

**ALGORITMES BÀSICS PER LA LINTEL·LIGÈNCIA ARTIFICIAL**  
GRAU EN INTEL·LIGÈNCIA ARTIFICIAL

# Pràctica de Búsqueda Local – AZAMON

Nadia Fernandez, Jana Roman,  
Júlia Pedrol i Paula Justo

Universitat Politècnica de Catalunya  
Dilluns 11 de Novembre 2024.

## ÍNDEX

<b>1. INTRODUCCIÓ</b>	<b>3</b>
<b>2. OBJECTIUS PRINCIPALS</b>	<b>4</b>
<b>3. PREGUNTES ABANS DE COMENÇAR</b>	<b>5</b>
<b>3. DESCRIPCIÓ DEL PROBLEMA I EL SEU FUNCIONAMENT</b>	<b>6</b>
3.1. REPRESENTACIÓ DE L'ESTAT (SOLUCIÓ)	6
3.2. ESTATS (SOLUCIONS) INICIALS PLANTEJATS	7
3.3. OPERADORS PLANTEJATS	9
3.4. FUNCIONS HEURÍSTIQUES	10
3.4. FUNCIONS ADDICIONALS	10
<b>4. EXPERIMENTACIÓ</b>	<b>12</b>
4.1. EXPERIMENT 1	12
4.2. EXPERIMENT 2	14
4.3. EXPERIMENT 3	16
4.4. EXPERIMENT 4	17
4.5. EXPERIMENT 5	19
4.6. EXPERIMENT 6	20
4.7. EXPERIMENT 7	21
4.8. EXPERIMENT 8	22
4.9. EXPERIMENT ESPECIAL	23
<b>5. COMPARACIÓ DELS RESULTATS OBTINGUTS ENTRE HILL CLIMBING I SIMULATED ANNEALING</b>	<b>25</b>
5.1. CONCLUSIONS DELS ALGORISMES	26
<b>6. CONCLUSIONS DELS TREBALL</b>	<b>27</b>

## 1. INTRODUCCIÓ

Durant el transcurs d'aquesta pràctica hem treballat amb la resolució d'un problema d'optimització utilitzant algorismes de cerca local, inicialment el *Hill Climbing* i posteriorment, adaptant-ho al *Simulated Annealing*.

L'objectiu d'aquest treball ha estat trobar solucions eficients per a la distribució de paquets d'una empresa de transport fictícia, considerant factors com el cost de transport, el temps d'emmagatzematge i la satisfacció/felicitat del client.

El desenvolupament de la pràctica l'hem realitzat en grup, amb la participació de quatre alumnes, combinant el treball col·laboratiu de les sessions de laboratori amb les hores de treball autònom. A nivell pràctic, hem implementat diferents estratègies per generar solucions ideals, hem comparat els resultats de diversos experiments i hem analitzant els factors que influeixen en la qualitat de les solucions obtingudes amb cadascun dels algorismes.

Aquest informe presenta el procés seguit, els experiments realitzats, els resultats obtinguts i les conclusions extretes després de comparar les diferents tècniques aplicades.

## 2. OBJECTIUS PRINCIPALS

Des d'un inici, quan se'ns va plantejar la pràctica, ens vam formular les següents dues preguntes:

- Com podem representar i gestionar efectivament els diferents elements del problema logístic (estat inicial, representació de l'estat, operadors, etc.) dins del programa.
- Com podem implementar els algorismes de cerca local, com el *Hill Climbing* i el *Simulated Annealing*, per resoldre'l de manera òptima.

Aquestes dues qüestions han estat el fil conductor del nostre treball. Durant tot el procés, hem cercat informació i solucions per resoldre aquests reptes, i ens hem marcat els següents objectius per guiar-nos en la realització d'aquest projecte:

1. **Entendre el problema i la seva representació:** hem intentat comprendre a fons els elements que conformen el problema d'assignació de paquets a ofertes de transport, i com aquests es poden representar i manipular dins del nostre programa, tenint en compte les restriccions de capacitat, temps i costos.
2. **Implementar les classe i algorismes demanats:** guiant-nos per les indicacions establertes, hem treballat en la implementació de les classes necessàries per simular els escenaris, així com els algorisme de cerca local (*Hill Climbing* i *Simulated Annealing*), amb l'objectiu de comparar el seu rendiment en diferents escenaris.
3. **Crear i realitzar experiments:** un cop implementat el programa, hem dissenyat una sèrie de proves i experiments per analitzar el comportament dels algorismes en situacions variades, avaluant els costos, la qualitat de les solucions i el temps d'execució.
4. **Millorar les nostres habilitats en Python:** a través de la implementació del codi, hem buscat consolidar i millorar el nostre domini del llenguatge Python, especialment en l'àmbit dels algorismes d'optimització i les estructures de dades eficients.
5. **Resoldre problemes i adversitats en el procés de programació:** hem après a afrontar els diversos problemes tècnics que han sorgit durant la implantació i l'execució dels experiments, treballant en equip per superar les dificultats i trobar solucions eficients.

### 3. PREGUNTES ABANS DE COMENÇAR

#### **Quins elements intervenen en el problema?**

- Els elements del problema són els paquets, les ofertes diàries de transport i els costos associats tant al transport dels paquets com l'emmagatzematge d'aquests.

#### **Quin és l'espai de cerca?**

- L'espai de cerca/búsqueda inclou totes les possibles assignacions/organitzacions de paquets en ofertes de transport, que han de ser explorades pels diferents algorismes de cerca local (Hill Climbing o Simulated Annealing).

#### **Quin és el tamany de l'espai de cerca?**

- El tamany de l'espai de cerca es pot considerar molt gran, ja que cada paquet pot ser assignat a diverses ofertes, incrementant exponencialment el nombre de possibles solucions.

#### **Què és un estat (solució) inicial?**

- Un estat/solució inicial és una assignació concreta dels paquets a les ofertes presentades aquell dia. L'enunciat requereix dissenyar almenys dues estratègies per generar aquesta solució inicial.

#### **Quines condicions compleix un estat final?**

- L'estat/solució final és una assignació on els costos de transport i emmagatzematge són mínims, o propers a l'òptim, segons els criteris de l'algorisme.

#### **Quins operadors permeten modificar els estats?**

- Els operadors se'ns presenten com les possibles modificacions a les assignacions dels paquets respecte les ofertes plantejades, com per exemple, aquests ens permeten canviar un paquet d'una oferta a una altra o modificar les combinacions de paquets entre les ofertes.

#### **Quin és el factor de ramificació dels operadors de canvi d'estat?**

- El factor de ramificació dependrà del nombre de paquets i ofertes, cada operador podria generar diverses noves assignacions per explorar/experimentar.

### 3. DESCRIPCIÓ DEL PROBLEMA I EL SEU FUNCIONAMENT

El problema consisteix a resoldre un problema d'optimització log''ítica en què s'han d'assignar una sèrie de paquets a les diferents ofertes de transport que es reben diàriament. Els elements clau que intervenen són els paquets, les ofertes de les companyies de transport i els costos associats, tant pel transport com per l'emmagatzematge dels paquets abans que sigui recollits.

L'espai de cerca inclou totes les possibles assignacions dels paquets a les ofertes, generant així un gran nombre de combinacions possibles que es poden explorar. Degut a la complexitat del problema, aquest espai de cerca és molt més ampli, ja que cada paquet pot ser assignat a diferents ofertes, incrementant exponencialment el nombre de solucions.

El problema comença amb una solució inicial, plantejada per nosaltres, què és una assignació concreta dels paquets a les ofertes proporcionades. A partir d'aquí, es poden aplicar diferents operadors que modifiquen aquesta assignació inicial, canviant els paquets d'una oferta a una altra per tal d'explorar noves solucions.

L'objectiu final és trobar una assignació on els costos de transport i emmagatzematge siguin mínims o propers a l'òptim, segons els criteris establerts pels algorismes de cerca local, com el *Hill Climbing* o el *Simulated Annealing*. Aquests algorismes exploren diferents estat (solucions) per tant de millorar successivament la solució fins a trobar una de satisfactòria.

#### 3.1. REPRESENTACIÓ DE L'ESTAT (SOLUCIÓ)

La representació que hem considerat més adequada per modelar l'estat del problema és una **llista de tuples**, on cada tupla correspon a una oferta de transport proporcionada per les companyies i dins de cada tupla, fem referència als paquets que han estat assignats a aquesta oferta concreta. Així, l'estat complet es pot veure com una seqüència de tuples, cadascuna representant una oferta i els paquets associats a aquesta.

La notació és la següent:

[ {paquet1, paquet 3, ...}, ... {paquet4, paquet8, ...}]

En aquesta representació cada tupla fa referència a una de les ofertes proporcionades i conté un allista de paquets assignats a aquesta. La llista de tuples permet mantenir l'estructura clara, on cada element està vinculat a la seva oferta corresponent, cosa que facilita el seguiment de les assignacions de paquets.

El raonament pel qual considerem que aquesta és una bona representació és perquè pensem que aquesta representació proporciona una manera senzilla i intuïtiva de visualitzar les assignacions de paquets, mantenint tota la informació rellevant en un sol lloc i amb un cost mínim. A més a més, la llista de tuples és una estructura que es pot modificar fàcilment, permet afegir o eliminar paquets de les ofertes sense complicacions. Aquesta flexibilitat és fonamental ja que ens permet modificar l'estat de manera ràpida i explorar noves solucions respecte el problema.

D'altra banda, l'ús d'aquesta garanteix una representació que és tant eficient en termes d'espai com en temps d'execució, perquè les operacions per accedir, modificar o iterar sobre els paquets i les ofertes són ràpides, cosa que resulta crucial en aquest context on duiem una exploració de múltiples estats dins un espai de cerca molt gran.

Per últim, considerem que aquesta estructura ens permet gestionar un nombre variable de paquets i ofertes i aplicar els operadors sobre aquesta per dur a terme un reassignament de paquets entre ofertes.

En resum, aquesta llista de tuples ens proporciona una representació clara, flexible i eficient que facilita tant la implementació com l'exploració de solucions dins del context del problema d'assignació de paquets a ofertes de transport.

### 3.2. ESTATS (SOLUCIONS) INICIALS PLANTEJATS

En un inici, durant les primeres hores d'execució, ens vam plantejar diversos estats inicials:

- Ordenar paquets i ofertes per dies de més a menys i per pes també de més a menys. Assignar un paquet a una oferta i si no cap en aquest, al següent (comencem a mirar perquè arribin a temps).
- Ordenar paquets i ofertes per dies de menys a més. Assignar paquets a una oferta i si no hi cap, a la següent.
- Ordenar paquets i ofertes per dies de menys a més i per pes de més a menys. En aquest, el que fem és assignar un paquet a una oferta i si no cap a la següent (mirem sempre totes les ofertes).

La diferència entre els dos primers és que en el segon estat presentat, tota l'estona hem de mirar si hi caben els paquets; en canvi, en el primer només hem de mirar des dels que arriben bé del dia, és a dir, si ja ha passat el 5 passarem a mirar el 4 i no tornarem a mirar el 5.

Tot i això, aquests estats plantejats no ens van servir ja que tots ells utilitzen d'alguna forma o altra un mètode d'ordenació, per tant, treien feina al algorismes i això no ens permet comprovar la seva eficiència absoluta, ja que en l'estat inicial, ja els tenen ordenats d'una forma concreta, per tant, vam haver de pensar nous estats inicials els qual no tenen cap ordenació prèvia a la del algorisme.

Després de plantejar-nos diverses opcions però totes relacionades amb una ordenació prèvia, vam acabar obtenint un estat inicial que complia les característiques demanades:

- L'estat inicial és una distribució inicial de paquets assignats a diferents ofertes d'enviament en funció de les seves restriccions de pes màxim i els dies d'entrega requerits. En aquest, exposem que cada oferta té un límit de pes que pot suportar i un màxim de dies per entregar els paquets, per tant, aquest estat té com a objectiu distribuir els paquets comprovant en cada assignació que es comprovi que cada paquet sigui assignada a una oferta que compleixi el límit de temps d'entrega.

Per programar aquest, ho vam fer dins la funció `generate_action`, amb l'objectiu d'assignar els paquets a les ofertes de la manera més ajustada possible segons les seves restriccions de pes i prioritat. La funció que hem fet, comença inicialitzant dues estructures de dades: una llista de conjunt (`v_ofertes`) per representar els paquets assignats a cada oferta i una llista (`peso_ofertas`) que conté el pes màxim disponible per cada oferta. Això ens permet tenir un registre clar dels paquets que es poden assignar a cada oferta i del pes màxim que encara podem afegir-hi.

Per gestionar els dies d'entrega segons la prioritat de cada paquet, vam definir una funció interna `dias_prioridad` que converteix la prioritat d'un paquet (expressada en 0, 1 o 2) en el nombre de dies que aquest pot esperar per ser lliurat. Així, quan avaluem si un paquet pot assignar-se a una oferta, comparem aquest nombre de dies amb els dies d'entrega màxims que l'oferta pot oferir.

El bucle principal revisa cada paquet i intenta assignar-lo a l'última oferta disponible, començant des de la darrera de la llista d'ofertes (amb l'índex més gran) cap a la primera. Aquesta prioritat d'assignació ens ajuda a esgotar primer les ofertes amb més restriccions. Si un paquet compleix les restriccions de pes i dies d'entrega d'una oferta, s'afegeix a aquesta, actualitzant la capacitat de pes restant per a aquella oferta. Si no compleix els requisits, el paquet es revisa per a l'oferta següent fins a trobar una opció vàlida o fins a arribar al final de la llista d'ofertes. Aquesta estructura ens permet controlar que només els paquets que realment compleixen les condicions siguin assignats i ajuda a reduir el nombre de càlculs innecessaris.

En acabar, la funció retorna un objecte de la classe `StateAzamon`, que inclou la llista d'ofertes, la llista de paquets, i la representació final dels paquets assignats a cada oferta a `v_ofertes`, creant així l'estat inicial que es pot utilitzar com a punt de partida per optimitzacions o per a un algoritme de cerca.

### 3.3. OPERADORS PLANTEJATS

Dins la funció `generate_actions` també vam definir dos operadors per modificar la distribució de paquets entre les ofertes en funció de les seves restriccions, aquests operadors són:

- Moure un paquet d'una oferta a una altra
- Intercanviar dos paquets entre ofertes diferents
- Intercanviar tots paquets entre les ofertes

Aquestes operacions ens ajuden a crear noves configuracions d'estats, explorant diferents assignacions de paquets per millorar l'eficiència de la distribució.

Pel que va a l'operador **`moure_paquet`**, aquest revisa cada paquet en cada oferta per veure si pot ser traslladat a una altra oferta on també compleixi les restriccions de pes i dies d'entrega. Primer, la funció calcula el pes lliure que queda a cada oferta (el pes màxim disponible) després d'haver restat el pes dels paquets assignats. Aquesta informació la guardem a la lista `peso_libre_ofertas`.

Per cada paquets, el codi revisa totes les ofertes per veure si el paquet pot ser mogut a alguna d'aquestes, per fer això, comprova que no sigui la mateixa oferta, la restricció de dies establerta i la restricció de pes.

Per últim, quan es compleixen aquestes condicions, es genera un objecte `MourePaquet` amb els índex del paquet i de les ofertes d'origen i destinació per indicar l'acció de moure aquest a una altra oferta. El factor de ramificació en aquest cas és  $O(P*O)$ , ja que canviem un paquet ( $P$ ) d'una oferta ( $O$ ) a una altra.

En referència a l'operador **`intercanviar_paquet`**, aquest intenta fer un intercanvi entre dos paquets de diferents ofertes per veure si això millora la distribució. Per cada parella de paquets, en el codi, comprovem que les ofertes siguin diferents, i que es compleixen les restriccions de dies i pes. Quan totes aquestes condicions es compleixen, es genera un objecte `IntercanviarPaquet` que conté els índex dels dos paquets a intercanviar. El factor de ramificació d'aquest operador és  $O(P^2)$ , on  $P$  és el paquet canviem.

Per últim, vam implementar un operador anomenat **`intercanviar_oferta`**, aquest operador busca intercanviar tots els paquets entre dues ofertes, sempre que es compleixin les restriccions de pes màxim i temps d'entrega. Primer, revisa cada parella d'ofertes per assegurar-se que cada oferta tingui la capacitat de pes necessària per suportar els paquets de l'altra. A més, comprova que els terminis d'entrega de cada oferta permetin complir amb els dies de prioritat dels paquets de l'altra. Si totes les condicions es compleixen, es genera una acció d'intercanvi, creant noves opcions de distribució de paquets per optimitzar l'eficiència del sistema. El factor de ramificació en aquest cas és  $O(O^2)$ , ja que canviem tot el contingut d'una oferta ( $O$ ) per tot el contingut d'una altra.

### 3.4. FUNCIONS HEURÍSTIQUES

Pel que fa a les funcions heurístiques plantejades, n'hem programat 4.

- Una funció heurística que hem fet és utilitzar únicament tinguent en compte la felicitat i intenta buscar solucions que maximitzin únicament el seu valor, sense tenir en compte els valors obtinguts en el cost d'emmagatzematge.
- Seguidament, vam programar la mateixa però únicament tenint en compte els costos d'emmagatzematge amb l'objectiu de maximitzar aquest valor i troba la solució més òptima.
- Per últim, tenint en compte els valors obtinguts en cada una de les heurístiques anteriors, vam buscar un valor de proporcionalitat d'aquests dos. Aleshores el que volem fer és minimitzar els costos i maximitzar la felicitat, però intentant mantenir una relació entre els dos. Volem donar més importància als costos que a la felicitat, però que no hi hagi un gran equilibri. És a dir, hem buscat que sigui 60% els costos i 40% la felicitat, això ho hem aconseguit mirant els valors que ens han sortit de les dues heurístiques anteriors.

### 3.4. FUNCIONS ADDICIONALS

Per poder completar el codi de manera eficient i dur a terme càlculs de forma més simplificada vam programa 5 funcions addicionals.

Les funció les vam pensar per calcular diferents aspectes relacionats amb la prioritat, els costos d'emmagatzematge i transport, així con la felicitat dels paquets i resultat total de felicitat.

- **Funció dias\_prioridad(self, p\_i):** aquesta funció determina els dies associats a la prioritat d'una paquet, Si la prioritat del paquet en la posició p\_i de la llista de paquets és 0, retorna 1, indicant que aquest paquet ha de ser enviat en un dia. si la prioritat és 1, es retorna 3, i en cas que sigui una prioritat superior, es retorna 5 dies. Aquesta ens permet crear un sistema en el qual els paquets amb menor prioritat tringuen més a ser processats.
- **Funció costes\_acmacenamiento(self, o\_i):** aquesta calcula el cost d'emmagatzematge d'una oferta específica o\_i. Si el nombre de dies d'emmagatzematge de l'oferta és 4 o 3, el cost és de 0,25 unitats multiplicat pel pes de l'oferta. Si són 5 dies, el cost es redueix a la meitat de 0,25 unitats multiplicat pel pes de l'oferta. Per a altres valors de dies, no hi ha cap cost d'emmagatzematge. El cost total calculat es retorna com a resultat.
- **Funció costes\_trasportes(self, o\_i):** calcula el cost de transport d'una oferta o\_i basant-se en el seu pes i el preu per unitat de pes. Multiplica el pes de l'oferta pel preu unitari del transport associat a aquesta oferta, retornant així el cost total de transport.
- **Funció felicidad(self, p\_i):** aquesta funció calcula la felicitat o satisfacció associada a un paquet segons la prioritat i els dies de lliurament. Primer, troba l'oferta assignada al paquet p\_i i consulta els dies de lliurament d'aquesta oferta. Si el paquet té una prioritat de 2 i l'oferta es lliura en menys de 4 dies, la felicitat és la diferència entre 4 i el nombre de dies de lliurament.

Si la prioritat és 1 i es lliura en menys de 2 dies, la felicitat és la diferència entre 2 i els dies de lliurament. Si cap d'aquestes condicions es compleix, la felicitat és zero.

- **Funció `felicitad_total(self)`:** aquesta calcula la felicitat total de tots els paquets en una llista anomenada `paquetes`. Recorre cada paquet en la llista, cridant a la funció `felicitad` per calcular la seva felicitat individual, i acumula aquest valor en una variable `fel`. Finalment, retorna el valor total de felicitat acumulada.

## 4. EXPERIMENTACIÓ

### EXPERIMENTACIÓ

Aquí es troba adjunt l'Excel on estan definits tots els resultats de les experimentacions realitzades.

#### 4.1. EXPERIMENT 1

En el primer experiment, vam analitzar com les diferents combinacions dels operadors `moure_paquet`, `intercanviar_paquet`, i `intercanviar_oferta` afectaven els resultats de l'heurístic i el temps d'execució en la distribució de paquets. Per a aquest anàlisi, vam utilitzar l'algorisme *Hill Climbing*, enfocant-nos en optimitzar tant el valor heurístic com el temps d'execució per trobar la combinació d'operadors que millorava els resultats més ràpidament.

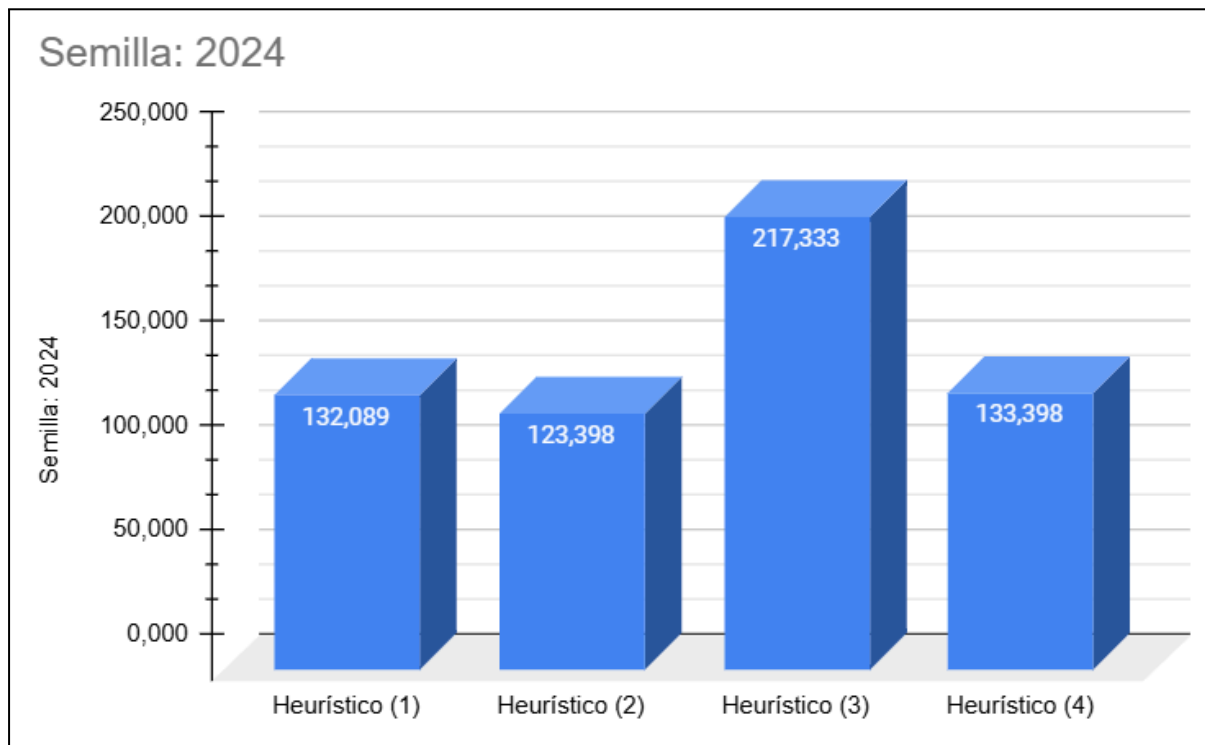
Per a això, vam definir quatre escenaris de combinació d'operadors, provant cadascun amb 10 llavors diferents (2024, 1, 1002, 1424, 311423, 86, 10, 1234, 876 i 564), i a partir dels resultats obtinguts vam calcular les mitjanes per extreure conclusions més destacables. Els escenaris van ser els següents:

1. **Escenari 1:** Utilitza els operadors `moure_paquet` i `intercanviar_paquet`, permetent `moure` un paquet a una altra oferta o `intercanviar` paquets entre ofertes diferents, ajustant així la distribució.
2. **Escenari 2:** Inclou els operadors `moure_paquet` i `intercanviar_oferta`, és a dir, mou paquets entre ofertes o intercanvia ofertes completes, per observar si una redistribució global dels paquets afecta l'eficiència.
3. **Escenari 3:** Combina `intercanviar_paquet` i `intercanviar_oferta`, enfocat en reorganitzar ofertes completes o paquets específics entre ofertes per aconseguir una millor assignació global.
4. **Escenari 4:** Avalua l'ús dels tres operadors junts (`moure_paquet`, `intercanviar_paquet` i `intercanviar_oferta`), per determinar si una estratègia més completa millora significativament els resultats.

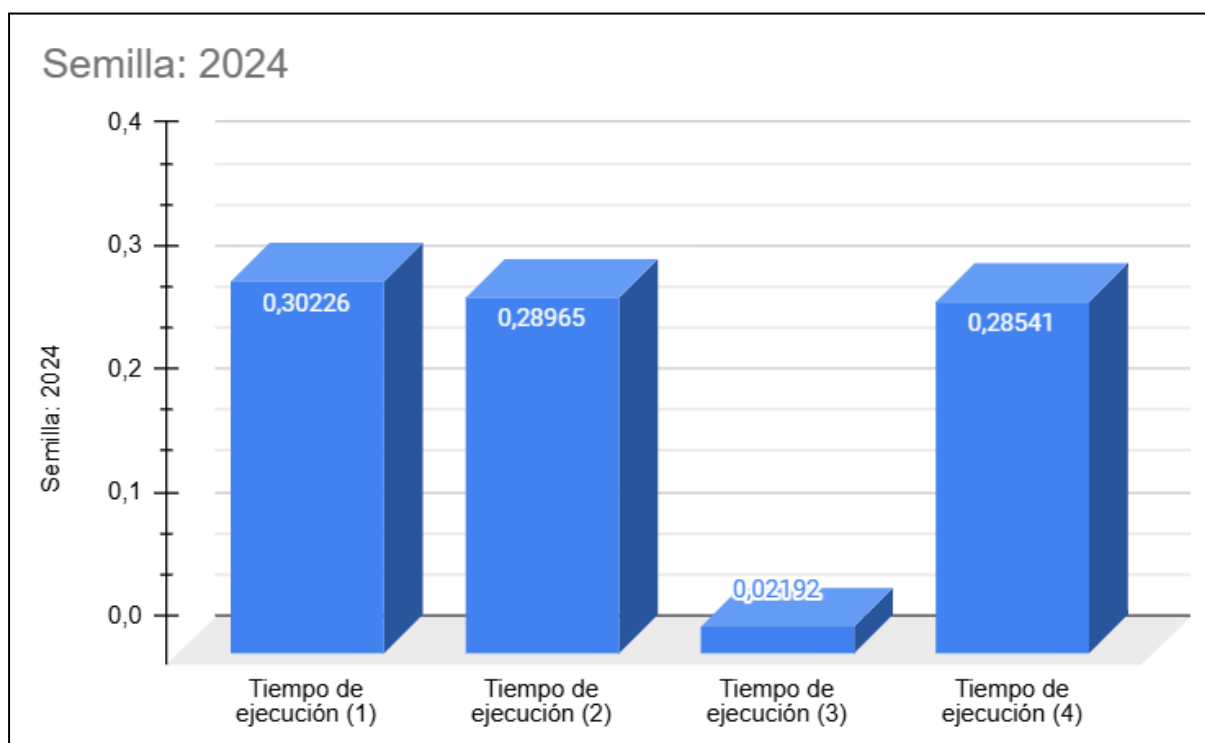
Per a cada escenari, vam calcular el temps d'execució mitjà, obtenint-lo com la mitjana dels temps de totes les iteracions realitzades durant cada execució de l'experiment.

Els resultats van indicar que la combinació d'operadors `moure_paquet` i `intercanviar_oferta` (Escenari 2) va produir el millor valor heurístic amb el temps d'execució més curt. Aquesta combinació va resultar ser la més eficient per distribuir els paquets de manera òptima i aconseguir una assignació de recursos més efectiva. En comparació, les altres combinacions van ser menys eficients o van requerir més temps d'execució, per la qual cosa vam decidir utilitzar aquesta combinació d'operadors per a la resta de l'experimentació.

A través dels gràfics següents, podem observar la relació més coherent entre el valor de l'heurístic i el temps d'execució en cada escenari, cosa que reforça la conclusió sobre l'eficiència de l'Escenari 2.



*figura1*



*figura2*

Pel que fas als resultats referents a les iteracions amb cadascuna de les llavors, aquest es troben indicats en el full de càlcul d'experimentació, on es pot consultar el detall de cada execució per a cada combinació d'operadors i cada valor de la llavor.

Pel que fa als resultats obtinguts, podem concloure que la decisió presa era una de les opcions més raonables, ja que inicialment havíem previst que les combinacions plantejades en els escenaris 1 i 2 serien les més prometedores en relació amb l'objectiu establert. En canvi, esperàvem que l'escenari 3 oferís resultats més elevats, degut a que els dos operadors en aquest cas semblaven dur a terme tasques molt similars.

Finalment, els resultats obtinguts han confirmat les nostres previsions: l'escenari 2 ha demostrat ser el que ha obtingut els millors resultats, tal com esperàvem des del principi.

## 4.2. EXPERIMENT 2

L'Experiment 2 es va centrar en determinar quina estratègia de generació de la solució inicial proporcionava millors resultats utilitzant el mètode Hill Climbing.

En aquest experiment es van definir dues formes diferents d'inicialitzar les solucions: la primera consistia en una generació totalment aleatòria sense seguir cap patró preestablert mentre que la segona seguia un ordre determinat amb la intenció de crear una base més coherent. Una vegada generades aquestes solucions inicials es va aplicar l'algorisme d'optimització Hill Climbing per veure quina de les dues estratègies resultava més eficient.

Els resultats ens van mostrar que la solució inicial aleatòria va obtenir un valor heurístic mitjà de 258,723 amb un temps d'execució de 0,0001 segons mentre que la solució inicial ordenada va tenir un valor mitjà lleugerament superior de 272,58 i un temps d'execució de 0,0002 segons. Un cop aplicat l'algorisme Hill Climbing els valors es van reduir considerablement en tots dos casos però la solució inicial aleatòria va millorar fins a un valor de 146,961 mentre que la solució ordenada es va quedar en 148,58.

Tot i que inicialment s'esperava que la solució ordenada tingués un rendiment millor perquè teòricament hauria de facilitar la cerca d'una solució òptima els resultats van demostrar el contrari. La generació aleatòria va permetre aconseguir una optimització més eficaç. Això ens sembla indicar que un punt de partida aleatori permet ampliar l'espai de cerca i permet una exploració més àmplia abans de convergir cap a un òptim, ja que pot ser que si determinem una solució inicial, ja no permetem a l'algorisme explorar tot l'espai de cerca i li estem limitant l'espai. És probable que una solució inicial massa estructurada limiti les possibilitats de sortir d'òptims locals cosa que no passa quan s'inicia des d'una posició més aleatòria i flexible. Per tant els resultats ens indiquen que és més recomanable

utilitzar una estratègia de generació aleatòria en futurs experiments ja que ha demostrat ser més efectiva a l'hora d'obtenir resultats òptims amb el mètode Hill Climbing i això ens portarà a explorar més opcions i millorar l'eficàcia global del procés d'optimització.

EXPERIMENT 2	ESTATS INICIALS			
	ALEATORI		ORDENAT	
	HEURÍSTIC	TEMPS MITJÀ D'EXECUCIÓ	HEURÍSTIC	TEMPS MITJÀ D'EXECUCIÓ
SOLUCIÓ INICIAL	258,723	0,00010116933	272,58	0,000211950266
SOLUCIÓ HC	146,961	-	148,58	-

figura3

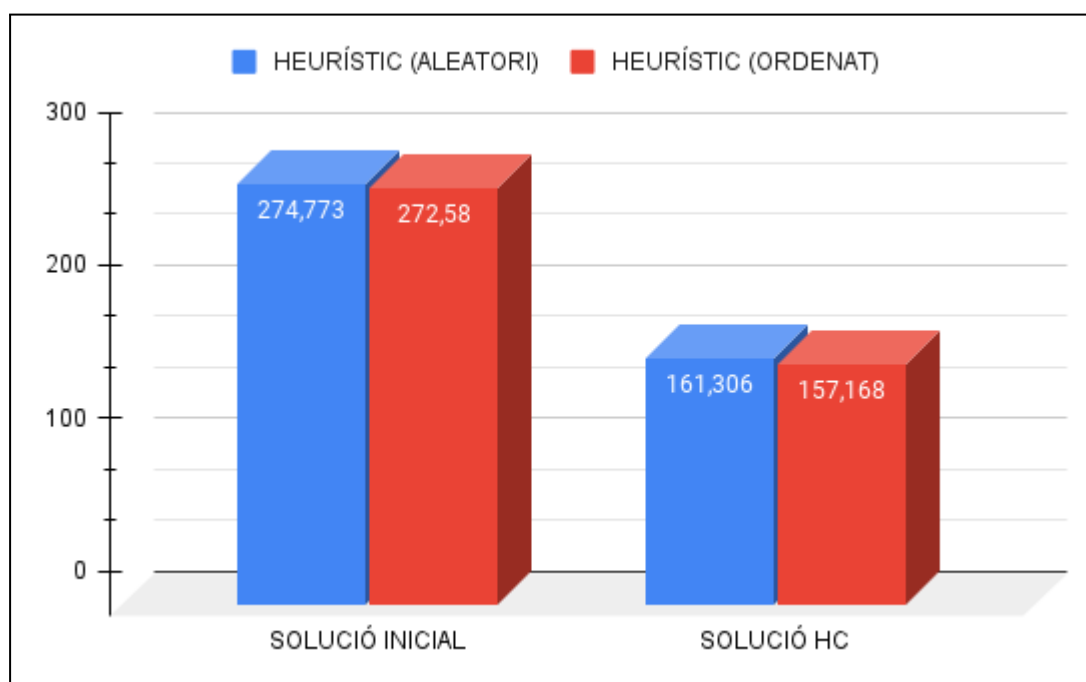


figura4

A partir d'aquest gràfic, podem extreure la conclusió que l'algorisme *Hill Climbing* (HC) millora significativament el valor heurístic en comparació amb la solució inicial, tant en l'escenari aleatori com en l'ordenat. Els valors heurístics disminueixen de 274,773 a 161,306 en l'escenari aleatori, i de 272,58 a 157,168 en l'escenari ordenat, indicant una distribució de recursos més eficient després de l'optimització.

A més, cal destacar que l'escenari ordenat permet explorar tot l'espai de solucions de manera més exhaustiva. Aquesta capacitat d'exploració contribueix a obtenir un valor heurístic lleugerament millor en comparació amb l'escenari aleatori.

### 4.3. EXPERIMENT 3

L'objectiu d'aquest experiment es determinar quins paràmetres del Simulated Annealing proporcionen millors resultats. Els paràmetres a ajustar són els factors de refredament (K i LAM) i el límit de passos màxims (LIMIT).

Inicialment, vam considerar la possibilitat de determinar els valors dels paràmetres mitjançant la probabilitat d'acceptació de solucions pitjors. No obstant, després d'avaluar aquesta opció, vam decidir no seguir-la, ja que hi havia el risc que, si la probabilitat no arribava a ser zero en el moment oportú, l'algoritme pogués acabar acceptant estats pitjors. A més, a l'agafar l'operador de manera aleatòria, no decreixia uniformement, ja que els valors eren aleatoris. Finalment, hem optat per determinar aquests valors provant valors extrems, de manera que poguéssim observar com afectaven tant l'heurístic com el temps d'execució.

Els resultats obtinguts amb les diferents combinacions de paràmetres (es podem veure en la taula de la *figura5*), on es comparen els valors de la funció heurística i el temps mitjà d'execució. Les combinacions més òptimes són:

- K = 0,01, LAM = 0.0005, LIMIT = 2000 amb un valor de l'heurístic de 161.558 i un temps d'execució mitjà de 0.7907 segons.
- K = 1, LAM = 0.0005, LIMIT = 5000 amb un valor de l'heurístic de 154.242 i un temps d'execució mitjà de 1.9926 segons.

Tot i que la segona combinació té un temps d'execució més gran, el seu valor heurístic és significativament millor. Per tant, podem dir que, a canvi d'un increment raonable en el temps d'execució (1.2019 segons més), tenim un clar avantatge en termes de la qualitat de la solució amb LIMIT = 5000.

SELECCIÓ DE PARÀMETRES				
K	LAM	LIMIT	HEURÍSTIC	TEMPS MITJÀ D'EXECUCIÓ
0,01	0,0005	20000	172,1439	8,0584
<b>0,01</b>	<b>0,0005</b>	<b>2000</b>	<b>161,558</b>	<b>0,7907</b>
0,01	0,0005	5000	173,197	1,9439
0,1	0,0005	2000	176,013	0,7875
0,1	0,0005	5000	162,896	2,1682
1	0,0005	2000	164,052	0,8345
1	0,005	2000	163,19	0,7734
<b>1</b>	<b>0,0005</b>	<b>5000</b>	<b>154,242</b>	<b>1,9926</b>
1	0,0005	25000	157,5429	9,3854

*figura5*

Cal mencionar que també es va provar utilitzant un límit d'iteracions molt més gran, però aquest ens donava un temps d'execució molt més elevat i, a més, un valor d'heurístic pitjor, de manera que vam considerar que el més oportú era reduir aquest per tal de tenir una millor compensació entre el valor del heurístic i el temps d'execució.

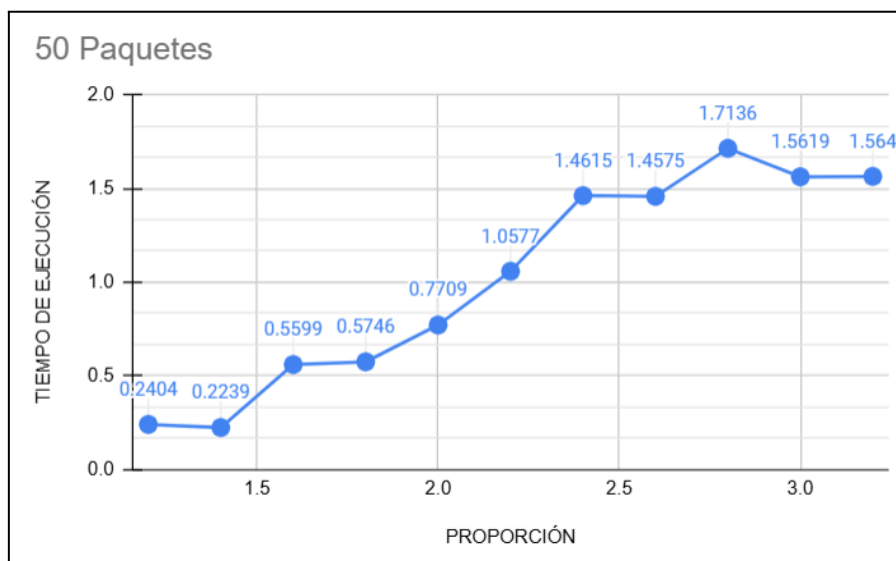
Per aquest experiment s'esperava que un augment en el límit podria oferir millors resultats per la funció heurística, a costa d'un increment en el temps d'execució (ja que ha de realitzar més iteracions). Els nostres resultats confirmen el que esperàvem, per un temps d'execució una mica més gran millorem significativament la qualitat de la nostra solució (amb un valor de l'heurístic menor).

Cal mencionar que, en cas que el temps d'execució sigui crític i es pugui acceptar una lleugera pèrdua en la qualitat de la solució, la combinació de paràmetres mencionada que utilitza  $LIMIT = 2000$  és una bona opció, ja que ofereix un bon equilibri entre el temps d'execució mitjà i el valor del heurístic.

#### 4.4. EXPERIMENT 4

Aquest experiment el vam realitzar amb els operadors i la generació inicial decidits en els experiments 1 i 2. Ja que eren amb els que més bons resultats hem obtingut. Aleshores hem realitzat dos tipus d'experiments diferents:

El primer experiment, va ser deixar el nombre de paquets amb 50, la semilla també amb 1234 i anem augmentant la proporció del pes transportable de les ofertes. Aleshores, hem realitzat una taula per veure com influència l'augment d'aquesta proporció en el temps d'execució de la nostra búsqueda. Amb aquesta taula hem realitzat el gràfic que podem observar a la *figura6*.

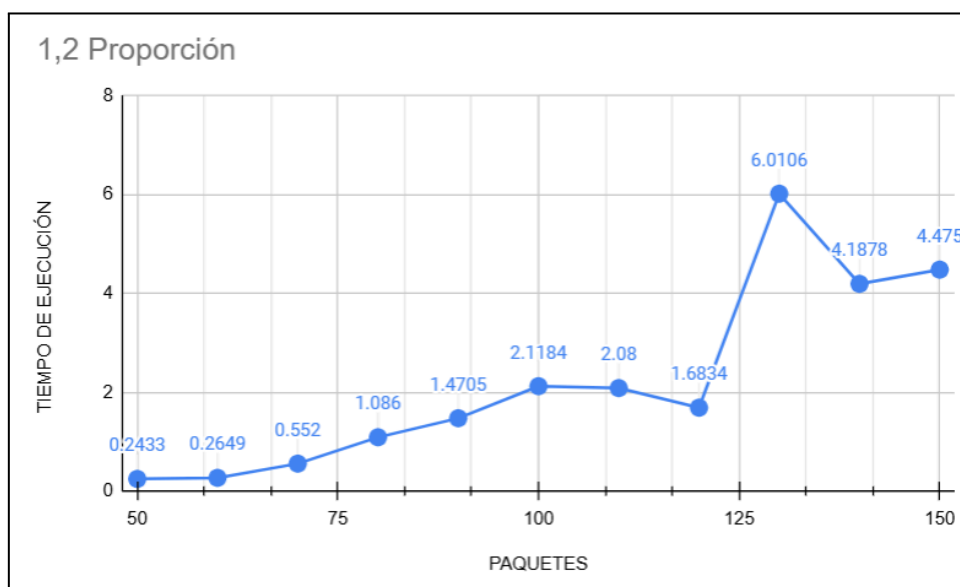


*figura6*

Podem observar que a mesura que anem augmentant la proporció, va augmentant el temps d'execució, però això no coincideix en tots els casos. Per exemple, dels dos primers punts (proporció de 1.2 a 1.4), veiem com disminueix, però en canvi, el de 1.6, és a dir el següent punt, augmenta més respecte els dos anteriors. Llavors, veiem que quan augmentem la proporció, va augmentant de mica en mica el temps. A més, veiem que hi ha dos valors, el setè i vuitè punt, que es mantenen amb un temps d'execució més o menys iguals (1.4615 i 1.4575). Llavors veiem com el temps torna a augmentar, però quan la proporció és igual a 3 disminueix 0.2 respecte l'anterior.

Nosaltres ens esperàvem que conforme anéssim augmentant la proporció, el temps d'execució augmenta sempre, ja que tenen una proporció entre els dos. En canvi, com podem observar, veiem que hi ha moments on no augmenta molt significativament, sinó que és gairebé constant, fins i tot com podem observar arriba a disminuir. Això podria ser degut a que com més gran és la proporció, més tarda en trobar la solució més òptima i per això el temps d'execució és més gran

El segon experiment que vam realitzar, va ser deixar la proporció de pes a 1.2, i anar augmentant el nombre de paquets de 10 en 10 començant per 50 paquets fins a arribar a 150 paquets. Per tenir coherència, vam deixar la semilla a 1234. El temps d'execució respecte le nombre de paquets el podem observar en el gràfic de la *figura7*.



*figura7*

Com podem observar a mesura que augmenta el nombre de paquets el temps d'execució augmenta, això en el cas de 50 a 100 paquets. Llavors, veiem que al augmentant els paquets a 100 i 110 el temps no varia molt, però que llavors al 120 disminueix una mica respecte el nombre de paquets anterior.

Aleshores al 130 augmenta el temps notablement i llavors al 140 disminueix i amb 150 paquets el temps no varia molt significativament, respecte quan utilitzem 140.

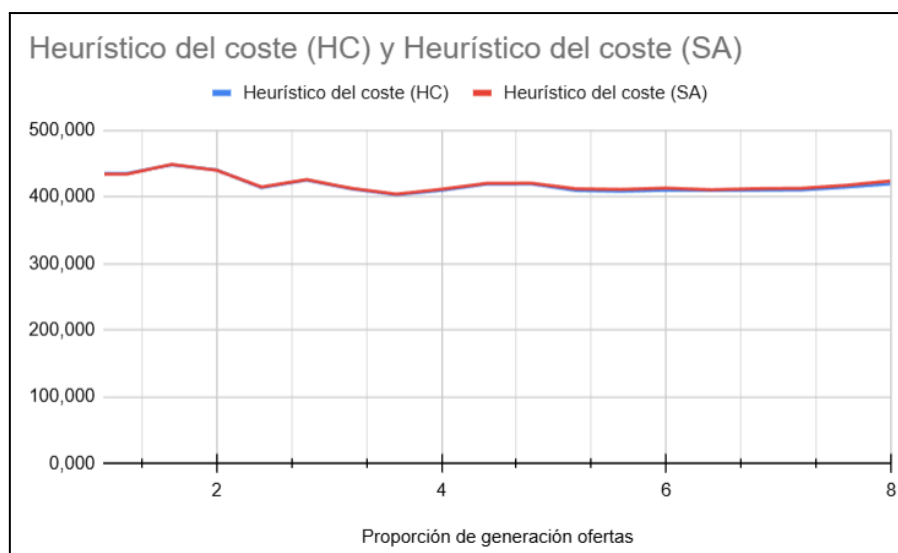
Els resultats que ens esperàvem obtenir, era que hi hagués un augment més proporcional i que no tinguéssim un pic (màxim). En canvi hem obtingut que amb una proporció de 1.2 veiem que el temps d'execució mínim és quan tenim 120 paquets i que després quan augmentem el nombre de paquets el temps augmenta molt significativament.

Amb aquests resultats obtinguts en els dos experiments, podem observar que existeix una relació entre la proporció del pes transportable i el nombre de paquets. Podem observar com existeix una relació, però no lineal. Sembla que a mesura que anem augmentant el nombre de paquets el temps d'execució també augmenta mantenint una proporció de 1.2 i que quan anem augmentant la proporció també augmenta el temps d'execució. Llavors veiem que hi ha moments on no es compleix aquesta relació com quan augmentem a 130 paquets i això podria significar que amb una proporció de 1.2 amb tants paquets tarda molt més en trobar la solució més òptima.

#### 4.5. EXPERIMENT 5

Vam realitzar aquest escenari amb 50 paquets i amb una semilla de 1234. Aleshores per veure com influenciaven els costos mentre anem augmentant la proporció per al pes transportable per les ofertes, hem utilitzat l'heurístic que només té en compte els costos, no tenim en compte la felicitat. A més, hem volgut observar com es modificaven els heurístics en el cas de Hill Climbing i de Simulated Annealing, mentre anem augmentant la proporció de la generació de les ofertes.

Els resultats obtinguts mentre anem augmentant la proporció de 0.4 en 0.4 començant per 1 fins arribar a 8, han estat els que podem observar a la *figura 8*. Per l'augment de totes les proporcions, podem observar com l'heurístic del cost, és similar tant per el Hill Climbing com per el de Simulated Annealing, ja que podem observar com les línies es sobreposen unes respectes les altres.

*figura8*

Aleshores, si analitzem la *figura8*, que de la proporció de 1.2 fins a proporció igual a 2, l'heurístic en els dos casos va augmentant de 435,009 a 440,26. Si és més petit de 1.2 heurístic no canvia. Llavors per molt que anem augmentant la proporció de la generació de les ofertes dins del rang de 1.2 a 2, veiem que l'heurístic va augmentant i disminuint.

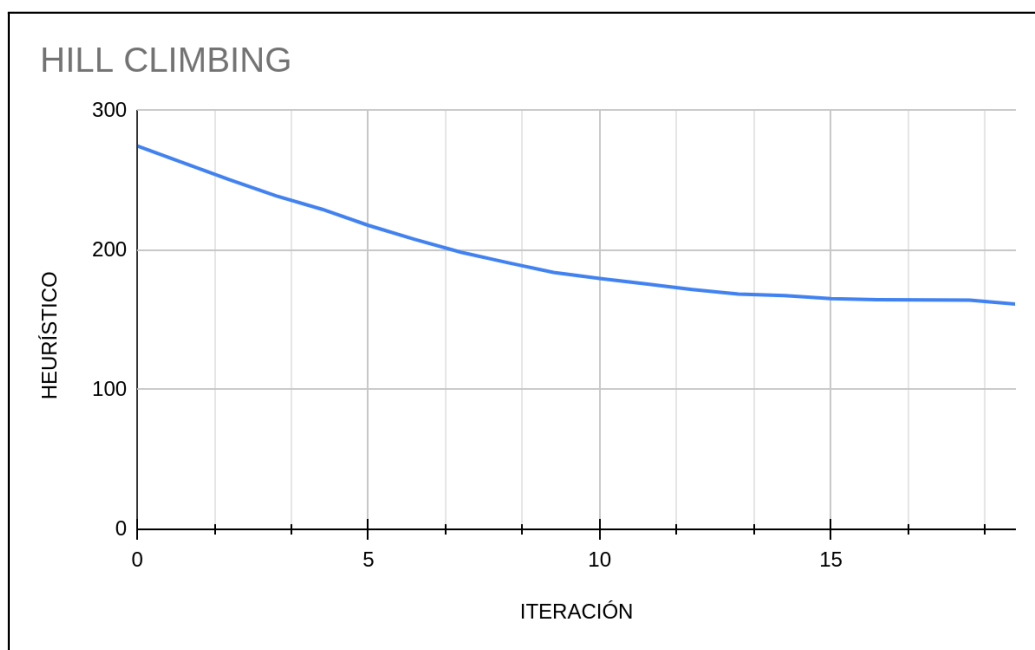
Llavors, per molt que anem augmentant la proporció amb més de 2, heurístic millora respecte l'interval anterior ja que pren un valor més baix, però no millora molt més conforme anem augmentant la proporció de la generació de les ofertes, ja que podem observar com entre el 4 i el 6 ens dóna gairebé el mateix valor.

S'esperava que els dos heurístics donessin el mateix ja que els costos tinguent totes les mateixes condicions de creació, son les mateixes. Però el que no esperàvem era que conforme anem augmentant més la proporció heurístic no millores més significativament. Això ens fa pensar que conforme més gran anem fent la proporció del pes de les ofertes, segons la nostra representació i càlculs els costos no van canviant molt, ja que tenim el mateix nombre de paquets i aquests s'han de repartir igualment en l'oferta segons la prioritat i el pes que els hi toca.

#### 4.6. EXPERIMENT 6

Aquest experiment tracta de veure com evoluciona l'heurístic conforme avança l'execució del Hill Climbing. L'experiment el vam realitzar amb un escenari de 50 paquets, una proporció del 1.2 i una semilla de 1234. Els operadors que hem utilitzat són els que vam decidir més oportuns un cop realitzat el primer experiment. L'heurístic utilitzat és el que combina la felicitat i els costos de transport. Per

poder veure com evoluciona l'heurístic al llarg de les iteracions, hem hagut de fer una funció `hill_climbing` en la qual es printés el valor de l'heurístic en cada iteració de la funció i hem afegit u mètode `neighbour` a la classe `AzamonProblem` que retornés totes les solucions veïnes de la solució actual. També hem comprovat que el valor de l'heurístic ens donés el mateix que fent servir la funció de `aima.search`.



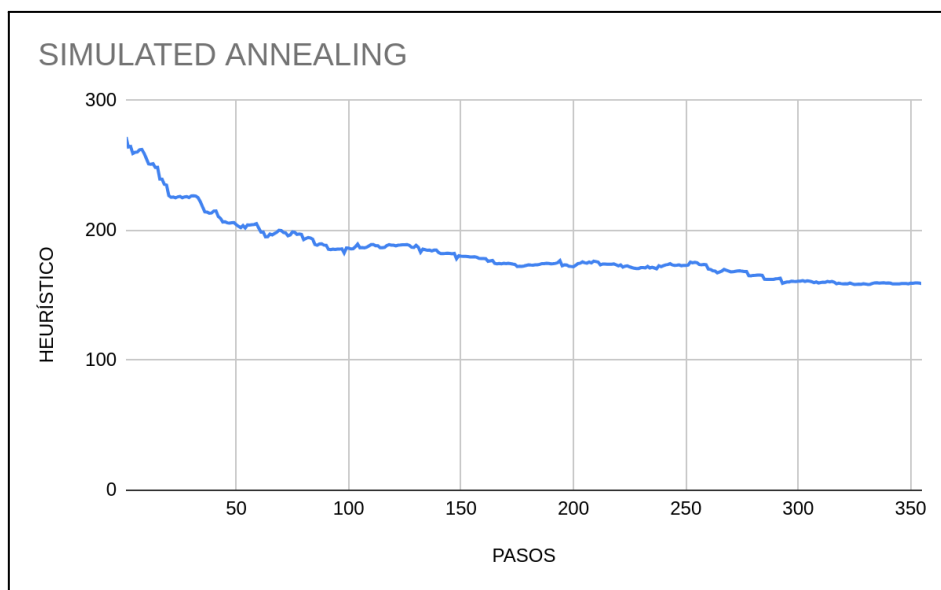
*figura9*

Així és com evoluciona el nostre heurístic al llarg de les iteracions del Hill Climbing. El resultat és com ens esperàvem, ja que va disminuint conforme avancem en l'execució fins trobar un mínim local. A més a més, hem pogut comprovar que, en aquest escenari en concret, només li fan falta 19 iteracions per arribar a aquest resultat, per això triga tan poc el temps d'execució.

#### 4.7. EXPERIMENT 7

Aquest experiment tracta de veure com evoluciona l'heurístic conforme avança l'execució del Simulated Annealing, i així poder-ho comparar amb el Hill Climbing. En aquest experiment, hem fet el mateix escenari que en l'anterior per poder comparar-los entre ells i veure les diferències entre Hill Climbing i Simulated Annealing. Els paràmetres que hem utilitzat amb el Simulated Annealing han sigut els que hem escollit a l'experiment 3:  $k=1$ ,  $\text{lam}=0,0005$  i 5000 iteracions. Igual que en l'experiment anterior, hem hagut de fer una funció `simultaed_annealing_with_print` que anés escrivint per pantalla com anava canviant el valor de l'heurístic al llarg de les iteracions. També hem hagut d'afegir un mètode més (`neighbors`) a la classe `AzamonProblemSa`. En aquest cas, no hem pogut

comprovar que le heurístic ens donés el mateix que amb la funció de l'aima.serch perquè depèn de valors randoms, per tant, no surt el mateix sempre.



*figura10*

En aquest gràfic veiem l'evolució de l'heurístic. Veiem que sí és com ens podem esperar, ja que té una tendència general decreixent però no sempre l'heurístic disminueix. També podem observar que, de 5000 iteracions, només canvia l'heurístic en menys de 360, perquè a la resta no escollirà la solució veïna aleatòria generada; per això amb aquest algoritme triga molt més i el resultat no és tant diferent al del Hill Climbing.

#### 4.8. EXPERIMENT 8

- **Sense fer cap experiment, com canviarien les solucions del resultat si variem el preu de l'emmagatzematge fixe diari de 0,25 augmentant-lo o disminuint-lo?**

Si disminuïm aquest preu, la solució tindrà més paquets a ofertes que triguen més dies, ja que tenen més dies d'emmagatzematge i el preu d'aquest es menor que abans. En canvi, la pujada d'aquest preu provocarà que les ofertes amb més dies tinguin menys paquets, ja que emmagatzemar aquests suposarà més cost que abans i augmentarà la felicitat dels usuaris, els quals tindran els seus paquets abans.

#### 4.9. EXPERIMENT ESPECIAL

Aquest experiment es va realitzar amb un escenari que inclou 50 paquets i una proporció del 1.2 per al pes transportable per les ofertes. Per tal d'assegurar la coherència en els resultats es va utilitzar la semilla 1234. Els operadors i la inicialització utilitzats són els emprats en els experiments 1 i 2.

Els resultats obtinguts per a la Solució Inicial i per la Solució obtinguda amb Hill Climbing (HC) són els que es mostren en la taula de la *figura 11*.

EXPERIMENT 9	EXPERIMENT ESPECIAL					
	HEURÍSTIC COST	TEMPS MITJÀ D'EXECUCIÓ	HEURÍSTIC FELICITAT	TEMPS MITJÀ D'EXECUCIÓ	HEURÍSTIC	TEMPS MITJÀ D'EXECUCIÓ
SOLUCIÓ INICIAL	457,955	0,0000884213	0	0,0000884213	274,773	0,0000884213
SOLUCIÓ HC	435,009	0,04272	-38	0,09794	161,306	0,2235

*figura 11*

Per aquest experiment hem realitzat l'execució del programa amb tres heurístics diferents:

- HEURÍSTIC COST: fa l'optimització del cost de transport i emmagatzematge sense tenir en compte la felicitat dels clients
- HEURÍSTIC FELICITAT: fa l'optimització de la felicitat dels clients sense tenir en compte el cost de transport i emmagatzematge
- HEURÍSTIC: fa la combinació d'ambdós factors mitjançant una ponderació on el 60% del pes recau sobre els costos de transport i emmagatzematge i el 40% representa la felicitat dels clients.

(Per la resta d'experiments s'ha utilitzat únicament el tercer heurístic, ja que equilibrar el cost i la felicitat)

Amb els resultats obtinguts observem que, en referència amb heurístic que té en compte sols el cost, aquest millora significativament el cost del transport, passat de 457.955 (en l'inicial) a 435.009 . Per tant, això ens indica una reducció notable en els costos operatius del sistema de transport. Però, en comparació amb els altres heurístics és el que té un canvi menor, i això ho podem atribuir a la natura de la solució inicial, ja que aquesta agrupa els paquets entre ofertes tenint en compte la prioritat i el pes màxim.

Si ens fixem en l'heurístic de felicitat, aquest ha aconseguit augmentar la felicitats dels clients (passant de zero o -38), la qual cosa indica una millora en la qualitat del servei considerable, ja que la solució inicial no té en compte res de la felicitat dels clients.

L'últim heurístic, el que conté la ponderació entre els costos i la felicitat, on els costos representen un 60% i la felicitat un 40%, també millora, molt considerablement, respecte al valor de la solució inicial, passant de 274.773 a 161.3059 .

Pel que fa als temps d'execució, com era d'esperar, per a la solució inicial el temps és mínim, 0.0000884213 segons (0.0884213 mil·lisegons), ja que és simplement la solució per defecte sense cap optimització. En canvi, la solució un cop s'ha aplicat Hill Climbing, el temps augmenta, ja que el procés d'optimització busca millores iterativament, de manera que amb l'heurístic que té en compte tant el cost com la felicitat (el que més tarda en executar-se) té un temps d'execució mitjà de 0.2235 segons (223.5 mil·lisegons). Un cost raonable tenint en compte les millores aconseguides en el valor del heurístic.

S'esperava que l'aplicació de l'algoritme Hill Climbing millorés tant el cost total de transport com la felicitat, a canvi d'un temps d'execució més llarg. Els resultats obtinguts confirmen aquestes expectatives, ja que gràcies a les aplicacions dels heurístics i l'augment (raonablement petit) de temps s'ha aconseguit una millora molt considerable en la qualitat de la solució.

## 5. COMPARACIÓ DELS RESULTATS OBTINGUTS ENTRE HILL CLIMBING I SIMULATED ANNEALING

En aquesta comparació entre els algorismes Hill Climbing (HC) i Simulated Annealing (SA), podem observar les diferències de rendiment i comportament en els gràfics proporcionats.

- **Hill Climbing (HC):**

- Al gràfic de Hill Climbing (Experiment 6 - *figura9*), observem que l'heurístic disminueix ràpidament en les primeres iteracions, amb una caiguda pronunciada del valor heurístic cap a un òptim local. Això indica que HC és un algorisme molt directe, que intenta trobar la millor solució possible amb rapidesa.
- Tot i això, aquest enfocament té un risc significatiu de quedar-se atrapat en un mínim local o en una meseta, ja que Hill Climbing no té mecanismes per "sortir" de solucions no òptimes. Aquest fet es pot veure en el gràfic on, després d'un cert nombre d'iteracions, el valor de l'heurístic es manté més o menys constant sense millores substancials, la qual cosa indica que l'algorisme ha arribat a una solució estable però possiblement subòptima.

- **Simulated Annealing (SA):**

- En el gràfic de Simulated Annealing (Experiment 7 - *figura10*), es pot veure un comportament una mica diferent. Al principi, el valor va baixant com en altres algorismes, però després es manté amb pujades i baixades més suaus i regulars. Aquest patró mostra que SA permet, de tant en tant, fer moviments cap a solucions una mica pitjors, cosa que ajuda a explorar millor totes les possibles opcions.
- Aquesta característica d'exploració fa que SA pugui evitar caure en mínims locals, ja que es permet "escapar" d'aquests per intentar trobar una millor solució a llarg termini. No obstant això, aquest algorisme requereix més iteracions per assolir un resultat òptim i estable, i tal i com es mostra en el gràfic, SA tarda més a arribar a una heurística gairebé òptima en comparació amb HC.

En resum, tot i que ambdós algorismes arriben a un valor d'heurístic semblant, Hill Climbing ho fa amb menys iteracions mentre que Simulated Annealing necessita més temps per assolir una solució de qualitat similar però amb una major garantia d'haver evitat mínims locals.

## 5.1. CONCLUSIONS DELS ALGORISMES

Tal i com hem enfocat i plantejat aquest pràctica, podem extreure com a conclusió general que Hill Climbing és una millor opció per les raons següents:

1. **Rapidesa per arribar a una solució:** Hill Climbing és un algorisme que avança directament cap a una solució millor en cada pas, sense fer pauses per explorar altres opcions. Això fa que arribi a una resposta molt ràpidament, cosa que resulta ideal si ens interessa trobar una solució ràpida i no ens importa tant explorar totes les possibilitats. En aquest experiment, aquesta rapidesa ajuda a reduir el temps necessari per aconseguir una resposta acceptable.
2. **Qualitat de la solució final:** Encara que Simulated Annealing té l'avantatge d'evitar quedar-se atrapada en mínims locals o solucions no òptimes, en aquest experiment concret no hi ha una gran diferència en la qualitat de la solució que s'obté. Hill Climbing, tot i ser més directe i menys flexible, aconsegueix un resultat gairebé igual de bo amb menys passos.

En resum, per a aquest problema específic, Hill Climbing ens proporciona una solució similar a la de Simulated Annealing, però amb menys iteracions i, per tant, en menys temps. Això el fa més eficient per a aquesta situació on la rapidesa és més important que l'exploració extensa.

## 6. CONCLUSIONS DELS TREBALL

Al llarg d'aquest projecte, hem explorat diferents tècniques d'optimització per resoldre un problema de búsqueda, centrant-nos en la distribució eficient de paquets per minimitzar els costos i maximitzar la satisfacció dels clients. L'ús dels algorismes de cerca local, com el Hill Climbing i el Simulated Annealing, ens ha permès analitzar l'eficàcia de diferents estratègies en funció de diverses heurístiques.

Els resultats obtinguts i els gràfics extrets ens han revelat diferències clares entre els dos algorismes. Hill Climbing ens ha demostrat ser molt ràpid per arribar a solucions acceptables, especialment útil en situacions en què necessitem resultats immediats i no és essencial explorar a fons totes les alternatives. En canvi, Simulated Annealing es mostra més flexible, ja que permet explorar opcions menys òptimes en certs punts, fet que l'ajuda a evitar quedar-se encallat en solucions locals. Això resulta beneficiós quan el problema és més complex i requereix una solució més precisa o global.

A més, hem comprovat, a través de l'experimentació, que combinar diferents operadors de manera adequada i ajustar l'equilibri entre els costos i la satisfacció dels clients és essencial per obtenir solucions més completes i eficients. Aquesta combinació permet trobar un punt d'equilibri que no només optimitza els costos, sinó que també assegura un nivell alt de satisfacció, contribuint a una distribució de recursos que compleix amb els objectius plantejats de manera més efectiva i alineada amb les prioritats del projecte.

A més, aquest projecte ens ha permès consolidar les nostres habilitats en Python i entendre més profundament els fonaments dels algorismes de cerca local. El treball en equip i la resolució conjunta dels problemes tècnics han estat elements clau per assolir els objectius d'aquest treball.

En resum, els resultats obtinguts mostren que cada algorisme té avantatges particulars segons les condicions del problema. Mentre que Hill Climbing destaca per la seva rapidesa en situacions menys complexes, el Simulated Annealing és més eficaç en contextos on es requereix evitar òptims locals per assolir solucions de millor qualitat. Així, l'elecció de la tècnica més adequada ha de considerar un equilibri entre la rapidesa d'execució i la precisió en la solució, depenent de les prioritats del problema en qüestió.