Pràctica de Cerca Local

Projecte d'Intel·ligència Artificial

Grup 13

Joan Casahuga Altimiras

Miquel Torner Viñals

Xavier Bernat López

Departament de Ciències de la Computació

Universitat Politècnica de Catalunya

Octubre, 2022

Índex

1. Descripció del problema

1.1 Introducció

L'objectiu d'aquesta pràctica és resoldre el problema d'abastiment d'energia utilitzant algorismes de cerca local.
En aquesta pràctica, se'ns planteja una situació en què hem de gestionar una empresa de generació d'electricitat per poder abastir a tothom qui ho necessiti maximitzant el benefici, doncs pot donar-se que tenir centrals obertes generi més pèrdues que beneficis, ergo no surti a compte mantenir-les obertes.

1.2 Anàlisi del problema

El problema planteja una situació en la qual s'ha de gestionar la xarxa elèctrica assignant els clients que necessitin subministrament a les diverses centrals repartides pel territori tenint en compte costos i beneficis per tal d'arribar a una solució el més bona possible.
A l'hora d'analitzar el problema, el primer que s'ha de veure és quina estratègia aplicar, ja que n'hi han de molt diverses. El primer que es pot descartar és intentar fer un algorisme únic per resoldre'l, ja que la complexitat d'un problema amb tals característiques faria que el cost temporal del algorisme necessari per resoldre massa elevat. Per tant, s'ha d'intentar aplicar un mètode d'Intel·ligència Artificial, que tot i no tenir un cost temporal perfecte, serà millor que l'altre alternativa. En aquest cas s'ha optat per emprar la Cerca Local, perquè en aquest cas no es busca el com arribar a una solució "correcta" en concret, sinó que el que es vol és arribar a una solució el millor possible (ja que no se sap quina és aquella solució "correcta"), sense que sigui rellevant el procés o les passes a seguir per arribar a aquest estat.
El primer que s'ha de tenir en compte és que hi ha tres tipus de centrals segons l'interval de producció mínim i màxim, i que cada central té un cost tant per estar produint com per estar parada. El tipus de central s'ha de tenir en compte per diversos motius: determina la capacitat de la central d'abastir a més o a menys gent i els tres tipus tenen costos tant de producció com de parada diferents.
Cal també tenir en compte que els clients tenen diferents paràmetres que afecten a al seu consum i quant paguen per cada MW. En primer lloc, hi ha clients que són garantitzats i d'altres que no, per tant, per arribar a una solució vàlida, caldrà que tots els clients garantitzats estiguin associats a una central no aturada, mentres que els no garantitzats podran estar sense associar, tot i que no associar-ne un implicarà haver de pagar-li una

indemnització, per tant, s'ha de mirar si surt més a compte o no associar-lo a una central, això dependrà alhora dels MW que demani i de l'energia que té una potencial central que pot ser assignada al client. Un dels altre factors a tenir en compte és que els clients no garantitzats paguen menys que els garantitzats.
A més a més també és necessari considerar dels clients és que es divideixen en 3 tipus: G, MG i XG (grans, molt grans i extra grans) i que cada tipus té un consum en un rang definit amb un preu establert per a cada MW, a part la seva indemnització també és diferent.
Finalment cal pensar també a l'hora d'assignar clients a les centrals en la distància, ja que una part de l'electricitat generada es perd pel camí, però al client cal subministrar-li tota la que demani i pagarà només per la consumeixi, per tant, s'haurà de produir més energia per tal de poder cobrir la demanda del client.

Un cop vistos tots els factors a tenir en compte, es pot definir el conjunt de condicions que ha de satisfer tota solució, per a aquest problema, per a considerar-se vàlida:

- $\forall i \in [0, nCentrals), ProduccióTotalCentral_i \geq \sum_{k=0}^{n} DemandaClient_k + \frac{1}{100-\%_{pèrdua}} * d_{Client\ k,\ central\ i}$

- $\forall i \in [0, nClients), NombreAssignacionsClient_i \leq 1$

- Tots els clients garantitzats han d'estar assignats a una central.

- $\forall i \in [0, nClients), DemandaClient_i \leq ElectricitatSubministrada$

- $CostTotal < DinersRecaptats$ (S'ha d'obtenir benefici).

Finalment, abans d'implementar res, cal especificar quin criteri es farà servir per determinar si una solució és més propera o no a la òptima, i aquest criteri serà el benefici. L'objectiu, per tant, d'aquesta cerca local serà maximitzar el benefici tenint en compte les restriccions anteriorment esmentades.

2. Adaptació del problema a la cerca local

2.1 Representació de l'estat del problema

2.1.1 Implementació

En el moment de decidir com representar l'estat del problema, és a dir, com parametritzar la solució, és necessari identificar què és rellevant en la solució per tal de optimitzar el cost temporal dels mètodes que s'utilitzen a l'heurística i així disminuir el temps d'execució (evitant càlculs innecessaris).
Per tant hem decidit implementar l'estat amb els següents elements:

- Clientes clients

Representa una *ArrayList* de clients. L'únic mètode que ofereix la classe, a part de les que extend de *ArrayList* és la constructora, la qual genera un determinat nombre de clients, tenint en compte el percentatge de clients de cada tipus (G, MG, XG) i la proporció de clients garantits, en posicions aleatòries segons una *seed* que també se li passa per l'argument.
Aquest objecte ens permetrà saber on estan ubicats els clients per tal de poder generar la solució inicial utilitzant les coordenades de cada client, el seu consum i si són garantits o no.

- Centrales centrals

Representa una *ArrayList* de central. L'únic mètode que ofereix la classe, a part de les que extend de *ArrayList* és la constructora, la qual genera una llista de centrals donat un vector amb el nombre de centrals de cada tipus que es volen en posicions aleatòries fent servir una *seed* que també es passa com a argument

- int[] assignedClients

Representa una *Array* la qual té la mida del nombre de clients i guarda a cada posició un valor per al client *i,* el qual té el mateix índex que en l'element Clientes clients, l'índex de la central a la que ha estat assignat. Aquesta estructura ens permetrà fer un seguiment de quin client està assignat a quina central per saber si es pot aplicar un dels operadors, en cas que no estigui assignat a cap central s'utilitzarà el valor *-1* per indicar-ho.

- double[] leftPowerCentral

Representa una *Array* la qual té la mida del nombre de centrals i guarda a cada posició un valor per la central *i,* la qual té el mateix índex que en l'element Centrales centrals, el nombre de MW que pot arribar a produir tenint en compte els clients ja assignats a la central. Aquesta estructura ens permet millorar l'eficiència del programa, doncs no és necessari sumar tots els consums del clients d'una central cada vegada que es vol aplicar un operador si la central per saber si es pot aplicar un operador.

- double benefDynamic

Aquest valor s'utilitza per guardar el benefici generat per l'estat actual, peça clau a l'heurística doncs obtenir una bona solució implica maximitzar el benefici, i aquest variable ens permet saber-lo en tot moment. El valor que conté s'actualitza cada vegada que s'aplica una operació de forma que no és necessari recalcular el benefici total, fet que permet millorar el cost temporal de programa.

- double guaranteedNotAssigned

Aquest valor s'utilitza per guardar el número de clients garantits no assignats de l'estat actual, cosa que ens permet saber en tot moment com de "lluny" estem de trobar una les solució vàlida al problema. El valor que conté s'actualitza cada vegada que s'aplica una operació de forma que no és necessari recalcular el número de garantits no assignats, millorarant l'eficiència temporal del programa.

- double distanceDynamic

Aquest valor s'utilitza per guardar la distància mitjana en l'estat actual, cosa que ens permet saber en tot moment com de lluny estan les centrals assignades als clients, fet important doncs com més gran sigui aquest valor voldrà dir que més energia està sent perduda pel camí. El valor que conté s'actualitza cada vegada que s'aplica una operació de forma que no és necessari recalcular la distància mitjana total, cosa que permet millorar l'eficiència temporal del programa.

- double powerLeftDynamic

Aquest valor s'utilitza per guardar l'energia en MW disponible a la xarxa que està disponible per ser utilitzada donat l'estat actual. El valor que conté s'actualitza cada vegada que s'aplica una operació de forma que no és necessari recalcular la variable a cada iteració, fet que permet millorar l'eficiència temporal de programa.

2.1.2 Anàlisi del espai de cerca

L'espai de cerca està format per totes les possibles solucions que hi ha per un problema donat una representació de l'estat del problema. Més concretament, en el nostre cas l'espai de cerca ve determinat per el nombre d'assignacions possibles de clients a centrals sempre que la suma de clients assignats a una central no tinguin superior a la producció que ofereix aquesta. Per tant en el nostre cas, donat un nombre de clients C i un nombre de centrals P, la mida de l'espai de cerca tindria una mida màxima de P^C suposant que totes les solucions fossin vàlides, doncs aquest nombre és el total de possibles combinacions.

2.2 Estratègies de generació de la solució inicial

L'estat obtingut per la generació inicial és el punt de partida a partir del qual aplicarem els diferent operador per poder millorar la nostra solució. És per això que en base a l'experimentació és necessari trobar un mètode que generi una bona solució inicial, doncs aquest efecte directament a la solució final que obtindrem. Així doncs, les funcions de generació inicial que hem implementat són les següents:

- `generateInitialSolutionClosestFull()`

Aquest mètode genera una solució on assgina el client a la central que estigui més a prop i amb espai necessari per a aquest, començant pels garantits per assegurar-nos que la solució inicial serà vàlida.
A simple vista, aquesta generació seria la que possiblement obtindrà un valor de partida més bo, ja que intenta maximitzar l'ocupació de les centrals i minimitzar l'energia perduda pel transport.

- `generateInitialSolutionRandomGuaranteed(int k)`

Aquest mètode genera una solució on assgina a tot client garantit a una central aleatoria i amb espai necessari per aquest.
L'objectiu d'aquesta generació és evitar que la solució màxima es pugui trobar lluny de minimitzar la distància amb l'afegit de que encara es poden afegir de forma més senzilla els clients no garantits.

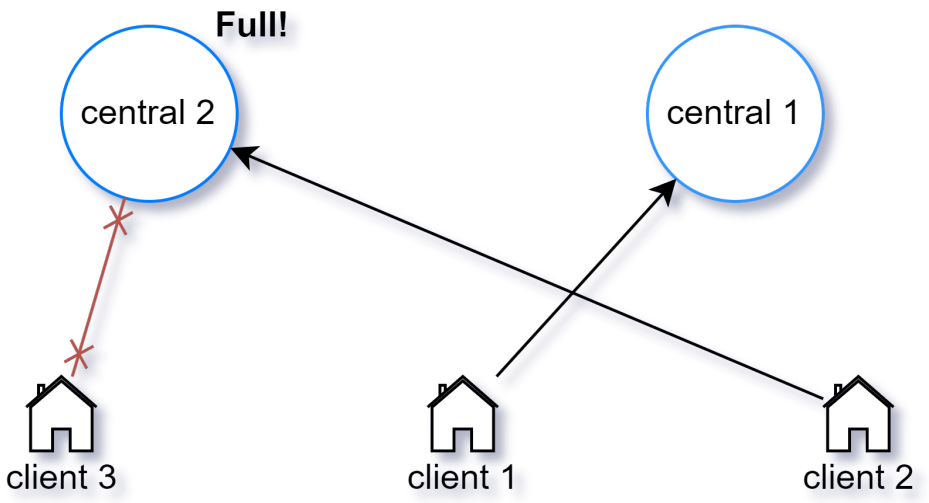
- `generateInitialSolutionRandom(int k)`

Aquest mètode genera una solució on assgina a tot client a una central aleatoria i amb espai necessari per a aquest.

L'objectiu és el mateix que en l'anterior plantejament, però també fer-ho de forma aleatòria per als no garantits.

Aquests dos últims mètodes obtenen solucions no vàlides de forma més freqüent, per evitar-ho, el que farem serà augmentar un factor k, aquest serà 1 a l'inici, indicant que només seleccionarem una central aleatoria i l'assignarem al client, quan k és dos seleccionarem dos centrals de forma aleatoria i assignarem el client a la més propera, aquest factor k anirà augmentant fins que tots els clients garantits tinguin un central assignada.

Per altre banda, fins i tot la generació inicial *`generateInitialSolutionClosestFull()`* pot donar solucions no vàlides, ho veiem amb un exemple:



Assignem el client 1 a la central més pròxima, que és la central 1, aquesta queda plena, assignem el client 2 a la central 2, aquesta queda plena degut a la distància i el client 3 es queda sense assignar, si els 3 clients fossin garantits la solució seria invalida degut al plantejament de la generació inicial, en totes les cinc que hem proposat, en canvi si assignessin el client 2 primer, el client 1 i 3 podrien estar assignats a la central 2(que és més gran que la central 1) i obtenir un estat valid.

2.3 Operadors

Els operadors que seleccionem ens permetran explorar l'espai de solucions. Tenint en compte que volem intentar arribar a les màximes combinacions d'assignacions possibles per tal d'explorar de forma més exhaustiva l'espai de cerca i així millorar el resultat final . És per això que ens hem decantat pels operadors següents: `moveClient` (la qual pot assignar a un client a una central que encara hi queda energia i alhora també pot provocar que es canviï un client a una central més propera i per tant es malgasti menys energia, i com a conseqüència quedi un lloc lliure per assignar-hi un altre client a la central alliberada en el cas d'emprar Simulated Annealing que en cercar en profunditat prodria trobar millors estats que utilitzant el següent operador), `swapClient` (la qual permet intercanviar les central de dos clients i evitar malgastar energia en alguns estats en què Hill Climbing no seria capaç de trobar-ho fent servir l'operació anterior) i `resetCentral` (la qual elimina totes les assignacions dels clients d'una central, d'aquesta forma podem saber si surt a compte tancar una central quan emprem Hill Climbing, doncs no es podria donar que amb més d'una assignació a una central l'algorisme trobes una millora amb l'heurística utilitzada que veurem més endavant) .

2.3.1 Move

`mouClient(int client, int central)`

- Efecte: Assigna un client *c* a una central *p*.
- Condicions aplicabilitat: S'ha de satisfer que l'energia que consumirà el client *c* pot ser produïda per la central *p.*
- Factor de ramificació: Donat un nombre de clients N de clients i un nombre M, el factor de ramificació serà NxM.
- Retorna: Si es donen les condicions d'aplicabilitat de l'operació, el resultat que retornarà aquest serà l'assignació del client *c* a la central *p.*

2.3.2 Swap

`swapClient(int client1, int client2)`

- Efecte: Assigna al client *c1*, assignat a la central *p1*, a la central *p2,* assignada al client *c2,* i li assigna al client *c2* la central *p1*.
- Condicions aplicabilitat: S'ha de satisfer que l'energia no utilitzada de *p1* sumada a l'energia que deixarà de consumir *c1* serà superior a l'energia que necessitarà *c2* per part de *p1*.Tantmateix també s'haurà de satisfer que l'energia no utilitzada de *p2*

sumada a l'energia que deixarà de consumir *c2* serà superior a l'energia que necessitarà *c1* per part de *p2.*

- Factor de ramificació: Donat un nombre de clients N, el factor de ramificació serà (NxN-1) / 2.
- Retorna: Si es donen les condicions d'aplicabilitat de l'operació, el resultat que retornarà aquest serà un intercanvi en les assignacions de centrals entre el clients *c1 i c2*.

2.3.3 Reset Central

`resetCentral(int central)`

- Efecte: Dessassigna tots els clients d'una central *p*.
- Condicions aplicabilitat: No hi ha clients garantits assignats a la central *p*.
- Factor de ramificació: Sent M el nombre de centrals, el factor de ramificació és M.
- Retorna: Si es donen les condicions d'aplicabilitat de l'operació, el resultat que retornarà aquest serà un estat en què tots els clients de la central *p* no estaran assignats a cap central.

2.4 Funcions heurístiques

2.4.1 Benefici

Donat que ens aquest problema, com s'ha vist abans, el que busquem és incrementar el benefici, és necessari trobar una heurística que ens maximitzi aquest valor. Així doncs, tot i que hi ha altres factors, el nostre objectiu final és obtenir l'estat amb millor benefici alhora que garantim la satisfacció de totes les restriccions imposades. És per això que hem decidit que la nostre primera heurística sigui el mateix benefici, ja que podem estar segurs que cada vegada que avanci a un nou estat aquest serà necessàriament millor que l'anterior doncs el nostre objectiu i heurística són la mateixa funció.

2.4.2 Distància energia

A partir de múltiples execucions del problema utilitzant l'heurística del benefici hem observat com evita crear espais extra en les centrals i reduir distàncies per intentar encabir més clients, que acostuma a resultar en un benefici, per tant ens vam decidir a provar la minimitzar tan distàncies com l'espai no utilitzat en les centrals, per tant pel càlcul utilitzarem la fórmula:

$$h = distanceDynamic^2 - powerLeftDynamic^2$$

El motiu d'elevar les dos variables al quadrat és evitar suavitzar el heurístic i no quedar-nos en una meseta, com passava amb la heurística del benefici.

2.5 Funcions successores

2.5.1 Hill Climbing

L'algorisme de Hill Climbing aplica tots els operadors possibles a l'estat actual, quedant-se amb el que tingui l'heurística més baixa, en aquest cas. L'inconvenient principal d'aquest algorisme és arribar a mínims locals i, per tant, no ens garantitza que la solució sigui immillorable.
A més, és poc eficient a nivell espacial i temporal en funció de l'heurística i depenent de com aquesta estigui implementada.

2.5.2 Simulated Annealing

En el cas de l'algorisme Simulated Annealing, el que es fa és explorar successor, de forma aleatòria. Això fa que, per un costat l'estat pugui tenir una heurística pitjor, però per l'altre, evita caure en mínims locals. A més, si la distribució probabilística que decideix quins estats s'exploren es fa bé, s'hauria d'arribar al mínim absolut.

2.6 Funció d'estat objectiu

Donat que estem resolent un problema emprant la cerca local, no és possible saber si l'estat en que ens trobem serà la solució la millor, és la millor que la classe objectiu sempre retornarà fals.

3. Experimentació

3.1 Experiment 1: Influència dels operadors

- Observació

Podem obtenir millors resultats segons els operadors que apliquem.

- Plantejament

Escollim diferents combinacions d'operadors i observem les seves solucions.

- Hipòtesis

Totes les combinacions d'operacions són iguals (H0) o hi ha combinacions millors que altres (H1).

- Mètode

- Farem l'experiment 3 vegades amb 3 *seeds* diferents.
- Farem 5 execucions de l'experiment per a cada *seed* per a la inicialització més pròxim i omplert per a cadascuna de les combinacions d'operacions.
- Experimentarem amb estats de 40 centrals i 1000 clients.
- Utilitzarem l'algorisme de Hill Climbing amb l'heurística del benefici.
- Mesurarem els diferents paràmetres per fer la comparació.

A continuació es mostra un gràfic amb els resultats de l'experiment anterior, les dades del qual podem trobar a la taula situada al annex 5.1.

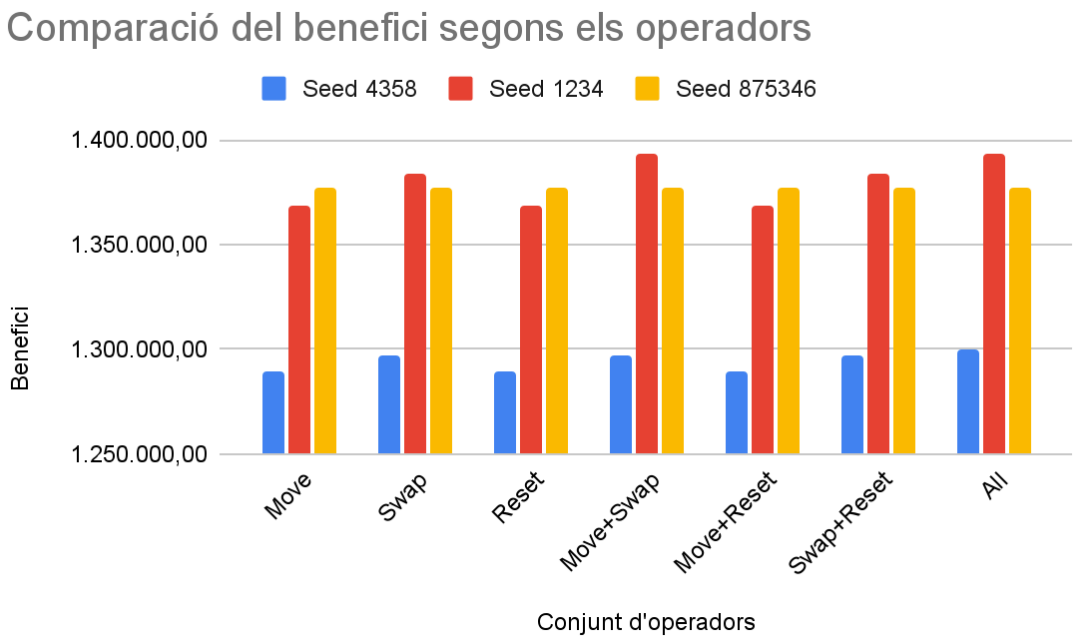


Figura 3.1.1. Comparació de la mitjana de beneficis segons el conjunt d'operadors.

Si observem el gràfic anterior podem veure que segons les combinacions d'operacions disponibles s'obté més o menys benefici (hem de tenir en compte que en tots els casos s'obté un mínim el qual és donat per la solució inicial) no obstant el conjunt d'operacions que ens permet millorar el benefici total acaba essent totes les possibles, ja que al emprar l'algorisme de Hill Climbing, el fet de tenir més ramificació per cada estat fa que sigui possible trobar millors estats pels que continuar en general, tot i que es poden donar casos en què això no succeeixi.

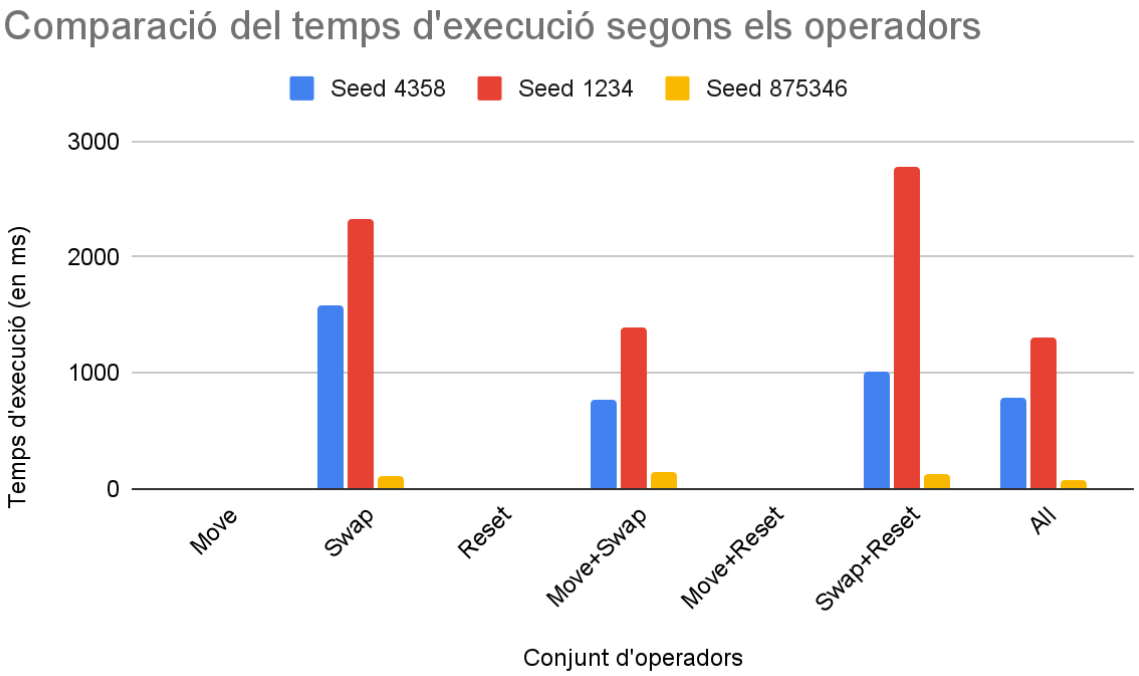


Figura 3.1.2. Comparació dels temps d'execució segons el conjunt d'operadors.

D'altra banda, poden observar el gràfic de la figura 3.1.2, el qual ens mostra la mitjana dels temps d'execució per a cadascuna de les combinacions d'operadors. Notem que la que ens donava millors resultats (All) no és la que té un cost temporal més elevat, en canvi, les combinacions *Swap* i *Swap+Reset* destaquen per sobre les altres. Això és degut a que el *Reset* gairebé no s'utilitza (ja que en la configuració hi ha 1000 clients per 40 centrals en el qual sempre queden clients sense subministrar en l'estat final i gairebé sempre hi ha un client garantit per central) i llavors s'usa només el *swap*, operació molt més costosa que el *reset* i *move*, de fet, en ocasions actua com si es tractés d'aquesta última.

Així doncs, tenim evidència per rebutjar la hipòtesis nul·la per la alternativa, doncs hem pogut veure com hi ha una millora si utilitzem la combinació de tots els operadors junts, en comptes de qualsevol de les altres possibilitats.

3.2 Experiment 2: Influència de l'estat inicial

- Observació

Podem tenir millors solucions inicials depenent del mètode que s'apliqui per generar-les.

- Plantejament

Escollim diferents mètodes de inicialització i observem les seves solucions.

- Hipòtesis

Tots els métodos d'inicialització són iguals (H0) o hi ha mètodes millors que altres (H1).

- Mètode

- Farem l'experiment 3 vegades amb 3 *seeds* diferents.
- Farem 5 execucions per cada *seed* per a la inicialització més pròxim omplert, per a la inicialització aleatòria garantida, per la inicialització aleatòria meitat més propera garantida, per la aleatòria i per la aleatòria meitat més propera, i farem la mitjana dels resultats.
- Experimentarem amb problemas de 40 centrals i 1000 clients.
- Utilitzarem l'algorisme de Hill Climbing amb l'heurística del benefici.
- Utilitzarem la combinació d'operadors obtinguda a l'apartat anterior.
- Mesurarem els diferents paràmetres per fer la comparació.

A continuació es mostra un gràfic amb els resultats de l'experiment anterior, les dades del qual podem trobar a la taula situada al annex 5.2.

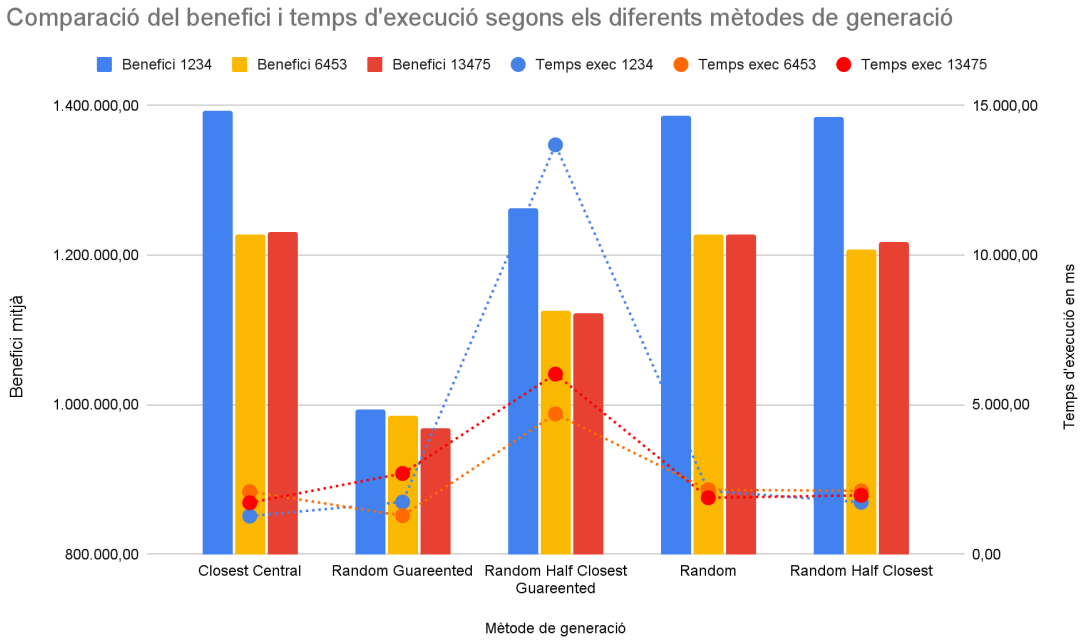


Figura 3.2.1. Comparació del benefici i temps d'execució en funció dels mètodes de generació d'estat inicial.

Al gràfic anterior podem observar com varia la mitjana del benefici per diverses *seeds* segons el mètode l'estat inicial (columnes amb l'eix a l'esquerra), tanmateix podem veure com fluctua el temps d'execució segons també segons l'estat inicial. Podem veure que tot i que la inicialització aleatòria i aleatòria meitat més propera obtenen bons resultats, és la inicialització més pròxim omplert qui obté els millors resultats si mirem la taula 5.2.1 (el qual és alhora que de mitjana té una millor solució inicial) amb un cost similar a tots els altres excepció de la inicialització aleatòria garantida. Això es degut a que només assignem als garantitzats, i això provoca dos efectes, centrals plenes probablement degudes a la distància de assignació i la heurística del benefici que no expandirà nodes que no impliquin la millora d'aquest, per tant reduirà l'espai que ocupen els clients entre centrals i finalitzarà l'execució.

Així doncs, tenim evidències per rebutjar la hipòtesis nul·la a canvi de l'alternativa, doncs hem vist que és la inicialització més pròxim omplert la que ens dona els millors estats inicials a partir dels quals poder avançar cap a la solució final.

3.3 Influència de l'heurística

- Observació

Podem tenir millors solucions inicials depenent de l'heurística emprada.

- Plantejament

Escollim dues heurístiques i observem les seves solucions.

- Hipòtesis

Totes les heurístiques són iguals (H0) o hi ha heurístiques millors que altres (H1).

- Mètode

- Farem l'experiment 3 vegades amb 3 *seeds* diferents.
- Farem 10 execucions per a cada *seed* per a la primera heurística (benefici) i 10 més per a la segona heurística (distància energia) i farem les mitjanes.
- Experimentarem amb estats de 40 centrals i 1000 clients.
- Utilitzarem l'algorisme de Hill Climbing.
- Utilitzarem la combinació d'operadors i l'estat inicial obtinguts en els apartats anteriors.
- Mesurarem els diferents paràmetres per fer la comparació.

A continuació es mostra un gràfic amb els resultats de l'experiment anterior, les dades del qual podem trobar a la taula situada al annex 5.3.

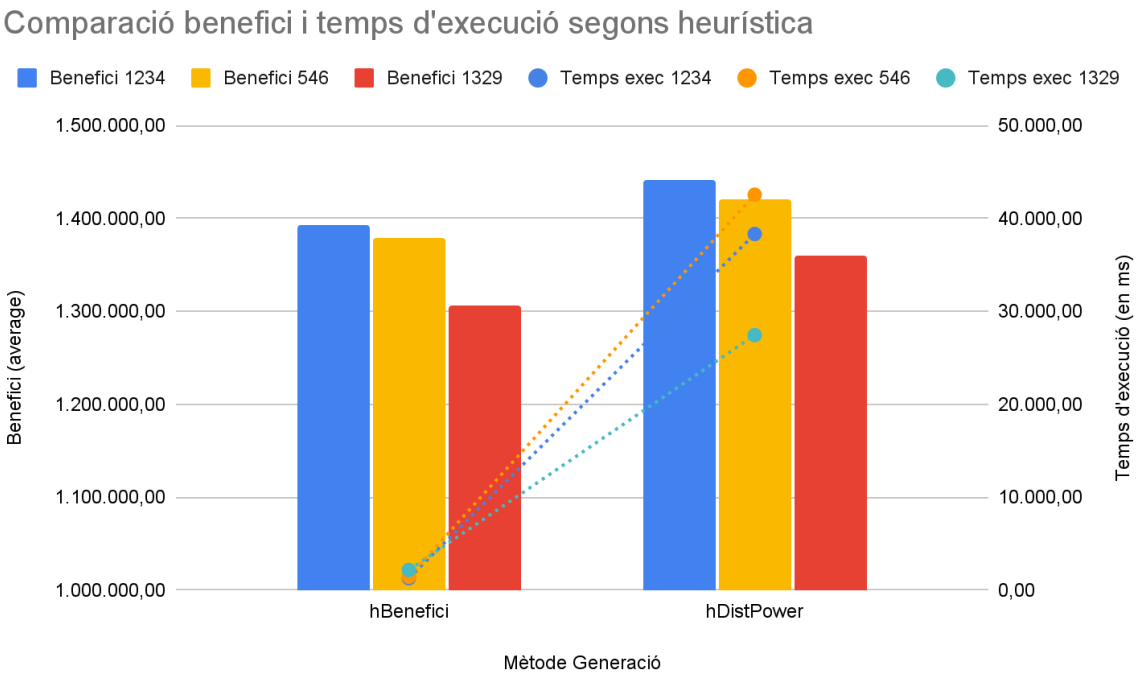


Figura 3.3.1: Comparació del benefici i el temps d'execució en funció de l'heurística utilitzada.

En el gràfic anterior podem observar que les solucions finals que ens dona la heurística distància energia són lleugerament superiors a les que s'obtenen mitjançant l'heurística del benefici. No obstant el temps que és necessari per trobar una solució executant l'algorisme amb la segona heurística incrementa de forma considerable respecte la primera, això és degut a que tarda molts més nodes per arribar al màxim(determinat per l'heurística), en comparació de mitjana heurística nova expandeix de mitjana 15 vegades més nodes que la heurística del benefici.

Nosaltres ens hem decantat per la segona heurística de cara a realitzar els següents experiments doncs el temps extra que es requereix és acceptable per la millora que produeix en la solució final.
Així doncs, en base als resultats que hem pogut observar, no tenim prou evidència per rebutjar la hipòtesis nul·la.

3.4 Experiment 3: Simulated Annealing

3.4.1 Primera part (proporció d'operadors)

- Observació

Podem tenir millors solucions inicials depenent de la proporció d'operacions pels successors establerts en el *simulated annealing*.

- Plantejament

Escollim els diferents proporcions a modificar i observem les seves solucions.

- Hipòtesis

Cap modificació en les proporcions produeix millores (H0) o hi ha proporcions d'operadors que en modificar-los millora el resultat (H1).

- Mètode

- Farem l'experiment 3 vegades amb 3 *seeds* diferents.
- Farem 10 execucions per a cada *seed* per tres proporcions d'operadors diferents 33,3% *move*, 33,3% *swap* i 75% *reset*; 24% *move*, 33,3% *swap* i 1% *reset*; i 49,5% *move*, 49,5% *swap* i 1% *reset,* i n'obtindrem les mitjanes.
- Experimentarem amb estats de 40 centrals i 1000 clients.
- Utilitzarem l'algorisme de Simulated Annealing.
- Utilitzarem la combinació d'operadors i l'estat inicial obtinguts en els apartats 3.1 i 3.2, juntament amb l'heurística de la distància energia.
- Mesurarem els diferents paràmetres fent la mitjana per fer la comparació.

A continuació es mostra un gràfic amb els resultats de l'experiment anterior, les dades del qual podem trobar a la taula situada al annex 5.4.1.

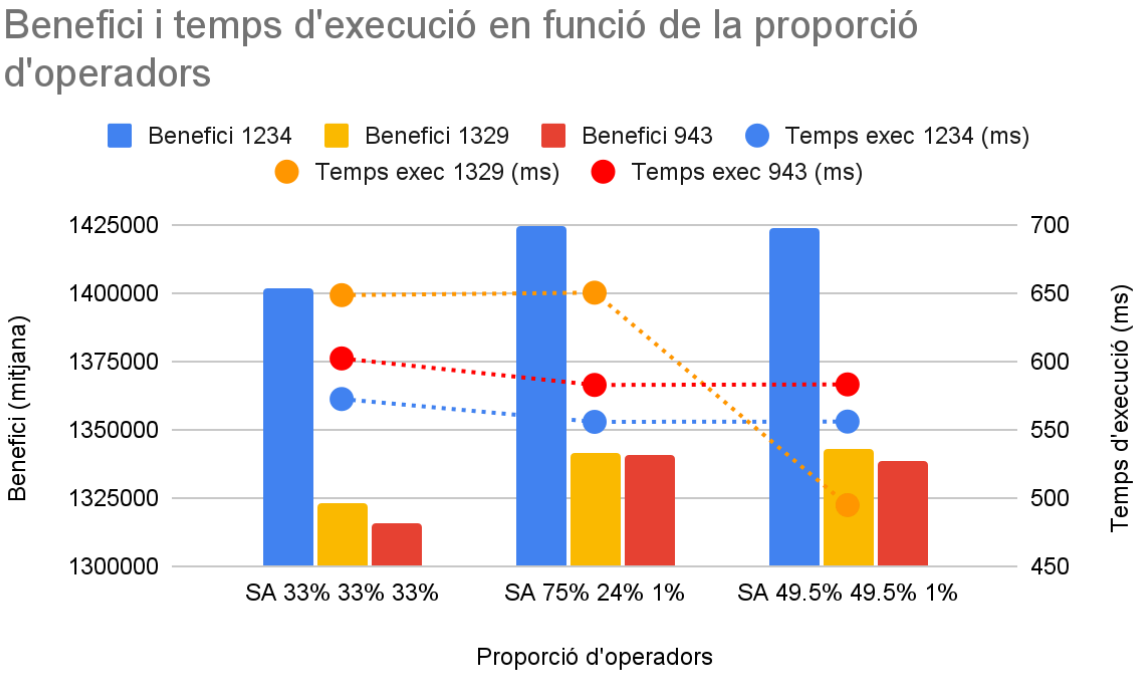


Figura 3.4.1.1: Comparació del benefici i el temps d'execució en funció dels paràmetres del Simulated Annealing (%ops).

En el gràfic anterior podem observar com si les proporcions són iguals per als tres operadors, el benefici es redueix de forma considerable respecte els altres dos. D'altra banda també es pot veure que el fet de tenir més possibilitats de fer *moves* fa incrementar lleugerament el benefici de la solució final. En quant al cost temporal aquest és gairebé el mateix per a totes .

3.4.2 Segona part (paràmetres)

- Observació

Podem tenir millors solucions inicials depenent dels paràmetres establerts en el *simulated annealing*.

- Plantejament

Escollim els diferents paràmetres a modificar i observem les seves solucions.

- Hipòtesis

Cap modificació dels paràmetres produeix millores (H0) o hi ha paràmetres que en modificar-los millora el resultat (H1).

- Mètode

- Farem l'experiment amb 1 *seed*.

- Farem 10 execucions per a cada *seed* per *k* (temperatura inicial) en l'interval de 5 a 160, per *lambda* (iteracions/1 unitat de temperatura) en l'interval de 0.0005 a 0.1, limit (profunditat) en l'interval de 25 a 300 i steps (nombre de nodes que s'expandeix fins al següent successor) en l'interval de 1000 a 1000000000, tots en 6 valors diferents dins del seu rang.
- Experimentarem amb estats de 40 centrals i 1000 clients.
- Utilitzarem l'algorisme de Simulated Annealing.
- Utilitzarem la combinació d'operadors i l'estat inicial obtinguts en els apartats 3.1 i 3.2, juntament amb l'heurística de la distància energia.
- Mesurarem els diferents paràmetres fent la mitjana per fer la comparació.

A continuació es mostra un gràfic amb els resultats de l'experiment anterior, les dades del qual podem trobar a la taula situada al annex 5.4.2.

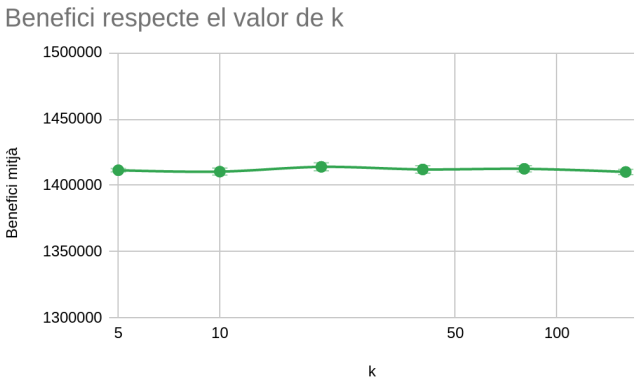


Figura 3.4.2.1. Benefici segons k.

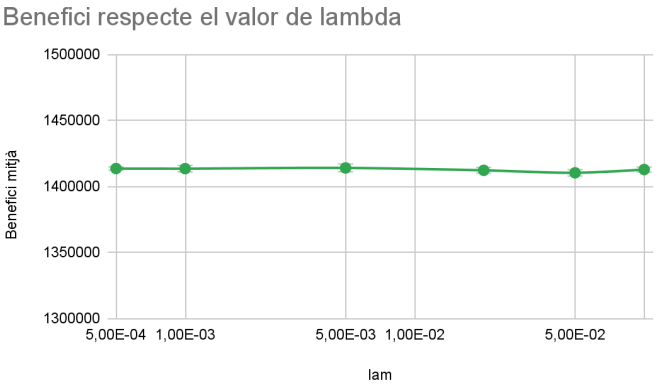


Figura 3.4.2.2. Benefici segons lambda.

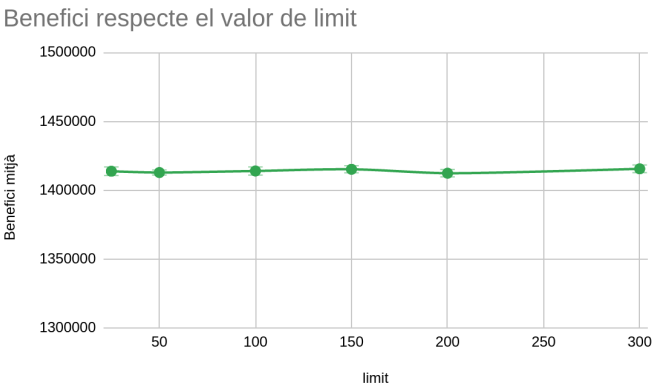


Figura 3.4.2.3. Benefici segons limit.

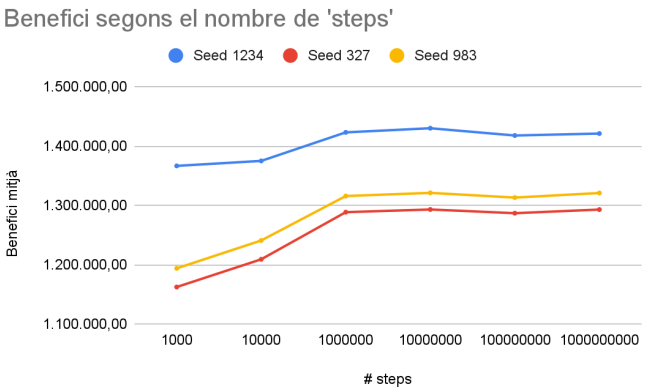


Figura 3.4.2.4. Benefici segons steps.

Com podem observar a les gràfiques anteriors, el fet de variar els paràmetres *k*, *lambda* i *limit*, no provoca efecte de cara a la solució final, es manté gairebé constant

independentment dels seus valors. No obstant és substancial la millora que es pot obtenir si s'incrementa el nombre de steps que fa el Simulated Annealing.

Així doncs, hem decidit mantenir els valor per defecte dels paràmetres del Simulated Annealing per als pròxims experiments a excepció dels *steps* que com hem vist provoca una millora.

Donat que al modificar un dels paràmetres els resultats milloren, tenim evidència per rebutjar la hipòtesis nul·la.

3.5 Experiment 4: Influència del nombre de clients i centrals

- Observació

El temps d'execució pot augmentar o disminuir depenent del nombre de clients i centrals.

- Plantejament

Escollim rang de clients i centrals, i els anem modificant. Observem les seves solucions.

- Hipòtesis

Per qualsevol rang de client o centrals els temps d'execució són iguals (H0) o el temps d'execució és diferent en variar el nombre de clients o centrals (H1).

- Mètode

- Farem l'experiment amb 1 *seed*.
- Farem 5 execucions per cada *seed* per a cada valor del l'interval de clients començant per 500 i augmentant en 500, fixant el nombre de centrals a 40, fins a veure una tendència.
- Executarem 1 experiment per cada *seed* per a cada valor del rang de clients començant per 40 i augmentant en 40, fixant el nombre de centrals a 1000, fins a veure una tendència.
- Utilitzarem l'algorisme de Hill Climbing.
- Utilitzarem la combinació d'operadors i l'estat inicial obtinguts en els apartats 3.1 i 3.2, juntament amb l'heurística de la distància energia.
- Mesurarem els diferents paràmetres per fer la comparació.

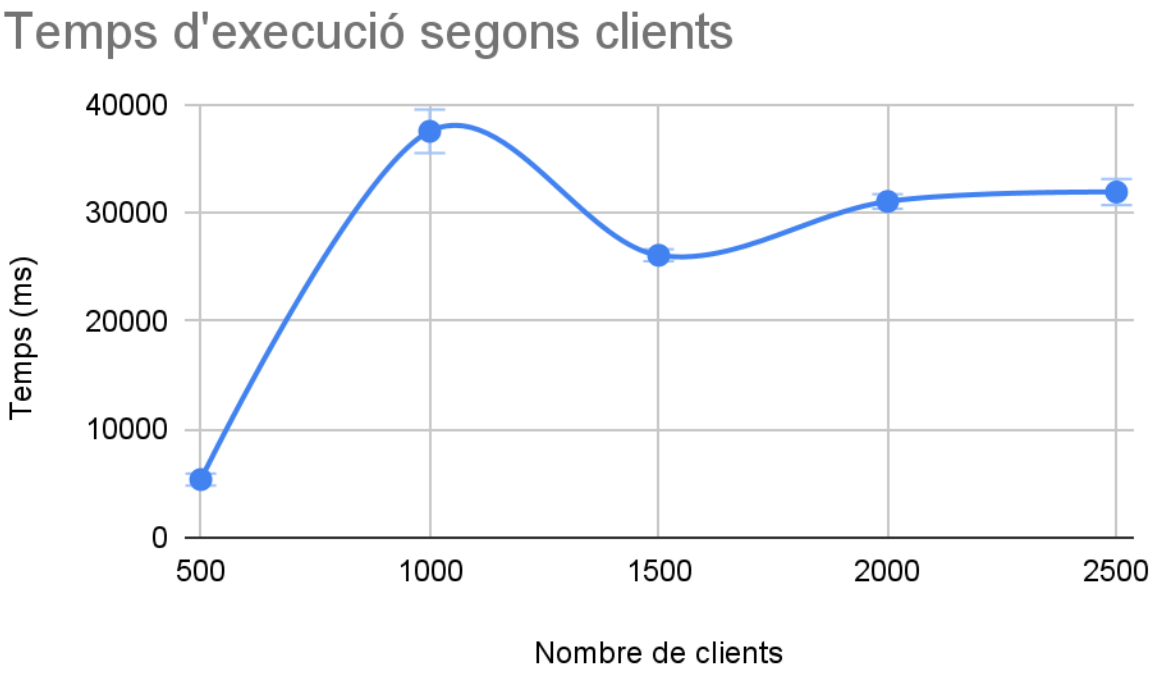


Figura 3.5.1: Temps d'execució en funció del nombre de clients.

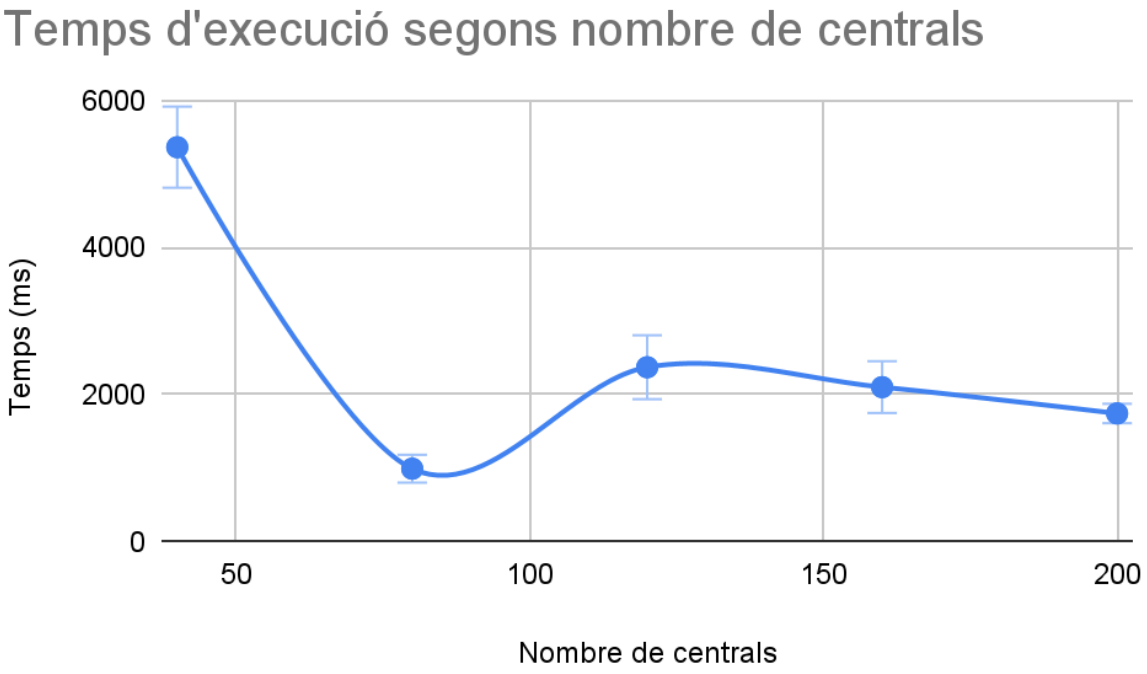


Figura 3.5.2: Temps d'execució en funció del nombre de centrals.

Observant la figura 3.5.1, on se'ns presenta el temps d'execució en funció del nombre de clients que hi ha, podem veure que, a mida que anem augmentant el nombre de clients (sense tocar el de centrals) el temps augmenta fins arribar a un màxim relatiu en els 1000

clients, amb una posterior davallada fins als 1500 clients, però després torna a remuntar i, a partir d'aquell punt, a més clients hi ha, més temps triga en executar-se el programa. Podem dir, per tant, que tenim prou evidència com per refutar la hipòtesis nul·la i llavors podem donar per vàlida la alternativa, concloent per tant que el nombre de clients molt probablement sí que afecta al temps d'execució, en aquest cas amb una tendència a l'alça a llarg termini (a més clients, més temps d'execució).

Si ens fixem ara en la figura 3.5.2, on es troben representats els temps d'execució en funció del nombre de centrals, tenint el nombre de clients fix, podem observar que, al principi hi ha una davallada bastant forta del temps d'execució per a 80 centrals, en comparació amb l'inicial de 40. No obstant, des de les 80 centrals fins a les 120, el temps d'execució creix, per tornar a mostrar una tendència decreixent a partir d'aquell punt. Així doncs, aquí també trobem prou evidència com per refutar la hipòtesi nul·la i poder acceptar l'alternativa, és a dir, que podem concloure que el nombre de centrals afecta al temps d'execució, en aquest cas, es veu una tendència a la baixa a llarg termini (a més centrals, menys temps d'execució).

3.6 Experiment 5: Penalització per no servir als clients garantits

- Observació

La solució final pot ser no vàlida si la penalització per deixar un client garantit sense energia és insuficient.

- Plantejament

Començarem amb una penalització de 1000 i anirem augmentant de 10 en 10, observem les solucions.

- Hipòtesis

A partir de 10000, els estats seran vàlids en els dos algorismes.

H0: El benefici és pitjor en cas de tenir l'heurística de Penalització.

H1: El benefici no és pitjor en cas de tenir l'heurística de Penalització.

- Mètode

- Escollirem 5 *seeds* aleatòries, una para cada rèplica.
- Executarem 1 experiment per cada *seed* des de 0 fins que es veu que totes les solucions són vàlides, utilitzant l'algorisme de Hill Climbing.
- Executarem 5 experiment per cada *seed* des de 0 fins que es veu que totes les solucions són vàlides, utilitzant l'algorisme de Simulated Annealing i n'extraurem la mitjana.
- Utilitzarem la combinació d'operadors i l'estat inicial obtinguts en els apartats 3.1 i 3.2, juntament amb l'heurística de la distància energia.
- Mesurarem els diferents paràmetres per fer la comparació.

	Estat Vàlid HC Heurística Nova	Estat Vàlid SA Heurística Nova
1000	false	false
10000	false	false
100000	false	false
1000000	false	false
10000000	false	false
100000000	true	false
1000000000	true	false

Taula 3.6.1. Valors de penalització vàlids.

En el cas del SA, veiem com no ens obté una solució vàlida, això ho atribuïm a que assigna gran part dels clients garantits de forma aleatoria veient que simplement maximitza l'heurística, però un cop ja ha utilitzat gran part de les centrals, és massa "complicat" moure

els clients per crear espais en les centrals, només si ho comproves per totes les possibles combinacions des d'un estat com ho fa el Hill climbing. Tot i així no descartem que sigui possible crear una heurística amb penalització que aconsegueixi un estat vàlid per SA.

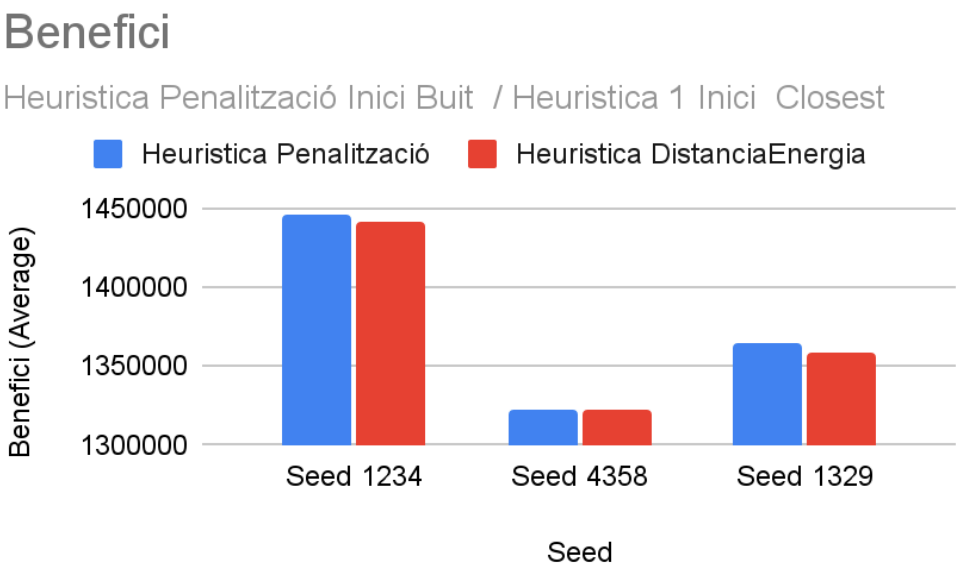


Figura 3.6.1: Benefici amb heurística distància energia i la de penalització per a diferents seeds.

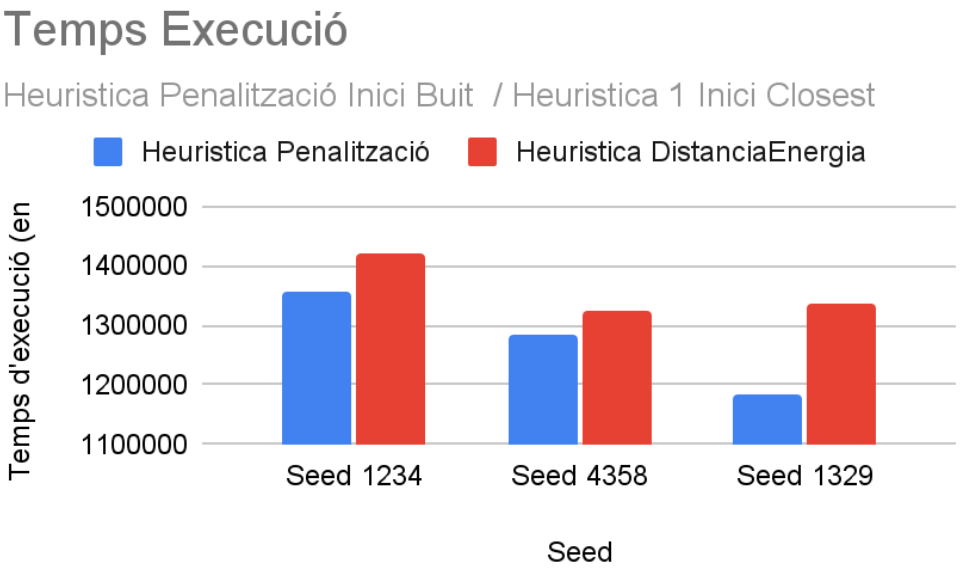


Figura 3.6.2: Temps d'Execució amb l'heurística DistanciaEnergia i la de penalització per a diferents seeds.

Durant l'experimentació hem observat que el Simulated Annealing quasi sempre dona resultats incorrectes, sense importar el valor dels paràmetres que s'hi posi, cosa que té sentit ja que cal recordar que aquest algorisme treballa de forma aleatòria, per tant, és possible que acabi en un cas on la heurística sigui pitjor (en aquest cas, que l'estat no sigui

vàlid) i, per tant, si es mou des d'aquell node, és bastant probable que no torni a un de vàlid, ja que la probabilitat es va reduint cada cop més.

Per altra banda, amb el Hill Climbing, observem que l'heurística de penalització (que ens porta sempre a estats vàlids a partir de 100.000.000, nombre tant gran degut a que elevem al quadrat a la nostre heurística altres valors, de penalització en accedir a un de no vàlid), ens dona un millor benefici de mitjana, essent el temps d'execució bastant menor que el de l'heurística distància energia (això es deu també a la forma en què es genera la solució inicial).

Així doncs, rebutjem la hipòtesis inicial, ja que el nombre és bastant major al que havíem suposat, i tenim evidència suficient per rebutjar la hipòtesis nul·la, suposant així que l'alternativa té bastant possibilitats de ser certa, al menys en el Hill Climbing (ja que l'annealing ens donava resultats invàlids).

3.7 Experiment 6: Influència nombre de centrals petites

- Observació

El temps d'execució pot augmentar o disminuir depenent del percentatge de centrals petites que hi ha.

- Plantejament

Escollim diferents proporcions de tipus de central i observem les seves solucions.

- Hipòtesis

H0: El temps és igual encara que augmenti el nombre de centrals de tipus C.

H1: El temps varia a mida que augmenti el nombre de centrals de tipus C.

- Mètode

- Escollirem 5 *seeds* aleatòries, una para cada rèplica.
- Executarem 1 experiment per cada *seed* per a 25, 50 i 75 centrals de tipus C amb 10 centrals de tipus B i 5 de tipus A, utilitzant Hill Climbing.
- Executarem 5 experiment per cada *seed* per a 25, 50 i 75 centrals de tipus C amb 10 centrals de tipus B i 5 de tipus A, utilitzant Simulated Annealing, farem la mitjana.
- Utilitzarem 1000 clients.
- Utilitzarem la combinació d'operadors i l'estat inicial obtinguts en els apartats 3.1 i 3.2, juntament amb l'heurística de la distància energia.
- Mesurarem els diferents paràmetres per fer la comparació.

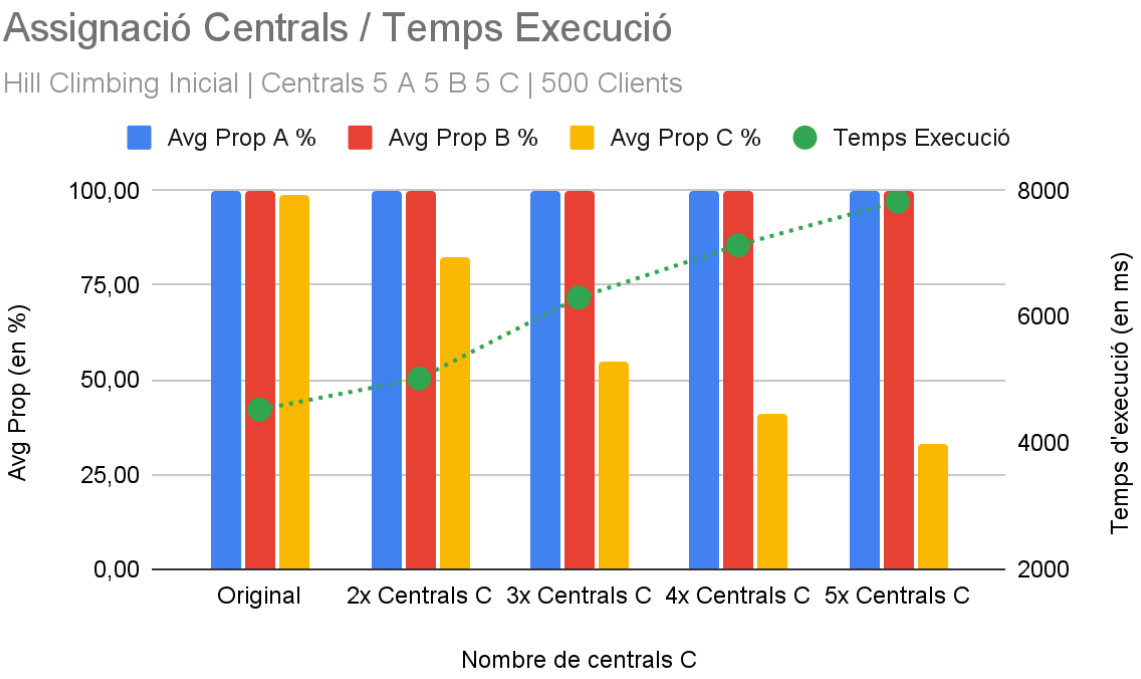


Figura 3.7.1: Assignació a centrals (segons el tipus) i temps d'execució en funció del nombre de centrals de tipus C (amb Hill Climbing)

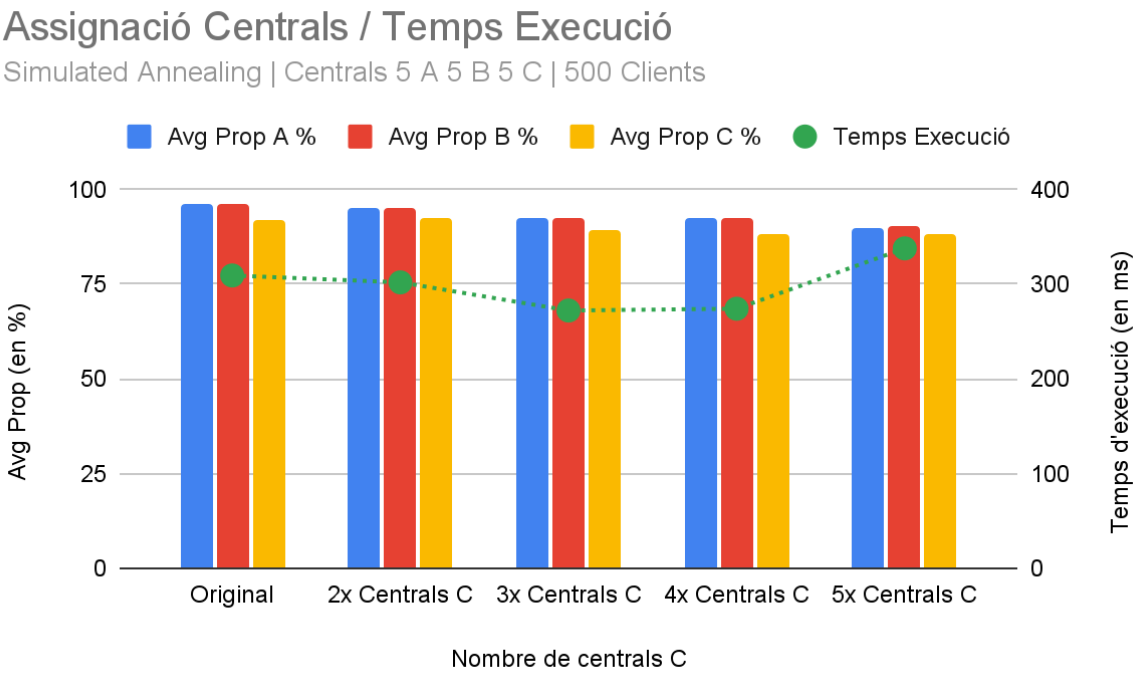


Figura 3.7.2: Assignació a centrals (segons el tipus) i temps d'execució en funció del nombre de centrals de tipus C (amb Simulated Annealing)

Fixem-nos que en el cas del Hill Climbing. Observant la figura 3.7.1, veiem però que la proporció d'ocupació de les centrals tipus C decau bastant a mida que més n'hi ha. Contràriament, el temps d'execució augmenta a mida que augmenta el nombre de centrals de tipus C, per tant es pot establir una relació directa entre el nombre de centrals de tipus C i el temps d'execució, tenint així prou evidència que sosté la nostra hipòtesis alternativa i, per tant, refutar la nul·la.

En el cas del Simulated Annealing (Figura 3.7.2), veiem que la proporció d'ocupació de les centrals de tipus C disminueix lleugerament a mida que el nombre d'aquestes augmenta, però en aquest cas (i en contraposició amb el que succeïa al HC) també ho fa el percentatge de les centrals de tipus A i B. En aquest cas és més complicat establir un patró amb el temps, ja que en alguns casos disminueix però sembla que al final la tendència és a l'alça (tot i que, al no tenir valors més enllà de 5x Centrals C i no tenir una trajectòria clara abans, no ho podem confirmar). Per tant, tot i que tenim prou evidència per refutar la hipòtesis nul·la i a canvi de l'alternativa, no podem predir amb tanta certesa com amb el Hill Climbing què passarà amb el temps d'execució a mida que el nombre de centrals de tipus C augmenti, per fer-ho necessitariem executar més experiments.

4. Conclusions

Després de realitzar els experiments vistos en l'apartat anterior, ja podem donar les idees clau i les conclusions pel que fa a l'anàlisis que s'ha dut a terme per cada estudi.

Primerament, pel que fa als algoritmes que hem utilitzat per dur a terme la pràctica hem vist que el Hill Climbing és més lent que el Simulated Annealing ja que ha d'expandir en totes les direccions per tal de trobar el millor successor (assegurant-nos d'aquesta forma que sempre anirà cap amunt, però amb l'inconvenient que pot quedar-se en màxim locals més fàcilment), en canvi Simulated Annealing és molt més ràpid com hem vist en els diferents experiments duts a terme, i té un factor d'aleatorietat bastant interessant, doncs tot i que no pots determinar si el resultat final que n'obtens serà millor o pitjor, aquesta aleatorietat permet poder obtenir millors resultats que amb el Hill Climbing depenent del cas d'aplicació.

A més a més també hem vist com el fet de seleccionar bons operadors i un bon mètode per generar l'estat inicial, poden ser claus de cara a millorar la solució final ja que aquesta se'n veu afectada directament. Tanmateix és també molt important fer una bona selecció de l'heurística, que en un inici pot semblar bona, encara que si s'experimenta podem veure que no dona tants bons resultats com hauríem pensat, doncs que l'heurística sigui també l'objectiu impedeix que s'utilitzin alguns dels recursos que es podrien emprar per millorar encara més la solució.

D'altra banda també cal destacar que prèviament l'execució era exegeradament llarga, això era degut a tres factors: la heurística que recalculava valors de: distància, benefici i assignacions…; errors alhora de fer comprovacions en les restriccions de les operacions *swap* o *move*; i finalment quan i com es feien les còpies dels estats ja que aquestes podien arribar a ser molt costoses i alhora innecessàries.
Passem doncs de tenir una execució de casi una hora reduïda a prop d'un minut, i en el cas del annealing tan sols uns segons.

A partir dels experiments realitzats en aquest treball, i de les conclusions extretes, hem arribat a la conclusió que sería útil realitzar més experiments, com provar diferents heurístiques, provar altres algorismes de cerca, o veure com afecta el nombre de clients de cada tipus al benefici i temps d'execució, entre d'altres. Tanmateix seria necessari augmentar de forma considerable el nombre de *seeds* que s'utilitzen a cadascun dels experiments i emprar algun tipus de scrypt per tal d'optimitzar la forma en què s'analitzen

les dades alhora que automatitza el procés ergo s'estalvia temps, temps que es pot usar per fer encara més execucions i per tant obtenir un resultat més fiable.

5. Annex

Link a les dades en cru: Experimentació LocalSearch

5.1 Dades de l'experiment de la influència dels operadors

Hipervincle a l'experiment 3.1.

	Comparació del benefici segons els operadors		
Operació	Seed 4358	Seed 1234	Seed 875346
Move	1.289.668,00	1.368.201,90	1.377.082,00
Swap	1.297.096,00	1.384.155,00	1.377.082,00
Reset	1.289.668,00	1.368.201,90	1.377.082,00
Move+Swap	1.297.096,10	1.393.356,90	1.377.082,00
Move+Reset	1.289.668,60	1.368.201,90	1.377.082,40
Swap+Reset	1.297.096,10	1.384.155,00	1.377.082,00
All	1.300.005,00	1.393.356,90	1.377.082,00

Taula 5.1.1. Mitjana dels beneficis segons conjunt d'operacions.

	Comparació del temps d'execució segons els operadors					
Operació	Seed 4358 (avg / desviació std)		Seed 1234 (avg / desviació std)		Seed 875346 (avg / desviació std)	
Move	7,00	2,83	7,20	2,17	13,4	10,33
Swap	1.588,60	120,17	2332,8	479,91	118,4	51,29
Reset	7,00	6,40	2,8	2,49	4	6,71
Move+Swap	774	192,25	1385	181,04	138,8	43,21
Move+Reset	6,2	8,90	10,4	5,18	13,8	9,58
Swap+Reset	1016,2	229,01	2777	364,72	122,2	46,00
All	788,6	183,03	1299	37,98	74,2	18,13

Taula 5.1.2. Mitjana dels temps d'execució segons conjunt d'operacions.

5.2 Dades de l'experiment de la influència de l'estat inicial

Hipervincle a l'experiment 3.2.

	Seed 1234		Seed 6453		Seed 13475	
Mètode Generació	Benefici 1234 (avg)	Temps exec 1234 (avg) (ms)	Benefici 6453 (avg)	Temps exec 6453 (avg) (ms)	Benefici 13475 (avg)	Temps exec 13475 (avg) (ms)
Closest	1.393.356,9	1.274,50	1.228.228,0	2.093,00	1.230.800,6	1.721,70

Central	0		0		0	
Random Guareented	994.314,04	1.746,30	985.457,30	1.286,60	968.544,49	2.700,20
Random Half Closest Guareented	1.262.000,60	13.684,10	1.125.682,60	4.693,00	1.123.057,35	6.021,20
Random	1.386.056,34	2.119,90	1.227.565,60	2.155,50	1.227.892,70	1.887,60
Random Half Closest	1.384.472,70	1.736,80	1.207.417,00	2.118,70	1.218.275,85	1.967,40
Mètode Generació	Benefici 1234 (desviacio std)	Temps exec 1234 (desviacio std)	Benefici 6453 (desviacio std)	Temps exec 6453 (desviacio std)	Benefici 13475 (desviacio std)	Temps exec 13475 (desviacio std)
Closest Central	0,00	14,25	0,00	29,04	0,00	203,96
Random Guareented	20.446,58	739,87	37.085,25	663,06	34.983,84	4.209,45
Random Half Closest Guareented	4.711,05	2.968,53	4.156,61	647,40	4.582,59	1.187,87
Random	7.453,75	662,98	3.166,99	271,17	3.699,41	124,13
Random Half Closest	4.582,83	247,66	5.586,50	130,92	3.049,47	144,81

Taula 5.2.1. Mitjana dels temps d'execució, beneficis i les seves respectives desviacions estàndard segons la generació inicial.

5.3 Dades de l'experiment de la influència de l'heurística

Hipervincle a l'experiment 3.3.

	Seed 1234		Seed 546		Seed 1329	
Mètode Generació	Benefici 1234	Temps exec 1234	Benefici 546	Temps exec 546	Benefici 1329	Temps exec 1329
hBenefici	1.393.356,90	1.274,50	1.378.454,00	1.435,60	1.306.852,00	2.161,00
hDistPower	1.441.187,00	38.307,10	1.419.841,00	42.538,30	1.359.075,00	27.427,90
Mètode Generació	Desviacions estàndard respectives					
hBenefici	0,00	14,25	0,00	23,91	0,00	21,17
hDistPower	0,00	2.873,30	0,00	5.928,38	0,00	1.182,57

Taula 5.3.1. Mitjana dels temps d'execució, beneficis i les seves respectives desviacions estàndard segons l'heurística emprada.

5.4 Dades de l'experiment del Simulated Annealing

Hipervincle a l'experiment 3.4.

5.4.1 Primera part (proporció d'operadors)

Hipervincle a l'experiment 3.4.1.

	Seed 1234		Seed 1329		Seed 943	
Proporció	Benefici 1234	Temps exec 1234 (ms)	Benefici 1329	Temps exec 1329 (ms)	Benefici 943	Temps exec 943 (ms)
SA 33% 33% 33%	1.401.971,74	572,40	1.322.818,60	648,40	1316040,1	602,1
SA 75% 24% 1%	1.424.786,90	555,70	1.341.358,20	650,20	1340998,7	582,7
SA 49.5% 49.5% 1%	1.423.797,30	555,90	1.342.659,40	494,80	1338867,9	583,1
Proporció	Desviacions estàndard respectives					
SA 33% 33% 33%	4.943,54	108,63	2.056,54	115,02	3.144,88	145,88
SA 75% 24% 1%	4.103,55	122,36	2.517,95	157,53	3.293,97	168,46
SA 49.5% 49.5% 1%	6.250,97	118,78	3.063,36	117,92	4.844,50	163,71

Taula 5.4.1.1. Benefici i temps d'execució segons la proporció d'operadors.

5.4.2 Segona part (paràmetres)

Hipervincle a l'experiment 3.4.2.

k	Benefici
5	1.411.402,40
10,00	1.410.352,40
20,00	1.413.978,40
40,00	1.412.014,90
80,00	1.412.490,90
160,00	1.410.131,90

Taula 5.4.2.1. Benefici segons k.

lambda	Benefici
5,00E-04	1.413.509,40
1,00E-03	1413435,9
0,01	1.413.978,40

2,00E-02	1.412.158,40
5,00E-02	1.410.285,90
1,00E-01	1.412.753,40

Taula 5.4.2.2. Benefici segons lambda.

limit	Benefici
25	1.413.782,40
50	1.412.844,40
100	1.413.978,40
150	1.415.206,90
200,00	1.412.354,40
300,00	1.415.546,40

Taula 5.4.2.3. Benedici segons limit.

steps	Seed 1234	Seed 327	Seed 983
1000	1.366.746,90	1.162.410,20	1.193.884,30
10000	1.375.118,40	1.209.176,20	1.240.919,30
1000000	1.423.261,40	1.288.653,20	1.315.880,30
10000000	1.430.228,90	1.293.203,20	1.321.226,30
100000000	1.417.978,90	1.286.940,70	1.313.284,80
1000000000	1.421.258,40	1.293.073,70	1.320.991,80

Taula 5.4.2.4. Benefici segons steps.

5.5. Dades de l'experiment de la influència del nombre de centrals i clients

Hipervincle a l'experiment 3.5.

Clients	Mitjana
500	5.362,50
1000	37.518,10
1500	26.070,50
2000	31.039,90
2500	31.911,30

Taula 5.5.1. Temps d'execució segons nombre de clients.

Centrals	Mitjana
40	5.362,50

80	994,50
120	2.372,90
160	2.102,60
200	1.745,05

Taula 5.5.2. Temps d'execució segons nombre de centrals.

5.6 Dades de l'experiment de la penalització per no servir als clients garantits

Hipervincle a l'experiment 3.6.

	Benefici Hill Climbing	
	Heuristica Penalització	Heuristica DistanciaEnergia
Seed 1234	1446101	1441187
Seed 4358	1322500	1322500
Seed 1329	1364260	1359075

Taula 5.6.1. Benefici obtingut utilitzant l'heurística de Penalització amb el Hill Climbing.

	Benefici Simulated Annealing	
	Heuristica Penalització	Heuristica DistanciaEnergia
Seed 1234	1357797	1422126
Seed 4358	1284339	1322500
Seed 1329	1185175	1338089

Taula 5.6.2. Benefici obtingut utilitzant l'heurística de Penalització amb el Simulated Annealing.

5.7 Dades de l'experiment de la influència de centrals petites.

Hipervincle a l'experiment 3.7.

	HC 500 Clients HBenefici Empty Solution 5 A 5 B 5 C				
	Original	2x Centrals C	3x Centrals C	4x Centrals C	5x Centrals C
Avg Prop A %	99,74	99,74	99,74	99,74	99,74
Avg Prop B %	99,74	99,74	99,74	99,74	99,74
Avg Prop C %	99,00	82,30	54,87	41,15	32,92
Temps Execució	4529,9	5019,7	6303,1	7134	7827,8

Taula 5.7.1. Proporcions d'ocupació mitjanes i temps d'execució mitjà amb Hill Climbing.

	SA 500 Clients HBenefici Empty Solution 5 A 5 B 5 C				
	Original	2x Centrals C	3x Centrals C	4x Centrals C	5x Centrals C
Avg Prop A %	96,056	95,309	92,6811	92,45	89,89
Avg Prop B %	96,056	95,309	92,6811	92,45	90,25
Avg Prop C %	92,043	92,298	89,518	88,22	88,45
Temps Execució	309,2	301,9	272,20	274,10	337,80

Taula 5.7.2. Proporcions d'ocupació mitjanes i temps d'execució mitjà amb Simulated Annealing.

6. Estat del treball d'innovació

Imagen Video és un sistema de text a generació de vídeo basat en una cascada de models de difusió. Imagen Video és capaç de generar video d'alta qualitat amb un text d'entrada fent ús de models de superesolució de video.

Fins ara la persona que s'ha dedicat a buscar més informació envers el treball d'innovació ha estat Joan Casahuga, tanmateix els altres dos integrants hem verificat la qualitat del material per decidir la viabilitat del projecte amb la informació disponible, a més d'haver buscat altres IAs pel projecte, que es van acabar descartant per falta d'informació relativa a elles, fet que va atrasar la posada en marxa del treball.

Ho, J., Chan, W., Saharia, C., Whang, J., Gao, R., Gritsenko, A., ... & Salimans, T. (2022). Imagen Video: High Definition Video Generation with Diffusion Models. *arXiv preprint arXiv:2210.02303*. Accessed on October 9, 2022. https://arxiv.org/abs/2210.02303

Imagen Video Website. Google Research. Accessed on October 9, 2022. https://imagen.research.google/video/.

Un dels inconvenients més importants en buscar informació que ens hem trobat ha estat el fet que Google és una de les empreses que col·labora amb el projecte i creiem que això fa que hi hagi menys informació disponible de la que ens agradaria. A més a més també haurem de vigilar em qualsevol font donat que el *paper* principal es tracta d'un *preprint*.