat les nostres dessicions de disseny: Secció 7
6. **Comparació d'algoritmes**: Hem demostrat les diferències entre Hill Climbing i Simulated Annealing: Secció 7

8.2 Conclusions principals

Les principals conclusions de la pràctica mostren que les tècniques de cerca local són una eina molt efectiva per abordar problemes reals de planificació de rutes. El model de representació utilitzat, basat en els viatges de cada camió, ha resultat eficient i flexible, i el conjunt d'operadors dissenyats permet explorar adequadament l'espai de solucions. També s'ha vist que la manera d'inicialitzar la solució influeix fortament en la velocitat de convergència i en la qualitat final dels resultats.
Pel que fa a la funció heurística, la combinació de beneficis, penalitzacions i costos de desplaçament s'ha demostrat equilibrada i capaç d'adaptar-se als diferents escenaris del problema. Els experiments han confirmat que els tres components són necessaris per mantenir un bon compromís entre ingressos i eficiència.
En relació amb l'escalabilitat del problema, hem comprovat que Hill Climbing és molt més lent que Simulated Annealing, i el marge de benefici és molt just per justificar el seu ús: l'algorisme de SA aconsegueix aproximadament els mateixos beneficis i el seu cost temporal és molt més elevat. Si hem de tractar el problema amb escenaris grans, hem d'utilitzar SA per sobre de Hill Climbing degut al cost en temps tan exagerat.

A nivell de problema, s'ha observat que la restricció més limitant és el nombre de viatges per camió, ja que el nombre de punts d'abastiment i el cost per quilòmetre no han estat gaire restrictius.
En conjunt, la pràctica ha servit per entendre millor la relació entre modelatge, heurístiques i estratègies de cerca, i per comprovar que amb una bona representació, una heurística adequada i una experimentació sistemàtica és possible obtenir solucions de qualitat en temps molt raonables. A més, l'experiència ha reforçat la importància del treball en equip, la planificació constant i la documentació com a factors clau per a l'èxit del projecte.

8.3 Valoració personal

8.3.1 Dificultats trobades

Ens va costar començar la pràctica, sobretot en la representació de l'estat. Ens va costar sobretot perquè no sabíem per on començar. També ens ha costat processar i emmagatzemar els resultats dels experiments per tal de tenir gràfiques adequades i trobar la manera d'executar alguns experiments que han trigat molt temps d'execució. Utilitzar Git com a eina de control de versions i gestió de codi, així com Latex per fer la documentació, no ha estat tan fàcil com ens pensàvem.

8.3.2 Valoració del treball en equip

El treball en equip ha estat fonamental per a l'èxit d'aquesta pràctica. Cada membre ha assumit responsabilitats clares i ens hem dividit la feina creant tasques diferenciades i independents. Hem aprofitat les classes de laboratori per fer reunions setmanals per sincronitzar el treball i discutir sobre el millor disseny de la pràctica.

Ens hem ajudat amb dubtes sempre que hem pogut i l'ajuda del Carles durant les hores de laboratori a on no arribem nosaltes també ha estat crítica. Seguir la planificació proposada a l'enunciat de la pràctica ens ha anat bé per anar al dia i tenir un progrés constant.

8.4 Millores de la implementació actual

Algunes millores que hem considerat i no hem implementat són:

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 ponderacions
    - Considerar la fase de la búsqueda (inici vs final)
    - Incorporar informació històrica de la búsqueda

9 Treball d'innovació

9.1 Tema del treball

Adobe Firefly i la innovació Generative Fill a Photoshop

9.2 Breu descripció del tema

Adobe Firefly és una plataforma d'intel·ligència artificial generativa creada per Adobe que permet generar i editar contingut visual mitjançant instruccions en llenguatge natural. La seva funció de Generative Fill a Photoshop permet afegir, eliminar o modificar elements d'una imatge de manera automàtica i realista. Aquesta innovació transforma el procés creatiu, fent-lo més ràpid, accessible i potent per a tot tipus d'usuaris.

9.3 Repartiment del treball entre els membres del grup

Hem dividit el treball en tres tasques diferencades que després repartirem i posarem en comú:

- Encarregat de la cerca d'informació i redacció dels apartats relacionats amb la introducció, la descripció del producte o servei i el context de la innovació dins d'Adobe.
- Encarregat d'investigar i desenvolupar els apartats tècnics: tècniques d'IA utilitzades (models de difusió), adaptació a l'ecosistema Adobe i la naturalesa de la innovació (tipus i comparació amb altres solucions).
- Responsable dels apartats d'impacte a l'empresa, impacte en els usuaris i la societat, conclusions i de la compilació i revisió general del document.

Fins ara només hem triat el tema i dividit les tasques, encara no hem començat amb l'informe ni la cerca d'informació. Encara no tenim bibliografia ni ens hem trobat amb gaires dificultats