PRACTICA 1: Busqueda local
Azamon

Lluís Itarte Sahun
Mikel Torres Martínez
Andreu Corden Moragrega
Lluc Martinez Busquets

October 28, 2024

Contents

1 Introducció

Aquest treball presenta el desenvolupament d'una pràctica sobre algoritmes de recerca local, en què l'equip format per Lluc, Lluís, Mikel i Andreu ha col·laborat per aplicar aquests algoritmes a un problema d'optimització logística. Aquesta pràctica s'ha desenvolupat en el marc de l'assignatura d'Intel·ligència Artificial de l'enginyeria informàtica de la Universitat Politècnica de Catalunya (UPC).

La recerca local s'utilitza habitualment per resoldre problemes complexos d'optimització, especialment aquells amb grans espais de cerca i amb múltiples solucions possibles. En aquest context, el treball proposa una sèrie d'experiments per analitzar l'eficàcia de dos algorismes principals: el Hill Climbing i el Simulated Annealing. La pràctica es centra en un cas fictici d'una empresa de distribució anomenada Ázamon, on s'estudien les solucions òptimes per a la distribució diària de paquets, tenint en compte factors com els costos de transport, d'emmagatzematge i la satisfacció dels clients.

El nostre objectiu en aquesta pràctica és dissenyar, implementar i avaluar diverses estratègies de generació d'estats inicials, operadors i funcions heurístiques, així com comparar els resultats i l'eficàcia dels dos algorismes esmentats en diferents escenaris. Això ens permetrà entendre les seves aplicacions i limitacions en problemes reals d'optimització.

2 El problema

Explicació del problema i les seves parts.

2.1 Descripció del Problema

El problema ens planteja una companyia fictícia de distribució de paquets que cada dia organitza els paquets que ha d'enviar a una ciutat. Per cada ciutat, rep vàries ofertes d'empreses de paqueteria que poden fer aquests enviaments. Cada oferta té un pes màxim que pot transportar, entre 5 i 50 kg amb intervals de 5 kg, un preu per kilogram transportat i el nombre de dies en els quals els paquets arribaran (entre 1 i 5 dies).

La companyia per cada paquet sap el seu pes i la seva prioritat. Els pesos dels paquets són múltiples de 0.5 i no poden pesar més de 10 kg.

Pel que fa a la prioritat de cada paquet, cada usuari pot escollir entre 3 prioritats pel seu paquet, cada una amb un preu. La prioritat 1 assegura que el paquet serà entregat l'endemà amb un cost de 5 euros addicionals. La prioritat 2 (2-3 dies) assegura que el paquet arribarà en com a maxim amb un termini de 3 euros amb un cost de 3 euros addicionals. L'última prioritat 3 (4-5 dies) assegura que el paquet arribarà com a màxim en 5 dies amb un cost d'1,5 euros addicionals.

A més a més, les companyies de transport no venen a recollir els paquets immediatament. Això provoca que els paquets hagin d'estar emmagatzemats fins que els recullin, amb el cost addicional de 0.25 kilograms/dia. És a dir un paquet de prioritat 2 es recull un dia després que s'hagi assignat l'enviament i un paquet de prioritat 3 es recull 2 dies després.

Per acabar si els paquets arriben abans de la data seleccionada, els clients seran més feliços. Per quantificar la felicitat es la diferencia amb la data seleccionada i la data d'arribada del paquet.

Aleshores el problema que se'ns planteja és obtenir per tots els paquets a enviar en un dia i a una ciutat concreta, una assignació a les ofertes d'enviament d'aquell mateix dia, tenint en compte els costos associats al transport i emmagatzematge, i la felicitat.

2.2 Cerca Local

Per poder trobar una solució aquest problema i trobar una assignació de paquets a ofertes amb el mínim cost tenint en compte l'esmentat en l'apartat anterior, utilitzarem un algorisme de cerca local com ara el HIll CLimbing o el Simulated Annealing, A.

Els algorismes de cerca local consisteixen en trobar una solució inicial i a partir d'aquesta explorar estats pròxims per trobar la solució òptima. Aquests algorismes tenen una complexitat temporal molt inferior a metodes com ara el backtracking, ja que el backtracking explora totes les possibles solucions, mentre que els algorismes de cerca local els estats més propers i va avançant fins que trobar la solució òptima local.

Aquesta metodologia afavoreix molt als problemes que compleixen les següents condicions:

- És un problema d'optimitzacio amb l'objectiu d'obtenir el millor estat segons la funcio heuristica.
- El cami fins arribar a un estat final es irrellevant.

Amb el nostre problema podem veure que compleix la primera condició, ja que volem minimitzar el preu i maximitzar la felicitat. També es pot observar que la segona condició es compleix, ja que no volem saber el camí de com hem d'assignar els paquets sinó que només ens interessa l'assignació final.

2.3 Espai de Cerca

Podríem definir l'espai de cerca com totes aquelles solucions candidates, o estats finals, que te el nostre problema. És important saber l'espai de solucions, ja que si és un espai de solucions polinòmic voldria dir que podríem resoldre de manera factible el problema amb força bruta. També ens donaria una idea de la magnitud del problema que estem afrontant.

L'espai de cera per aquest problema, donat que tenim n paquets i m ofertes, seria el conjunt de totes les possibles assignacions vàlides dels paquets a les ofertes.

- Cada paquet P té m possibles ofertes a les quals es poden assignar, llavors per n paquets el nombre total d'assignacions possibles, sense considerar les restriccions, és de m a la n.
- si tenim en compte les restriccions com el pes màxim de les ofertes i els temps d'entrega la magnitud de l'espai de cerca segueix sent exponencial en funció de n i m.

3 La IA

3.1 Implantació de L'estat

Per implementar el nostre estat hem fet servir els següents atributs:

- **static Paquetes paquets** És una llista de tots els paquets a entregar que estan ordenats per prioritat, aquesta llista s'utilitzarà per crear les solucions inicials, indexar i saber el pes o prioritat d'un paquet en concret.
- **static Transporte transport** És una llista de totes les ofertes ordenades primer per prioritat i després per pes, aquesta llista s'utilitzarà per crear les solucions inicials, indexar i saber el preu per kilogram, el pes màxim que pot transportar i els dies fins que es faci l'entrega
- **private ArrayList ¡Integer¿ assignacions** Serà la llista que tindrà una mida de paquets.size() i mostrarà les assignacions de paquets a ofertes, aquesta arrayList es farà servir especialment en els operadors, per canviar les assignacions dels paquets.
- **private ArrayList ¡Double¿ CapRestantActual** És la llista que té mida transport.size() i mostra la capacitat restant que els hi queda a les ofertes, aquesta llista es farà servir tant per calcular les solucions inicials com en els operadors per comprovar sí el paquet cap dins l'oferta.
- **private int Felicitat** Serà la variable que emmagatzema la suma de la felicitat total.
- **private double costTransport;** Serà la variable que emmagatzema la suma total del cost total del transport.
- **private double costMagatzem;** Serà la variable que emmagatzema la suma total del cost total d'emmagatzemar els paquets.

Aquestes tres ultimes variables, Felicitat, CosTransport i CostMagatzem seran modificades en el procés de crear la solució i en el procés d'aplicar els operadors.

Tant la llista de paquets i la llista d'ofertes són variables estàtiques, ja que són 2 vectors que mai canvien, d'aquesta manera només es crea una còpia de la variable per totes les instàncies de la classe, el que pot estalviar memòria així fent el codi més eficient.

3.2 Operadors

Per poder modificar l'estat que s'ha implementat farem servir els següents operadors.

mover-paquet

- *Paràmetres:* l'índex del paquet *paq-idx* i l'índex de l'oferta *n-of*
- *Precondició:*
  - L'índex del paquet *paq-idx* i l'oferta *n-of* indexada han d'estar dins del rang de les llistes paquets i transport.
  - El paquet *paq-idx* no ha d'estar ja assignat a l'oferta *n-of*
  - La prioritat del paquet *paq-idx* ha de ser compatible amb l'oferta *n-of.*
  - El paquet *paq-idx* ha de cabre dins l'oferta *n-of.*

- *Postcondició:*
  - Si les condicions de la precondició es compleixen, el paquet *paq-idx* es reassigna a l'oferta *n-of*, i s'actualitzen la felicitat, costTransport i CostMagatzem.
  - Si alguna condició falla l'operador retornar *false* y no es realitzarà cap canvi en les assignacions o en l'estat.
- *Factor de Ramificacio:* En el cas pitjor si tots els paquets poden ser assignats a totes les ofertes. Per tant, es de l'ordre de $O(n \cdot m)$, on n son els paquets i m les ofertes.

intercambiar-paquete

- *Paràmetres:* L'índex paquet *paq1* i l'índex del paquet *paq2*.
- *Precondició:*
  - *Paq1* i *paq2* han d'estar dins del rang de la llista *paquets*.
  - Els paquets *paq1 paq2* no poden estar assignats a la mateixa oferta.
  - La prioritat de *paq1* ha de ser compatible amb la oferte de *paq2* i viceversa.
  - Els paquets han de caber dins l'oferta de desti.
- *Postcondició:*
  - Si es compleix la precondició, s'intercanviarà les assignacions de *paq1* i *paq2*, i i s'actualitzen la felicitat, costTransport i CostMagatzem.
  - Si la precondició no es compleix, la funció retornara *false* i no es farà cap canvi de les assignacions.
- *Factor de ramificació:* És determinat pel nombre de combinacions de paquets que es poden intercanviar, supossant que tots els paquets es poden intercanviar entre si i que tenim un total de n paquets, el factor de ramificació serà de l'ordre $O(\frac{N \cdot (N-1)}{2})$.

intercambiar-2x1-paquete

- *Paràmetres:* Els índexs dels 3 paquets, *paq1*, *paq2* i *paq3*.
- *Precondició:*
  - Els paquets *paq1*, *paq2* i *paq3* han d'estar dins del rang de la llista *paquets* i assignats a una oferta.
  - Els paquets *paq1* i *paq2* han d'estar assignats a la mateixa oferta i *paq3* a una oferta diferent.
  - La prioritat dels paquets *paq1* i *paq2* ha de ser compatible amb l'oferta *paq3* i viceversa.
  - La capacitat de les ofertes han de ser adequades per poder fer l'intercanvi de paquets.
- *Postcondició:*
  - Si les Precondició compleix *paq1* i *paq2* es reassignaran a l'oferta de *paq3*, i *paq3* a l'oferta de *paq1* i *paq2*.
  - Si la precondició no es compleix la funció retorna *false* i no cap canvi d'assignacions.
- *Factor de ramificació:* És determinat pel nombre de combinacions de paquets que es poden intercanviar 2 per 1, suposant que tots els paquets es poden intercanviar entre si i que tenim un total de n paquets, el factor de ramificació serà de l'ordre $O(\frac{N \cdot (N-1) \cdot (N-2)}{6})$.

Creiem que aquests 3 operadors ens poden ajudar a trobar millores de l'estat, i que podem explorar tot l'estat de solucions.

3.3 Funcions Heurístiques

Per fer els experiments vam decidir implementar 2 funcions heurístiques per poder observar la diferència que podria provocar al trobar la solució.

Heurística 1: Com l'objectiu principal del problema és minimitzar tant el Cost de transport com el cost d'emmagatzematge, aleshores vam decidir fer una heurística que només tingui en compte el Cost total.

Heurística 2: En aquest cas vam decidir implicar la felicitat en el càlcul de l'heurística, ja que un objectiu secundari era poder maximitzar la felicitat. Aleshores per fer l'heurística vam decidir sumar el cost total amb la felicitat, però només amb això no n'hi havia prou, ja que el cost era sempre més gran que la felicitat llavors aquesta mai influenciava l'heurística, d'aquesta manera vam afegir ponderacions, tant pel cost total com per la felicitat per poder anivellar la influència que té cada variable sobre l'heurística.

Totes dues heurístiques són molt senzilles, però el seu funcionament és bo, ja que a part que són heurístiques fàcils de calcular, compleixen la seva funció de minimitzar el cost i maximitzar la felicitat. També si es volgués només maximitzar la felicitat sense tenir en compte el preu total seria tan fàcil com canviar la ponderació del cost total a 0.

3.4 Solucions Inicials

A l'hora d'implementar la classe Estado, vam implementar 3 solucions inicials diferents. Aquestes tenen els requisits següents:

- Enviar tots els paquets
- Complir amb els terminis d'enviament

Les tres solucions utilitzen una implementació basada en *Backtracking* però prioritzen diferents assignacions. Les opcions són les següents:

1. Assigna paquets de major prioritat a ofertes mes ràpides
2. Assigna paquets més grans a ofertes més econòmiques
3. Assigna els paquets de major prioritat i més grans a ofertes més ràpides i econòmiques

4 Experiments

4.1 Experiment 1

L'objectiu d'aquest experiment es comparar els diferents operadors que hem explicat anteriorment en l'apartat **3.2**. Aquest s'han combinats de 4 formes diferents per establir les següents opcions:

1. Només moure paquets
2. Només intercanviar paquets
3. Combinat moure i intercanviar 1x1
4. Combinat moure, intercanviar 1x1 i intercanviar 2x1

Tenint en compte els diferents operadors, plantegem que no tots obtindran els mateixos resultats, tant en minimització de costos, com en el temps d'execució.

Hipòtesis: L'operador 4 serà el que obtingui costos més minimitzats, ja que és el que més espai de cerca inclou. Per altre banda, aquest tindrà un temps d'execució més elevat al ser el que més opcions ha d'analitzar.

Metodologia:

- Fixarem la quantitat de paquets a 100, la proporció a 1.2
- Fixem l'heurística a la 1 i el la solució inicial també a la 1
- Utilitzarem l'algorisme de *Hill Climbing*
- Utilitzarem 10 llavors aleatòries
- Executarem aquestes per a cada operador
- Mesurarem el temps d'execució, el cost obtingut i els nodes estesos

Resultats:
Un cop executat hem realitzat els següents gràfics, els quals representen les dades obtingudes. Aquestes són **temps d'execució, el cost obtingut i els nodes estesos**.

Com podem observar els tres primers operadors tenen un temps reduït, però l'operador 4 triga considerablement més. Aquests temps són els següents:

Operador 1	**Operador 2**	**Operador 3**	**Operador 4**
67.25	61.2	187.25	3407.9

Table 1: Resultats en ms del temps d'execució

Tot i això, observem que el cost més elevat l'obté el que només pot realitzar intercanvis. Però el cost més reduït l'obté l'operador 4. Els altre dos operadors, no obtenen gaire diferencia entre ells.

Operador 1	**Operador 2**	**Operador 3**	**Operador 4**
1173.73	1339.85	1171.95	1057.30

Table 2: Resultats del cost total per operador

En quant a els nodes estesos, veiem que l'operador 2 no ha fet gaires canvis de l'estat inicial. Però la resta si han tingut un recorregut de un 60-80 nodes, sent l'operador 4 el que més nodes estén. Els resultats son els següents:

Operador 1	**Operador 2**	**Operador 3**	**Operador 4**
61.95	15.25	64.5	73.95

Table 3: Resultats dels nodes estesos

Amb les dades obtingudes també hem realitzat els següents gràfics:

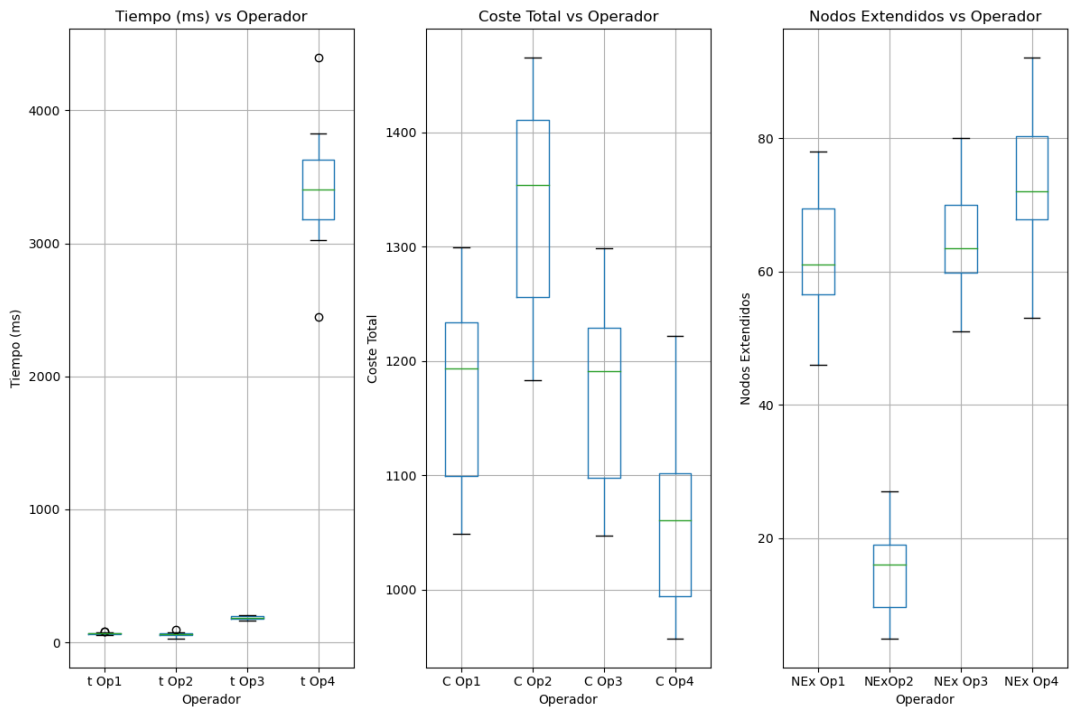

Figure 1: Resultats depenent del operador utilitzat

Conclusions

Observant les dades observem que si volem minimitzar el cost el millor operador és el 4. Però haurem de tenir en compte que aquest té un temps d'execució més elevat. Això en mostres més grans pot ser un problema, però ja ho mirarem en un altre experiment.

4.2 Experiment 2

L'objectiu d'aquest experiment és comparar les diferents solucions inicial plantejades a l'apartat **3.4**. En aquest es plantejaven, depenent de les prioritats d'assignació, les 3 solucions inicial següents:

1. Prioritat (-) Temps
2. Mida (-) Preu
3. Prioritat + Mida (-) Temps + preu

Considerant les diferents solucions inicials, considerem que aquestes alteraran els resultats de la cerca. Per tant, ens plantegem comparar-los.
Hipòtesis: Creiem que la solució 3 serà la que obtindrà costos més baixos, tot i que aquest tingui un temps de càlcul major. Això es degut a que la solució ideal, està més minimtzada que la resta de solucions.

Metodologia:

- Fixarem la quantitat de paquets a 100, la proporció a 1.2
- Fixem l'heurística a la 1 i l'operador al 4
- Utilitzarem l'algorisme de *Hill Climbing*
- Utilitzarem 20 llavors aleatòries
- Executarem aquestes per a cada solució inicial
- Mesurarem el temps d'execució, el cost obtingut i els nodes estesos

Resultats:
Un cop executat hem realitzat els següents gràfics, els quals representen les dades obtingudes. Aquestes són **temps d'execució, el cost obtingut i els nodes estesos**.

Primer observem que al contrari del que pensàvem, el temps de cerca del mètode 3, no és superior al mètode 1. A més observem que el mètode 2, té un temps de cerca més reduït. El resultats són els següents:

Solució Inicial 1	**Solució Inicial 2**	**Solució Inicial 3**
3296.65	650.2	3070.7

Table 4: Resultats dels nodes estesos per cada Solució Inicial

En el cas del cost, observem que no hi ha gaire diferència entre la solució 1 i 3. Però la solució 2, obté uns costos més elevats. Els resultats són el següents:

Solució Inicial 1	**Solució Inicial 2**	**Solució Inicial 3**
1086.37	1196.14	1101.80

Table 5: Resultats del Cost Total per cada Solució Inicial

En quant a nodes estesos, observem una similitud amb el temps, on la solució 1 i 3 tenen un número de nodes similar, sent la solució 3 menor. I per altre banda la solució 2 tenint un número de nodes molt més reduït.

Solució Inicial 1	**Solució Inicial 2**	**Solució Inicial 3**
69.85	1196.14	1101.80

Table 6: Resultats dels Nodes Estesos per cada Solució Inicial

Amb les dades obtingudes hem realitzat les següents gràfiques per poder obseravar millor les diferències:

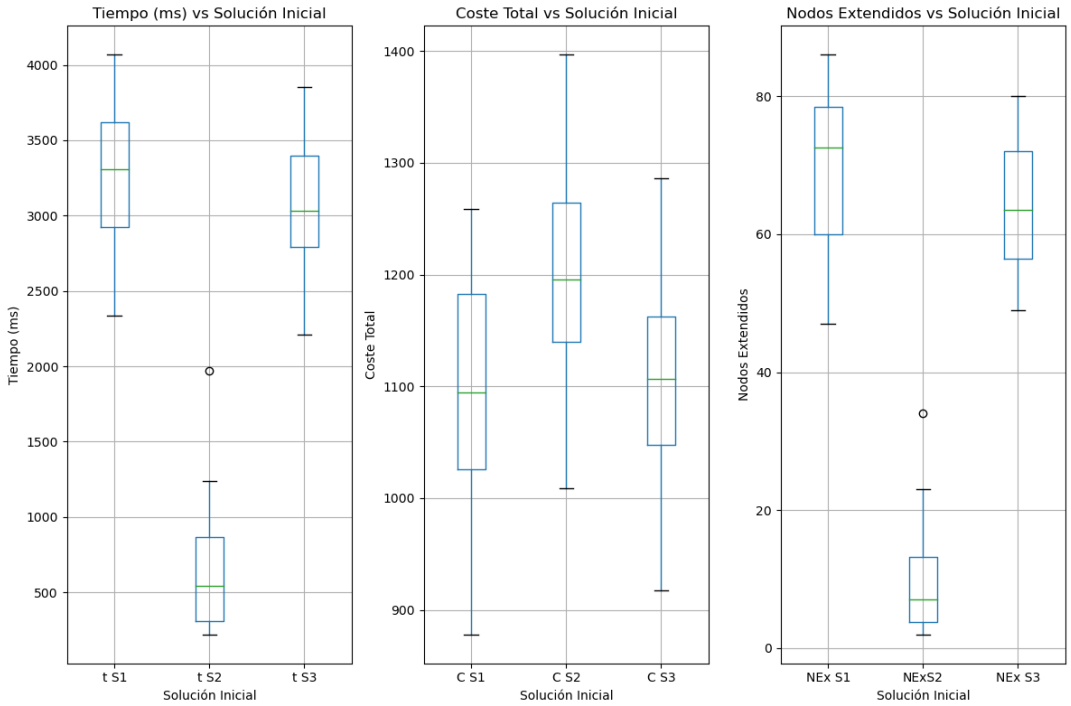

Figure 2: Resultats depenent de la solució Inicial utilitzada

Conclusions

A diferència del que havíem plantejat a la hipòtesis, la solució 3 no té un temps més elevat a la solució 1, és més triga menys sense perjudicar el cost total. Per aquesta raó per la resta d'experiments utilitzarem la Solució Inicial 3.

4.3 Experiment 3

En aquest experiment se'ns planteja buscar els valors de K i λ de Simmulated Annealing que generen millors resultats.
Observació: Diferents valors de K i λ donen diferents resultats.

Plantejament: Per fer aquest experiment hem decidit prova diferents valors de K i λ per trobar el que doni menys cost total.

Hipòtesis: Vam suposar que els valors que venen per defecte no serien els millors i que si variéssim els valors podríem trobar una combinació d'aquest 2 que donarien un resultat millor que el per defecte.

Metodologia

- Farem servir una quantitat de 100 paquets, les *seed* de paquet i la *seed* del transport seran variables escollides a l'atzar amb un número *random* entre 0 1000, la proporcio estar afixada a *1.2* i es farà sevir només l'operador de *mover-paquet* i l'heuristica que fa servir només el cost total.
- Usarem l'algorisme Simulated Annealing amb 10000 nombres de passos.
- Determinarem els possibles valors de K i λ que serà:
  - K = [1, 5, 25, 125, 625, 3125]
  - λ = [1.0, 0.1, 0.0001, $1.0 \times 10^{-5}, 1.0 \times 10^{-6}, 1.0 \times 10^{-7}$]
  - stiter = [10, 20, .., 90, 100]
- Executarem el programa 10 vegades per cada combinació de variables.
- Calcularem la mitjana de les 10 iteracions de cada combinació.
- Trobarem la mitjana màxima entre totes les combinacions de variables.

Resultats Hem elaborat un gràfic dels resultats amb *stiter* = 70, que conte el millor resultat.

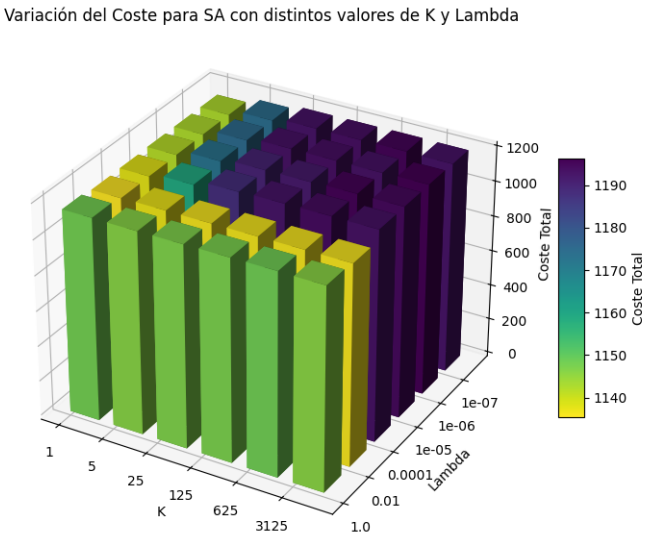

Figure 3: Variacio del Cost per SA amb diferents valors de K i λ

Com es pot observar en el gràfic anterior (Figue 3) podem veure que la millor combinació seria *stiter* = 70 i $\lambda = 0.01$, però costa diferenciar quina es la K mes favorable.

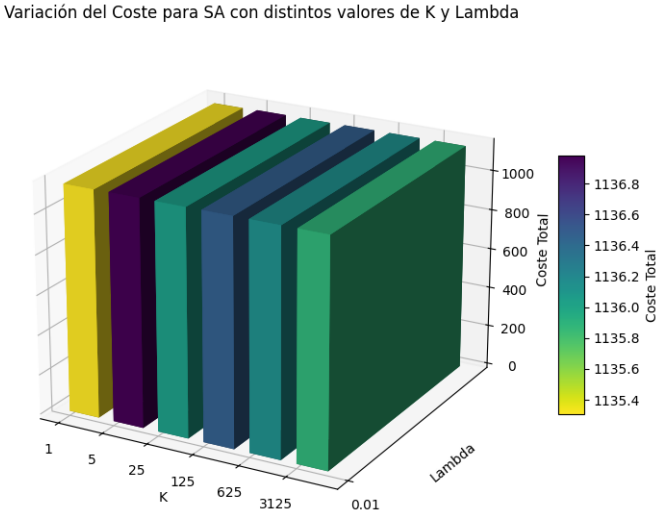

Figure 4: Variació del Cost per SA amb diferents valors de K i $\lambda = 0.01$

En aquest gràfic *(figure 4)* es pot observar millor quina seria la k per la millor configuració de Simulated Annealing.

Conclusions Per trobar el cost total mínim per transportar els paquets al seu destí amb Simulated annealing, haurem de fer servir *stiter* = 70, K = 1 i $\lambda = 0.01$. Ens semblen raonables ja que tenint en compte el funcionamnet de l'algorisme no son valors massa extrems.

4.4 Experiment 4

En aquest experiment, s'analitzarà l'evolució del temps d'execució de l'algoritme de Hill Climbing en funció de la proporció de pes transportable i el nombre de paquets. En la primera fase, es fixarà el nombre de paquets a 100 i es variarà la proporció de pes transportable, incrementant-la de 0,2 en 0,2, per observar com afecta a l'eficiència de l'algoritme.

A la segona fase, es mantindrà la proporció de pes a 1,2 i es farà variar el nombre de paquets a partir de 100, augmentant-lo de 50 en 50.

Amb aquestes proves, es buscarà comprendre com la complexitat del problema influeix en el rendiment de Hill Climbing, utilitzant la mateixa heurística per garantir la coherència dels resultats.

4.4.1 Experiment 4.1

En aquest experiment es busca analitzar la relació entre el temps d'execució de l'algorisme i la proporció de pes transportable disponible per a una distribució diària de paquets.
En un entorn logístic, com el de la companyia fictícia Ázamon, optimitzar l'assignació de paquets en funció de la capacitat de transport disponible pot ser determinant per reduir costos i millorar la satisfacció dels clients.
L'escenari proposat manté constant el nombre de paquets a 100, mentre que la proporció de pes transportable s'incrementa progressivament per observar com això afecta el temps de càlcul.
Aquest estudi permet entendre si el model implementat pot gestionar eficientment increments en les ofertes de transport, una situació comuna en operacions logístiques amb demanda variable.

Hipòtesis: Es va plantejar que el temps d'execució de l'algorisme creixeria de forma exponencial a mesura que augmentés la proporció de pes transportable, mantenint el nombre de paquets constant en 100. Aquesta hipòtesi es basava en la idea que una major proporció de pes incrementaria la complexitat dels càlculs de l'assignació de paquets a les ofertes de transport disponibles.

Metodologia:

- Fixarem la quantitat de paquets a 100
- Fixem l'heurística a la 1
- Fixem l'operador al 4 i la Solució Inicial a la 3 (com hem obtingut a l'experiment 1 i 2)
- Utilitzarem l'algorisme de *Hill Climbing*
- Utilitzarem 5 llavors aleatòries
- Executarem aquestes per cada valor de proporció
- Inicialitzant la proporció a 1.2, anirem augmentant el valor en 0.2 fins arribar a 4.0
- Mesurarem el temps d'execució

Resultats

Els resultats obtinguts es representen en una gràfica que mostra l'evolució del temps d'execució per cada increment de la proporció de pes transportable, des de 1,2 fins a 4,0. A mesura que la proporció de pes augmenta en intervals de 0,2, s'observa una clara tendència logarítmica en el temps d'execució, on el creixement es fa progressivament menys pronunciat amb cada increment de la proporció de pes.

En els primers increments (1,2, 1,4, 1,6), el temps d'execució augmenta de manera més perceptible, però aquesta variació es redueix a partir d'una proporció de pes transportable de 2,0. En aquest punt, el creixement del temps es va estabilitzant, fins i tot quan es dupliquen les capacitats de transport fins a 4,0. Aquest patró logarítmic es manté constant al llarg de l'experiment, suggerint que el model d'algorisme s'adapta eficaçment a increments elevats de la proporció de pes sense un cost computacional exponencial.

Aquests resultats són especialment visibles en la gràfica, que il·lustra com la pendent del temps d'execució disminueix progressivament, especialment a mesura que ens acostem als valors màxims de pes transportable. Així, la gràfica ens ajuda a visualitzar com el comportament logarítmic de l'algorisme permet mantenir el temps d'execució sota control fins i tot en escenaris amb capacitats de transport considerablement altes.

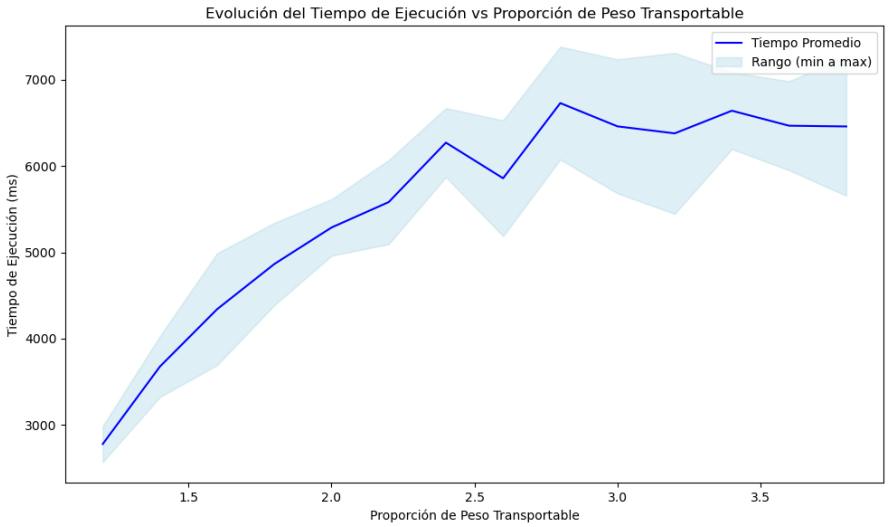

Figure 5: Temps d'execució depenent de la proporció

Conclusió

Aquesta observació suggereix que l'algorisme és més eficient en adaptar-se a increments en la proporció de pes transportable del que s'havia anticipat. El comportament logarítmic del temps d'execució implica que es pot incrementar la proporció de pes sense un augment significatiu del temps de càlcul, cosa que aporta robustesa a l'algorisme per a escenaris amb grans volums de pes total a transportar. Això podria ser un avantatge per aplicacions pràctiques on la capacitat de transport pot variar considerablement.

4.4.2 Experiment 4.2

En aquest experiment, es vol analitzar com influeix l'augment del nombre de paquets en el temps d'execució de l'algorisme de recerca local.
Aquest enfocament és rellevant per entendre la capacitat del model per gestionar grans volums de dades en un entorn logístic, com el de la companyia fictícia Ázamon. El nombre de paquets s'incrementa des de 100 fins a 500, per analitzar com el rendiment de l'algorisme varia quan s'incrementa la càrrega de treball.
Aquesta anàlisi ens permetrà avaluar la sostenibilitat de l'algorisme en escenaris de creixent demanda.

Hipòtesis: La hipòtesi inicial és que el temps d'execució de l'algorisme augmentarà de manera exponencial a mesura que es creixi el nombre de paquets. Aquest plantejament es basa en la idea que el nombre de combinacions i càlculs necessaris augmenta exponencialment amb l'increment de paquets, fent que la complexitat computacional es dispari en proporció.

Metodologia:

- Fixarem la proporció a 1.2
- Fixem l'heurística a la 1
- Fixem l'operador al 4 i la Solució Inicial a la 3 (com hem obtingut a l'experiment 1 i 2)
- Utilitzarem l'algorisme de *Hill Climbing*
- Utilitzarem 5 llavors aleatòries
- Executarem aquestes per cada nº de paquets
- Inicialitzant la proporció a 100, anirem augmentant el valor en 50 fins arribar a 500
- Mesurarem el temps d'execució

Resultats
Els resultats mostren una creixent i pronunciada pujada en el temps d'execució a mesura que s'incrementa el nombre de paquets de 100 a 500. Per a cada pas, el temps es va enregistrar amb els valors següents:

100	3135.0
150	17888.8
200	54710.8
250	164247.2
300	523602.8
350	961567.8
400	1355325.6
450	2483646.6
500	8054177.4

Table 7: Resultats per N de paquets

A la gràfica dels resultats, es pot observar que el temps d'execució creix de manera quasi-exponencial. Inicialment, per als primers valors de paquets (fins a 200), l'increment en el temps és perceptible però més moderat, amb un augment de l'ordre de desenes de mil·lisegons. Tanmateix, a partir dels 250 paquets, el temps d'execució comença a augmentar dràsticament. Aquesta tendència es manté a mesura que ens acostem als 500 paquets, on s'assoleix el temps

màxim de 8,054,177.4 ms, confirmant la hipòtesi d'un creixement exponencial.

Els increments entre cadascun dels valors són especialment rellevants per evidenciar aquesta tendència:

- Entre **300 i 400 paquets**, el temps gairebé es triplica, passant de 523,602.8 ms a 1,355,325.6 ms.
- De **400 a 500 paquets**, el temps d'execució es més que dobla, arribant al valor màxim observat de 8,054,177.4 ms.

Aquest temps tan elevat mostra les limitacions de l'algorisme en termes d'escalabilitat quan es gestionen volums elevats de dades. La gràfica dels resultats indica un creixement fortament no lineal, reforçant la hipòtesi de comportament exponencial amb un augment continuat del temps d'execució a mesura que es processa un major nombre de paquets.

Per il·lustrar de manera més senzilla les dades, s'ha representat en la següent gràfica:

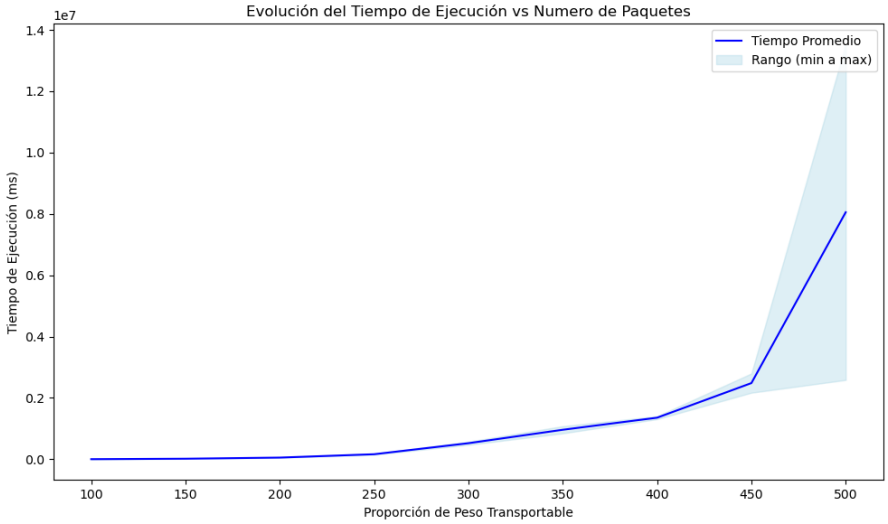

Figure 6: Temps d'execució depenent del numero de paquets

Conclusió: L'experiment ha confirmat la hipòtesi que l'increment en el nombre de paquets provoca un augment exponencial en el temps d'execució de l'algorisme, la qual cosa indica que aquest no és adequat per a escenaris amb grans volums de paquets sense optimitzacions addicionals. Donada la magnitud del temps d'execució, que arriba a 8.054.177,4 ms per a 500 paquets, es planteja repetir l'experiment utilitzant l'operador 3, amb l'objectiu d'explorar si aquest pot reduir significativament el temps de càlcul i, així, oferir una alternativa més eficient en escenaris de major demanda.

4.4.3 Experiment 4.2 amb Operador 3

Aquest experiment repeteix l'anàlisi de l'efecte del nombre de paquets en el temps d'execució, però ara utilitzant l'operador 3. En l'anterior execució, l'operador 4 va mostrar un increment exponencial molt elevat en el temps d'execució, amb valors insostenibles en escenaris amb una gran quantitat de paquets.
Amb l'ús de l'operador 3, esperem obtenir temps d'execució significativament més baixos, tot i mantenir una tendència exponencial, per tal de millorar l'eficiència del model i assegurar una major escalabilitat.

Hipòtesis: La hipòtesi manté que el temps d'execució continuarà amb un comportament exponencial, però s'espera que els valors siguin significativament menors que amb l'operador 4, degut a l'optimització en la generació d'estats que ofereix aquest operador.

Metodologia:

- Fixarem la proporció a 1.2
- Fixem l'heurística a la 1
- Fixem l'operador al 4 i la Solució Inicial a la 3 (com hem obtingut a l'experiment 1 i 2)
- Utilitzarem l'algorisme de *Hill Climbing*
- Utilitzarem 5 llavors aleatòries
- Executarem aquestes per cada nº de paquets
- Inicialitzant la proporció a 100, anirem augmentant el valor en 50 fins arribar a 750
- Mesurarem el temps d'execució

Resultats
Els resultats amb l'operador 3 van confirmar una reducció significativa del temps d'execució comparat amb l'operador 4. Tot i que el creixement exponencial persisteix, els temps obtinguts són molt menors, amb un valor màxim de 51,981 ms per a 750 paquets. A continuació, es mostren els valors més destacats:

100	168.4 ms
250	1,247.6 ms
500	11,418.6 ms
750	51,981.0 ms

Table 8: Resultats amb l'operador 3

La gràfica que representa aquests resultats il·lustra clarament la tendència exponencial, però també ressalta que els increments en temps són molt més graduals que amb l'operador anterior. Per exemple, per a 500 paquets, el temps d'execució amb l'operador 3 és 11,418.6 ms, en comparació amb els 8,054,177.4 ms obtinguts amb l'operador 4.

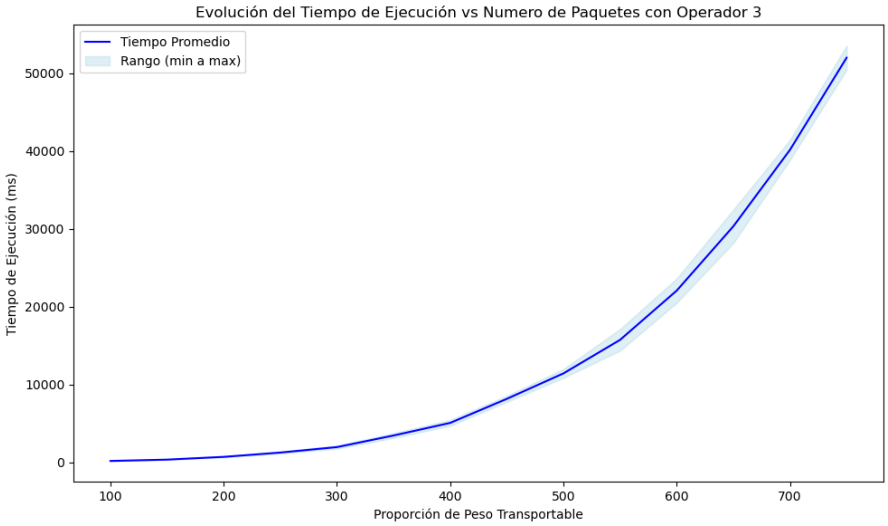

Figure 7: Temps d'execució depenent del numero de paquets (Operador 3)

Conclusió:
La gràfica que representa aquests resultats il·lustra clarament la tendència exponencial, però també ressalta que els increments en temps són molt més graduals que amb l'operador anterior. Per exemple, per a 500 paquets, el temps d'execució amb l'operador 3 és 11,418.6 ms, en comparació amb els 8,054,177.4 ms obtinguts amb l'operador 4.

4.5 Experiment 5

Analizando los resultados del experimento en el que se aumenta la proporción de las ofertas ¿cuál es el comportamiento del coste de transporte y almacenamiento? ¿Merece la pena ir aumentando el número de ofertas?

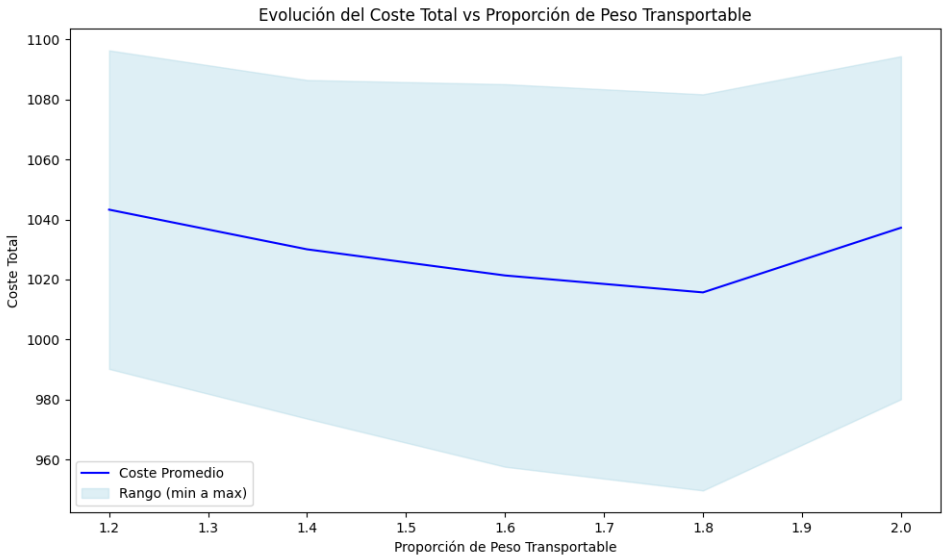

Figure 8: Evolució del cost total en funció de la proporció del pes transportable

Com es pot observar al gràfic, a mesura que s'augmenta la proporció de pes des de 1.2 fins a 2.0, el cost total dismunueix fins a un mínim que es troba a la proporció de pes de 1.8 i després torna a augmentar. Això suggereix que existeix un nivell òptim de proporció de pes transportable al voltant de 1.8, on els costos de transport són mínims.
El gràfic indica que augmentar la proporció del pes transportable (o les ofertes) inicialment redueix el cost total, però després d'un cert punt, el cost torna a augmentar. És a dir, incrementar la proporció de pes transportable fins a un punt és beneficiós en termes de cost total, peró si es continua augmentat més enllà del punt mínim segurament resultarà en un increment dels costos totals.

4.6 Experiment 6

El experiment 6 tracta sobre l'impacte de la felicitat al cost total del transport quan ho tenim en consideració en l'heuristica utilitzant l'algorisme de Hill Climbing.

Hipòtesi:
Com a més diferència hi hagui en la relació entre la valoració del cost i de la felicitat pitjor serà el cost.

Metodologia:
Per a implementar la felicitat en l'experiment hem decidit restar el valor de felicitat sobre el cost, ja que qualsevol cost extra causat per l'increment de la felicitat es negligible si la felicitat aconseguida es suficient gran per a contrarestar-ho.
S'aplica aquesta resta en la funció heurística que calcula la correctesa d'una possible solució emprant la següent formula: a*cost - b*felicitat. Les variables 'a' i 'b' serveixen per alterar el nivell d'importància que es dona a la relació entre el cost i la felicitat. En aquest experiment

ens interessa estudiar els canvis en cost de transport i emmagatzematge i el temps de càlcul variant els valors de les variables 'a' i 'b'.
Durant l'experiment fixarem els següent valors per estudiar les variacions causades amb el canvi de la 'b':

- Seed paquets: 1234
- Seed transport: 1234
- Numero de paquets: 100
- Proporció: 1.2
- Generadora de la solució inicial: 1
- Algorisme: Hill Climing
- Heurística: a*cost - b*felicitat
- a: 1.0
- Operador: [1,2,3,4]

Aquest experiment s'executa amb cada operador per veure com canvien els resultats depenent de la 'b' que s'usa. La cerca s'executa un cop per cada valor de 'b' i aquest per cada operador. Els valors de 'b' són [0.25,0.33,0.5,1.0,2.0,3.0,4.0]. S'han triat aquests per veure com són els resultats quan el cost té més importància en l'Heurística, quan els dos valors són iguals i quan la felicitat té més importància. Multiplicant la felicitat per valors menor que 1 ens permet simular augmentar la importància del cost sense haver de modificar la 'a'.

Resultats:
Resultats amb l'operador 1:

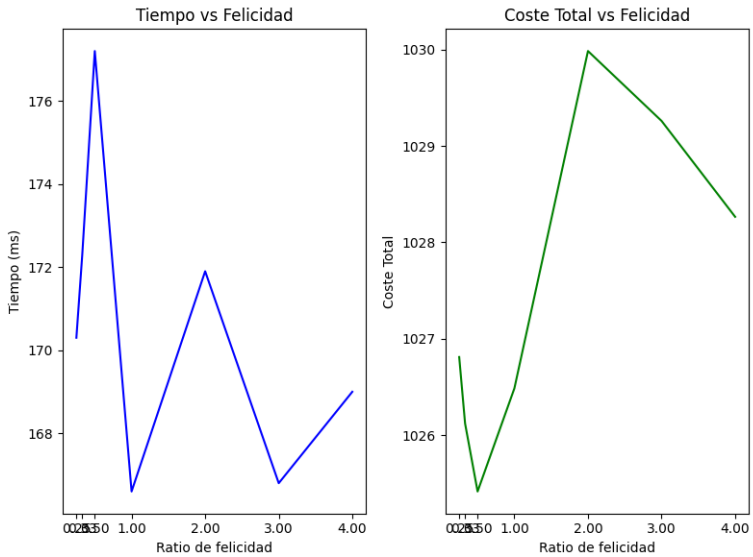

Figure 9: Evolución del tiempo y el coste total a partir de el cambio del peso de la felicidad aplicando el operador 1 con Hill Climbing.

Resultats amb l'operador 2:

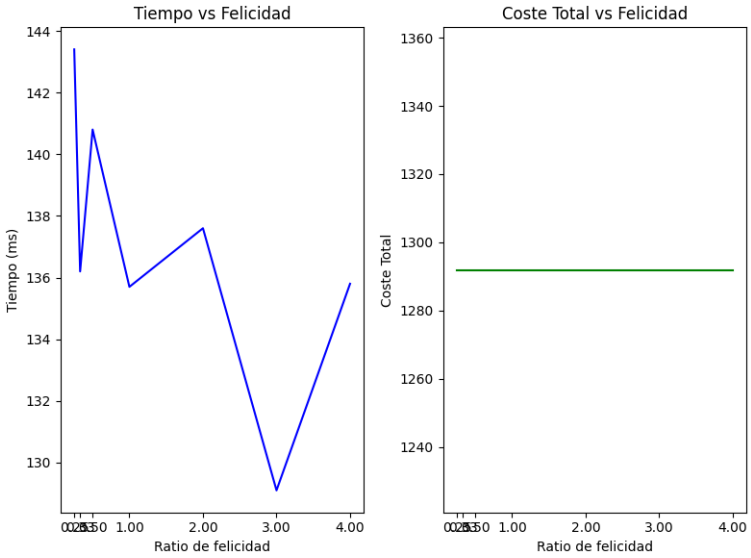

Figure 10: Evolución del tiempo y el coste total a partir de el cambio del peso de la felicidad aplicando el operador 2 con Hill Climbing.

Resultats amb l'operador 3:

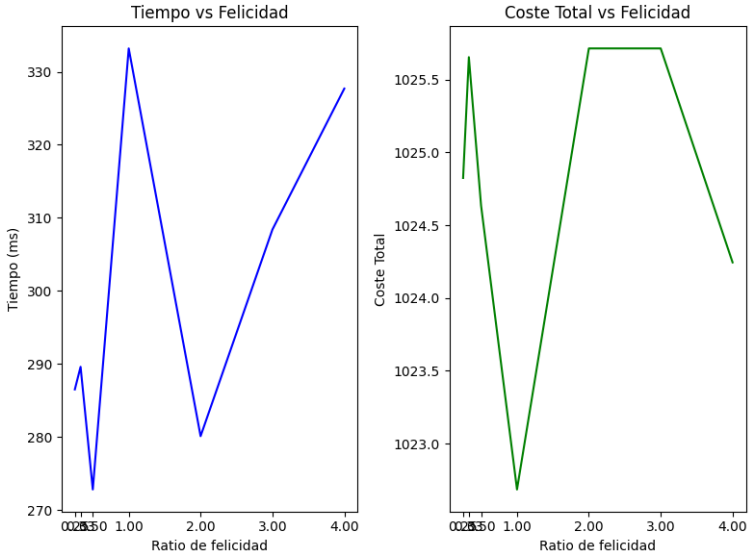

Figure 11: Evolución del tiempo y el coste total a partir de el cambio del peso de la felicidad aplicando el operador 3 con Hill Climbing.

Resultats amb l'operador 4:

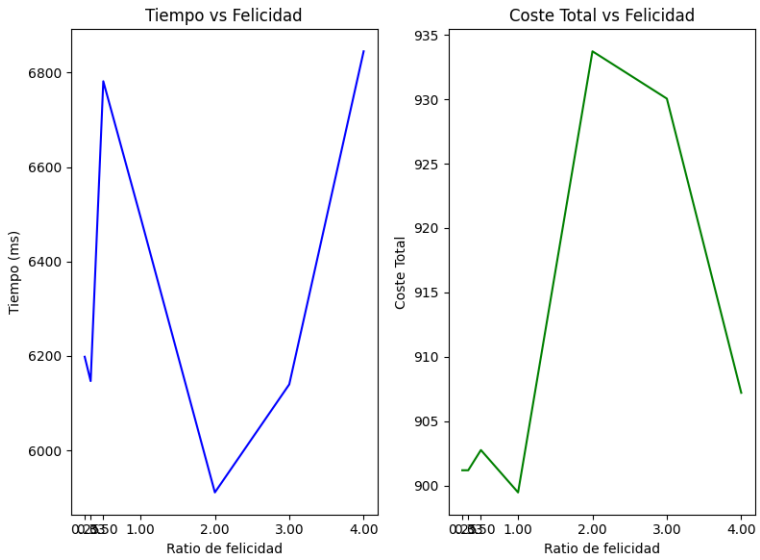

Figure 12: Evolución del tiempo y el coste total a partir de el cambio del peso de la felicidad aplicando el operador 4 con Hill Climbing.

Conclusions: Mirant els gràfics podem notar que el valor de relació de la felicitat causa grans canvis al cost temporal depenent de l'operació aplicada. En la primera operació les millors ponderacions són al punt 1.0 i al 3.0. En el dos sol es al punt 3.0. Al tercer els punts més eficients són al 0.5 i al 2.0. Mentre que a l'operació quatre el temps més eficient es quan la felicitat té el doble de pes que el cost. Es pot intuir d'aquests gràfics que no hi ha molta relació entre el valor de relació i el temps que tarda en ser calculat ja que els resultats varien molt per cada operació.
En quant a cost si que tenen major semblança. Al primer i quart gràfic els millors costos venen quan el valor de relació es igual o menor a 1.0 Al tercer no hi ha cap valor recomanable que no sigui una relació 1.0 ja que els resultats amb tots els altres valors surten bastant disparats. En cap dels tres gràfics mencionats es recomanable utilitzar un valor més gran que 1.0. Es pot deduir que el valor de relació afecta d'alguna manera al càlcul ja que aquests tres gràfics contenen una seqüència similar de dades. El gràfic dos no conté cap variació en el cost resultant, indicant que no hi afecta el valor de la felicitat al càlcul.
No hi ha cap relació enter el temps de càlcul i el cost final. Que el cost final sigui major o menor no te importància al temps gastat per calcular-ho.

4.7 Experimento 7

Al experiment 7 volem veure els efectes d'aplicar l'heurística 2 emprant l'algorisme de Simulated Annealing. Com afecta el cost quan tenim en compte la felicitat al fer la cerca.
Hipòtesi:
Com a més diferència hi hagui en la relació entre la valoració del cost i de la felicitat pitjor serà el cost.

Mètode: L'experiment ens demana el mateix que l'apartat anterior però aplicant l'algorisme de Simulated Annealing. Els valors triats per aquest experiment són els seleccionats com a més eficients en l'experiment 3.

- Seed paquets: 1234

- Seed transport: 1234
- Numero de paquetes: 100
- Proporcion: 1.2
- Generadora de la solució inicial: 1
- Algorisme: Simulated Annealing
- Steps: 1000
- Stiter: 70
- K: 1
- Lambda: 0.01
- Heuristica: a*cost - b*felicitat
- a: 1.0
- operador: [1,2,3,4]

Resultats:
Resultats amb l'operador 1:

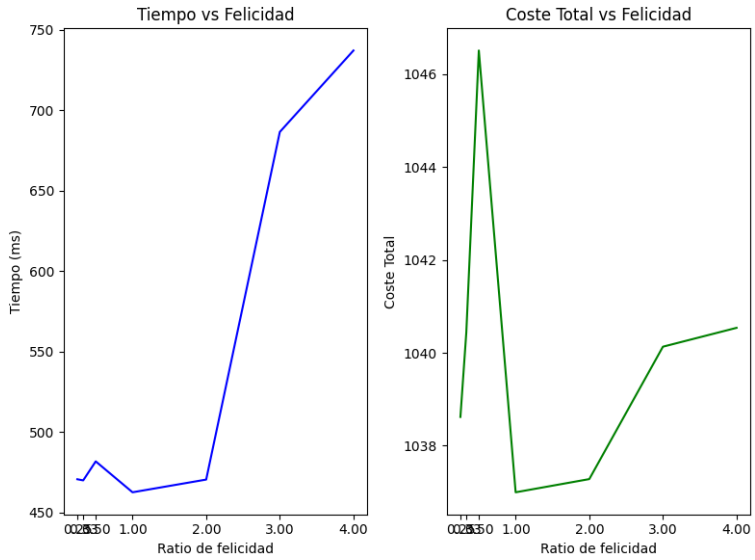

Figure 13: Evolución del tiempo y el coste total a partir de el cambio del peso de la felicidad aplicando el operador 1 con Simulated Annealing.

Resultats amb l'operador 2:

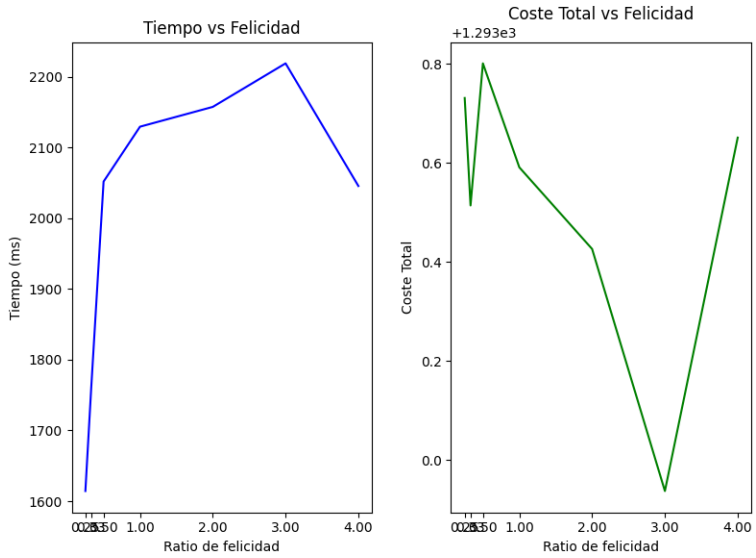

Figure 14: Evolución del tiempo y el coste total a partir de el cambio del peso de la felicidad aplicando el operador 2 con Simulated Annealing.

Resultats amb l'operador 3:

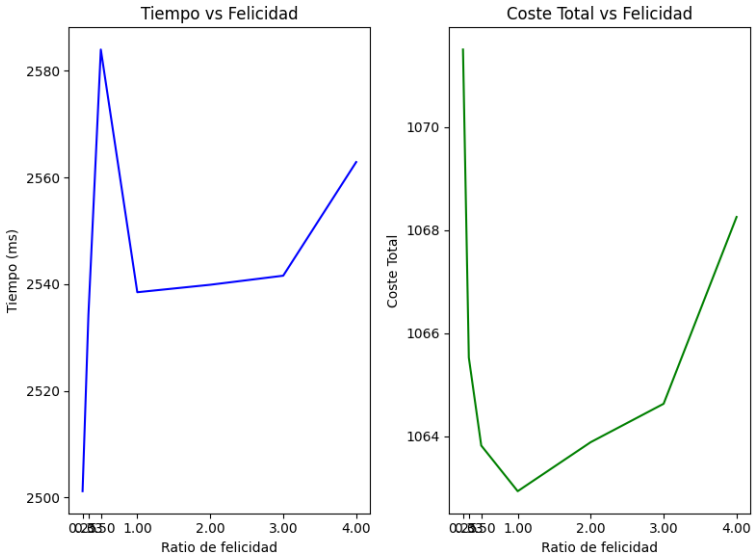

Figure 15: Evolución del tiempo y el coste total a partir de el cambio del peso de la felicidad aplicando el operador 3 con Simulated Annealing.

Resultats amb l'operador 4:

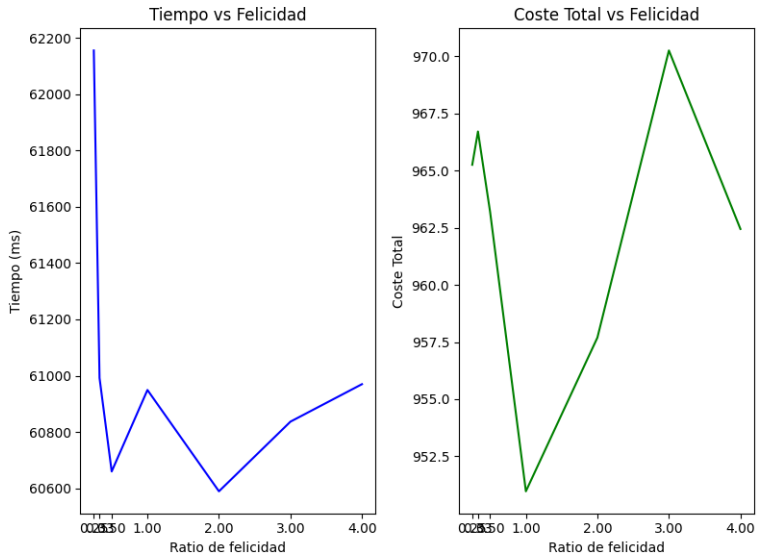

Figure 16: Evolución del tiempo y el coste total a partir de el cambio del peso de la felicidad aplicando el operador 4 con Simulated Annealing.

Conclusions:
Similar als gràfics de Hill Climbing el temps utilitzat en cada cerca no segueix cap tendència. El valor de relació no té cap efecte en el temps emprat en calcular el resultat. Els temps varien molt entre operacions.
A diferència del Hill Climbing en Simulated Annealing tampoc hi ha cap similitud en el cost calculat entre operacions. Cap gràfic segueix una tendència similar a un altre. L'única similitud que tenen entre si es que els valors que retornen el menor pes són entre el 1.0 i el 3.0.
Tampoc hi ha cap relació entre el temps gastat i el resultat obtingut.

4.8 Experimento 8

Hemos asumido un coste de almacenamiento fijo diario de 0,25 euros por kilo. Sin hacer ningún experimento ¿como cambiarían las soluciones si variamos este precio al alza o a la baja?

Si el cost d'emmagatzematge augmenta, la solució òptima tendirà a minimitzar el temps que els paquets estàn al magatzem, ja que cada dia addicional d'emmagatzematge representa un increment directe en el cost total. En aquest context, s'ha d'agilitzar el moviment de paquets per reduir els temps d'espera encara que això impliqui assumir un major cost al transport, ja que la despesa que s'evita en l'emmagatzematge pot compensar els costos addicionals del transport.
D'altra banda, si el cost d'emmagatzematge dismunueix, les solucions òptimes consistiran en reduir el cost de transport. En aquest cas, serà possible considerar ofertes de transport més econòmiques, encara que això impliqui temps d'emmagatzematge més llargs.
En resum, si l'emmagatzematge és econòmic, s'han de prioritzar les solucions que minimitzin el cost de transport i, en cas que l'emmagatzematge sigui més car, s'han de prioritzar solucions que minimitzin el temps d'emmagatzematge.

5 Conclusions

Gràcies a aquest treball hem après sobre el funcionament dels algoritmes Hill Climbing (HC) i Simulated Annealing (SA). També ens ha permès analitzar les diferències de rendiment entre *HC* i *SA*.

Pel que fa a l'eficàcia dels algorismes, els resultats dels experiments mostren que tots 2 algorismes són força eficients per resoldre el problema, però cadascun té avantatges i limitacions segons els paràmetres i els operadors utilitzats. Per exemple, Hill Cimbing demostra ser més rapid en escenaris amb menys complexitat, mentre que Simulatted annealing és més eficaç per evitar mínims locals en casos més complexos.

Si comparem els operadors podem dir que l'operador 4 tendeix a obtenir el millor cost però a canvi de temps d'execució. Això en suggereix que l'ús de més operadors proporciona solucions més bones.

També hem pogut comparar l'impacte amb diferents heurístiques, que evidencia que l'heurística ha de ser adaptada segons els objectius del problema.

A més si ens fixem en l'escalabilitat, podem dir que té un comportament quasi exponencial del temps d'execució amb un augment de la quantitat de paquets a entregar. Això faria considerar alternatives més escalables o versions més optimitzades dels algorismes utilitzats.

Per acabar aquesta pràctica ens ha ajudat a millor les competències en llenguatges de programació com Java, Python o eines de treball col·laboratiu com GitHub o LATEX. També em desenvolupat habilitats per la planificació i execució i redacció d'experiments.