Pràctica Cerca Local

Intel·ligència Artificial

Q1 2022-23

Pere Carrillo

Marc Ordóñez

Laura Pérez

Índex

1 Descripció del problema

Les empreses de generació d'electricitat són les encarregades de gestionar parcs de centrals elèctriques. Aquestes centrals són de diferents tipus i poden produir una certa quantitat de MW diaris. No és rendible mantenir totes les centrals enceses, ja que per norma general la producció de les centrals supera la demanda dels clients als que han d'abastir. El nostre programa serà l'encarregat de donar una solució a aquestes empreses per decidir quines centrals tenir enceses i a quins clients ha de subministrar cada central en funció de la demanda que hi hagi aquell dia per tal de maximitzar el benefici d'aquestes empreses elèctriques. La solució serà una assignació client per client a una de les centrals o a cap si no se li subministra energia.

1.1 Elements

Tenim 3 tipus de centrals: A, B, C. Cadascuna pot produir una certa quantitat de MW dins d'un rang de produccions en funció del tipus. Cada classe de central té un cost de posada en marxa (un cost fix i un variable que varia segons la producció màxima de la central) i un cost de parada (constant). Aquests costos són diferents depenent del tipus de central. Una central encesa sempre produirà la màxima quantitat d'energia possible tot i que els clients als que subministra no necessitin tanta. El cost de produir aquesta energia excedent també s'ha de tenir en compte tot i que no s'acabi subministrant a cap client.

Tipus	Producció	Cost marxa	Cost parada
A	250 a 750 MW	Prod*50 + 20000	15000
B	100 a 250 MW	Prod*80 + 10000	5000
C	10 a 100 MW	Prod*150 + 5000	1500

Per altra banda tenim els clients (consumidors). Els podem separar en dos grups principals: Garantits i No Garantits, en funció de la tarifa que escullin. Als garantits se'ls assegura el subministrament que tenen contractat en qualsevol circumstància. Als no garantits se'ls pot deixar sense el subministrament contractat. Si en un dia es deixa a un client no garantitzat sense subministrament, se l'indemnitzarà pagant-li 50€ per cada MW contractat.

Els clients que més consumeixen poden arribar a negociar diferents tarifes en funció de les seves necessitats per disminuir el preu que paguen per MW. Partint d'aquest criteri, distingim tres grups depenent del consum que necessiten: eXtra Grans (XG), Molt Grans (MG) i Grans (G). Cada tipus de client haurà d'escollir si vol un servei garantit o no garantit. En funció de tots aquests criteris es determina el preu que han de pagar.

Client	Consum	Garantit	No Garantit	Indemnització
XG	5 a 20 MW	400 €/MW	300 €/MW	50 euros/MW
MG	2 a 5 MW	500 €/MW	400 €/MW	
G	1 a 2 MW	600 €/MW	500 €/MW	

La solució que donem s'ha d'assegurar que a tots els clients garantits se'ls assigni l'energia contractada.

Tant els clients com les centrals tindran unes coordenades (x, y) concretes dins d'una àrea de 100 x 100 quilòmetres. El transport de l'electricitat a grans distàncies comporta una pèrdua d'energia que incrementa com més gran sigui la distància entre la central i el client al que subministra. Tindrem diferents rangs de distàncies amb percentatges de pèrdua més grans per a distàncies més grans, que utilitzarem per calcular l'energia que haurem de transportar.

Distància	Pèrdua
Fins a 10 km	0%
Fins a 25 km	10%
Fins a 50 km	20%
Fins a 75 km	40%
Més de 75 km	60%

A cada client només el podrà subministrar una única central i li haurà de proporcionar tota l'energia que hagi contractat. Caldrà tenir en compte la pèrdua d'energia per tal que al client li arribi l'energia sol·licitada, en els casos en que hi hagi pèrdua d'energia, la central haurà de produir més del que el client demana. Aquesta producció extra mai la pagarà el client, és una despesa que haurà de cobrir la central. La distància client-central es calcula fent servir la distància euclídea entre les coordenades de la central i les del client.

1.2 Estat

Per decidir com representar l'estat del problema primer vam veure quins elements i informació necessitàvem mantenir per poder guardar tots els paràmetres de l'estat. Calia tenir informació de tots els clients i de totes les centrals i guardar l'assignació de cada client a una central.

Vam plantejar diverses formes de guardar aquesta informació. En primer lloc, podíem tenir un vector on cada posició representés una central i assignar a cada posició un vector o llista amb tots els clients als que ha de subministrar aquella central. També vam pensar en fer-ho a la inversa: tenir un vector on cada posició representa un client i se li assigna una central o -1 si cap central subministra al client. Per tal de no guardar tots els clients al vector de centrals o les centrals al vector de clients es pot fer ús d'un índex que faci referència a una posició del ArrayList de *Centrales* o de *Clientes*.

Al final vam decantar-nos per l'opció de tenir un vector on cada posició representa un client i es guardava l'índex de la central que el subministra (o -1 si no té assignada cap central). Vam prendre aquesta decisió després d'analitzar els operadors que volíem fer servir. Tant pel *swap* com per l'*assignar* és molt més senzill tenir un vectors de clients, ja que pots saber a quina central estan assignats aquests clients només mirant la posició del vector. Si tinguéssim el vector de centrals hauríem d'anar una per una buscant en quina central estan assignats els clients que volem operar.

També vam veure que hi havia una sèrie d'informacions que necessitàvem accedir constantment i que si calculàvem tota l'estona ralentitzarien molt el programa. Per exemple, el benefici que s'obté amb una assignació concreta o l'energia excedent d'una central que es pot fer servir per subministrar a nous clients.

Finalment vam decidir guardar:

- Un double anomenat **benefici**, que haurà de tenir en compte els costos de posada en marxa de les centrals, els costos de parada, les indemnitzacions dels clients no garantits als que s'ha assignat cap central i el preu que pagaran els clients per l'energia que els subministrem.
- Un vector de doubles anomenat **excedent** de mida n, on n és el nombre de centrals, que guarda l'energia excedent d'una central. És a dir, la seva producció restant-li l'energia que ja ha consumit per subministrar als clients que ja té assignada.

- Un vector de ints anomenat **centralClients** de mida *cl*, on *cl* és el nombre de clients i que guarda l'índex de la central a la que s'assigna aquell client.
- L'arrayList de **clients**, que serà static, ja que per tots els possibles estats del problema els clients seran sempre els mateixos amb els mateixos valors.
- L'arrayList de **centrals**, que serà static, ja que per tots els possibles estats del problema les centrals seran sempre les mateixes amb els mateixos valors.
- Un double anomenat **energiaPerduda**, on guardarem tota l'energia que es perd des de totes les centrals cap a tots els clients als que subministren per culpa de la distància.

1.3 Operadors

Hem dissenyat quatre possibles operadors. En tots ells considerarem que un índex de central és vàlid si és un enter entre 0 i el nombre de centrals o si és -1. Un índex de client serà vàlid si és un enter entre 0 i el nombre de clients. En totes les funcions s'actualitzen els paràmetres que controlen l'estat del problema, com poden ser el benefici, els excedents o l'energia perduda. No ho explicarem en cadascuna de les funcions perquè són canvis que sempre es faran.

Assignar

- **Paràmetres:** `int idx_cl, int idx_centralNova`
- **Condició d'aplicabilitat**: *idx_cl* és un índex de client vàlid i *idx_centralNova* és un índex de central vàlid. La central on volem assignar el client té suficient excés de producció per poder subministra-li l'energia contractada (tenint en compte l'extra d'energia que es requerirà per les pèrdues).
- **Funció de transformació**: Assigna un client a una central, si aquest ja estava assignat, es desassigna al client de la seva antiga central i se l'assigna a la nova central.

Eliminar

- **Paràmetres:** `int idx_cl`
- **Condició d'aplicabilitat**: *idx_cl* és un índex de client vàlid i es tracta d'un client no garantit.
- **Funció de transformació**: Desassigna un client de la central on estigués assignat. Només es pot fer servir amb client no garantits.

Swap

- **Paràmetres:** `int idx_cl1, int idx_cl2`
- **Condició d'aplicabilitat**: *idx_cl1* i *idx_cl2* són índexs de clients vàlids. En cas que un d'ells no estigui assignat a cap central (client No Garantit), l'altre client haurà de ser No Garantit (perquè al fer swap quedarà desassignat i només podem desassignar als no garantits. En la central del client 1 ha d'haver prou excés de producció d'electricitat de manera que quan deixi de subministrar al client 1 pugui subministrar energia al client 2 (tenint en compte l'extra d'energia que es requerirà per les pèrdues). S'ha de complir la mateixa condició però a la inversa.
- **Funció de transformació**: Intercanvia les centrals de dos clients. En cas que un d'ells (client A) no estigui assignat a cap central i l'altre (client B) sigui No Garantit, client A passa a estar assignat a la central de client 2 i client 2 passa a no estar assignat a cap. Si tots dos clients no estan assignats a cap central, no farà res.

Buida

- **Paràmetres:** `int idx_central`
- **Condició d'aplicabilitat**: *idx_central* ha de ser un índex de central vàlid. Dins de la pròpia funció només s'aplicarà el canvi si s'aconsegueix assignar a tots els clients a noves centrals.
- **Funció de transformació**: Buida una central i assigna tots els clients que tenia a altres centrals que ja estiguin en marxa.

1.4 Heurística

h = - (benefici + (numClientsServits()(r[0] + r[1] + r[2] + 1))/(energiaPerduda));*

$$h = -benefici - \frac{clientsAssignats * \frac{ProduccióCentralsEnceses}{NombreCentralsEnceses}}{EnergiaPerduda}$$

Per calcular l'heurística tindrem en compte quatre paràmetres:

- El **benefici**: Pren valors propers al milió. És el valor més important de l'heurística perquè l'objectiu del problema serà maximitzar aquest valor. El problema d'utilitzar només el benefici és que en la majoria de situacions hi haurà molts estats i molts operadors que ens oferiran el mateix benefici però només podem aplicar un d'aquests operadors. Arribem a situacions en les que aplicant un únic paràmetre no és possible modificar el benefici. El benefici només podrà variar si assignem a un client que abans no tenia central a una central, si el desassignem, si obrim o si tanquem una central. Però hi ha moments en què no podrem aconseguir aquests canvis de benefici únicament aplicant un operador. La resta de paràmetres de la funció són els que ens guiaran a escollir aquell nou estat que de cara al futur pugui ser més beneficiós de manera que no ens quedem atrapats en un pas intermedi cap a una solució millor.

- L'**energia perduda**: Pren valors entre 0 i els milers. Es tracta de l'energia que es perd en el transport de l'electricitat d'una central cap a un client. Com més a prop estiguin els clients de les centrals que els subministren, més disminuirà l'energia perduda. D'aquesta forma evitem que es desaprofiti gran part de la producció de les centrals en generar energia extra que es perdrà pel camí. Aquest paràmetre ajudarà a fer intercanvis de clients a noves centrals de manera que, tot i no canviar el benefici, deixaran una central més buida i disminuirem l'energia perduda. D'aquesta forma anirem disminuint l'energia perduda fins que puguem arribar a un esta en què es pugui buidar una central i així estalviar-nos el cost de posada en marxa (sempre superior al cost de parada).

- Els **clients assignats** a alguna central: Pren valors menors al nombre de clients (en la majoria de proves tenim mil clients. Per tant, valors inferiors a mil). Volem maximitzar el nombre de clients assignats a centrals. D'aquesta forma evitem pagar les indemnitzacions i fem que els clients ens paguin a nosaltres per l'energia que els proporcionem.

- El **nombre i la producció de centrals enceses** en funció del tipus: Les centrals enceses solen ser entre 20 i 40 (en els casos en què el nombre de centrals és 40). La producció de les centrals enceses, serà el sumatori de l'energia que poden produir totes les centrals que es troben enceses. La producció dividida entre el nombre de centrals en funcionament (mitjana de producció de les centrals enceses) també és una fracció propera als 20. La idea serà disminuir el nombre de centrals enceses (ens estalviem els costos de producció) o augmentar la producció total d'aquestes. És a dir, tot i que no augmenti el nombre de centrals parades, es prioritzarà buidar les que produeixen menys. Ens vam adonar que en tots els costos de producció hi havia una part fixa que havien de pagar totes les centrals d'aquell tipus sense importar la seva producció i després hi havia una part variable en funció de la producció. Com que el cost fix l'haurem de pagar en tots els casos ens interessa que només els paguem en els casos en què després puguem treure més profit. És a dir, que preferim pagar el cost de posada en marxa per a les centrals de cada tipus que més produeixin més ja que després podrem subministrar a més clients perquè la central pot produir més.

En la fórmula volem maximitzar el benefici, el nombre de clients assignats i volem minimitzar l'energia perduda i el nombre de centrals enceses.

Hi ha diverses situacions que poden portar-nos a situacions que no estiguin maximitzant el benefici. Per exemple, com volem minimitzar l'energia perduda el programa podria intentar no assignar a cap client i d'aquesta forma l'energia perduda en el transport seria 0 perquè no es transporta energia. Una altra possibilitat seria que, per tal de minimitzar el nombre de centrals enceses, el programa deixés de subministrar a tots els clients no garantits. O per exemple, que per tal de subministrar energia a tots els clients comenci a encendre centrals que acabin repercutint en un benefici més negatiu.

Per tant, tots aquests paràmetres s'han de posar en un equilibri que impedeixi que per tal de millorar una mica un paràmetre empitjorem molt un altre. Vam provar amb altres heurístiques que també pensàvem que podien tenir sentit i que ens acostarien a la solució més òptima. També vam provar a variar els pesos dels diferents paràmetres i vam concloure que el que millor ens estava funcionant era l'heurística explicada anteriorment.

a) - benefici

b) - benefici - 1000*SumaExcedents

c) - benefici + 100*energiaPerduda - 100*numclientsServits

d) - benefici + numCentralsEnceses (amb pesos: 1500 A, 5000 B, 15000 C)

Vam començar amb l'heurística a) que només tenia en compte el benefici perquè realment era l'únic que ens demanava l'enunciat del problema. Com ja hem comentat aquesta solució donava problemes perquè en els casos en què l'heurística no variava el programa parava i no era conscient que aplicant més d'un operador podria arribar a una solució encara millor.

L'opció b) també tenia en compte els excedents de les centrals enceses i els intentava minimitzar. D'aquesta forma podia tractar de buidar tota una central encesa per minimitzar els excedents de la resta i així aconseguirem buidar una central, fet que repercutirà positivament al benefici. També vam provar d'assignar diversos pesos als excedents de les centrals.

L'heurística c) buscava minimitzar l'energia perduda pel camí perquè vam veure que com menys energia es perdés més augmentaria la producció excedent de les centrals i les centrals que quedessin enceses podrien subministrar a més clients de manera que es buidin centrals i es puguin apagar. Per compensar l'heurística i evitar que comencés a desassignar als no garantits amb la finalitat de reduir l'energia perduda total, vam afegir un potenciador del nombre de clients assignats per intentar maximitzar-ho. També vam provar diversos pesos.

En el cas d) teníem en compte quantes centrals de cada tipus teníem enceses, prioritzant sempre les centrals que més produeixen, que són les del tipus A i que tenen un cost de parada més elevat. Creiem que seria millor tenir buides més central petites (tipus B i C) que ens ofereixen menys producció i que el seu cost de parada no és tan elevat com el de les centrals tipus A.

1.5 Solucions Inicials

El factor en el que ens hem centrat més a l'hora d'idear les solucions inicials ha estat minimitzar la pèrdua d'energia.

Opció 1. Solució inicial ordenada.

En aquesta opció posem els clients amb més demanda a les centrals que més produeixen, restringint per la pèrdua d'energia. Vam tenir en compte l'espai que tenen disponible les centrals, de manera que les més grans estiguessin més plenes.

El primer que fem és ordenar els clients en funció del consum, de més gran al més petit i després les centrals, de les que més produeixen a les que menys. Un cop ordenat tot, mirem si l'energia perduda en el transport de la central al client sigui inferior a l'energia contractada pel client, en cas de no complir aquesta condició, quan haguem fet aquesta comprovació per tots els clients, l'assignem a la central lliure més gran.

La solució, però, pot assignar client a centrals que estan més lluny que altres i resultaria en una gran pèrdua d'energia.

Opció 2. Aleatòria.

La segona opció que hem pensat és agafar els client de manera aleatòria i anar assignant-los a la central que tenen més a prop.

En aquest cas, li donem més importància a la proximitat de les centrals als clients que al que consumeixen. Però hi afegim un component d'aleatorietat, ja que no assignarem primer els clients que més consumeixen.

El primer que fem és posar tots els clients garantits a la central lliure més propera en ordre aleatori i després assignar els que no són garantits de la mateixa manera.

Aquesta forma d'organitzar els clients i les centrals pot donar lloc a l'assignació de clients que estan molt lluny de la central més propera, de manera que li treu lloc a potencials clients que puguin estar més a prop, i amb els quals es podria aconseguir més benefici.

Opció 3. Greedy.

Assignar primer els clients que tenen la central més propera a menys distància.

Aquesta opció no només té en compte la central més propera a algun client, sinó que assignem primer els que estan més a prop de la central més propera. D'aquesta manera, primer tindrem en compte els clients que aprofiten tot el que demanen a la central i els últims, com que ja estaven a més distància, resulta més irrellevant si es perd més energia de la que ja es perdia amb la central més propera.

Per a arribar a aquesta solució inicial el primer que fem és calcular les distàncies a les centrals més properes de cada client. Un cop calculat, assignem primer els clients que tenen menor distància a la corresponent central, i si la central està plena, ens esperarem a assignar tota la resta de centrals per tornar a fer el mateix càlcul.

1.6 Com solucionar-ho?

Primerament cal analitzar el problema per veure quina és la millor forma de tractar-lo. El que ens estan demanant és que resolguem el problema per a un dia concret i que donem una solució. Aquesta solució haurà de determinar per a cada central quins clients té assignats. L'objectiu del problema és que retornem una solució vàlida que garanteixi totes les restriccions i aquesta solució ha de donar-nos el màxim benefici possible.

En el nostre cas no ens importa la seqüència de passos o d'operadors que ens porten de la solució inicial a la final, només ens interessa quina és aquesta solució final a la que arribem.

La possibilitat de trobar la millor solució s'escapa de les nostres possibilitats, ja que si haguéssim d'analitzar totes les possibles solucions fins a trobar la millor no acabaríem mai, pel fet que l'espai de solucions és massa ampli. Suposem un hipotètic cas en el que tenim 40 centrals i 1.000 clients. Les solucions possibles serien per a totes les centrals un vector de 1.000 bits indicant si la central ha de subministrar (bit a 1) o no (bit a 0) a aquell client. Per a cada central el nombre de combinacions de clients als que ha de subministrar seria 2^{1000} i si tenim 40 centrals la mida de l'espai de solucions seria 40* 2^{1000}. Aquest nombre de solucions (tot i que no totes són vàlides i hi hauria formes de

podar el problema) està fora de les nostres possibilitats. Ens conformem amb trobar una solució que sabrem que, tot i que potser no és l'òptima, maximitza molt el benefici.

Per tant la cerca local és molt bona opció, ja que podem crear una solució inicial de forma senzilla i anar passant a nous estats aplicant els operadors, de manera que la funció heurística ens indiqui com és de bona la solució, i vagi augmentant el benefici fins a arribar a un punt on ja no trobem cap solució millor.

2 Experimentació

2.1 Experiment 1

Utilitzar l'algorisme de Hill Climbing per determinar quin conjunt d'operadors dóna millor resultat per a la funció heurística i fixar els operadors.

La configuració inicial serà:

Centrals:

Tipus de centrals: 5 A, 10 B, 25 C.

Clients:

Número de clients: 1000.

Proporció de tipus de clients: 25% XG, 30% MG, 45% G.

Proporció de clients garantitzats: 75%.

Tenim els següents operadors: Assignar (A), Eliminar (E), Swap (S), Buidar (B).

Observació

Depenent dels operadors i les combinacions que utilitzem, podem obtenir millors o pitjors resultats.

Plantejament

Provarem diferents combinacions d'operadors.

Hipòtesi

A + E + S millors resultats i no caldrà B, ja que es podria fer implícitament amb A i E

Mètode

Anem provant les diferents combinacions d'operadors amb la mateixa seed, la mateixa configuració inicial i la solució inicial Greedy.

Utilitzarem l'algorisme Hill Climbing.

Mesurarem el temps, els nodes expandits, el benefici i el factor de ramificació.

Resultats

N: nombre de clients, M: nombre de centrals

Op. utilitzats	Temps (s)	Nodes expandits	Benefici (€)	Factor de ramificació
A	0	1	1.387.154,40	N
A + E	0	1	1.387.154,40	2N
B	0	1	1.387.154,40	M
S	67	305	1.440.810,90	N*(N-1)/2
S + A	58	264	1.458.580,90	N*(N-1)/2 + N
S + A + E	76	264	1.458.580,90	N*(N-1)/2 + 2N
S + B	76	313	1.474.205,40	N*(N-1)/2 + M
S + A + B	58	274	1.475.169,90	N*(N-1)/2 + N + M
S + A + E + B	87	274	1.475.169,90	N*(N-1)/2 + 2N + M

Conclusions

Com podem veure, les primeres tres combinacions d'operadors no ens afecten al benefici inicial. Amb les següents combinacions ja comencem a veure una millora i podem observar que l'operador d'Eliminar no l'utilitza, ja que el benefici amb ell és el mateix que sense ell. Amb això descartem la nostra hipòtesi inicial.

Per la resta d'experiments ens quedarem amb els operadors Swap, Assignar i Buidar, ja que, tot i tenir un factor de ramificació bastant elevat, és una de les combinacions que ens dóna millor benefici, a més de ser de les més ràpides.

2.2 Experiment 2

Utilitzar l'algorisme de Hill Climbing per determinar quina estratègia de generació de la solució inicial dóna millors resultats per a la funció heurística utilitzada a l'experiment 1 i fixar la solució inicial.

Observació

Depenent de la solució inicial que utilitzem, podem obtenir millors o pitjors resultats.

Plantejament

Considerarem les 3 solucions inicials que hem definit a dalt i mirarem si hi ha diferències a la hora de trobar solucions.

Hipòtesi

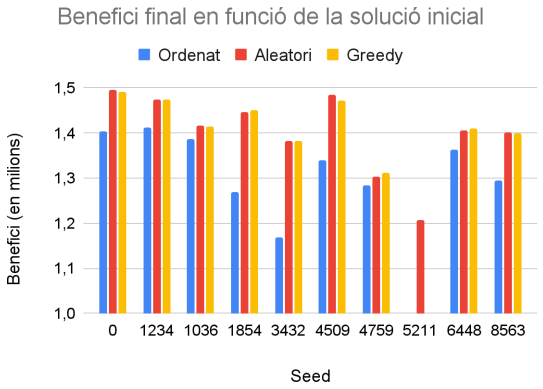
L'opció Greedy és la que genera millors solucions.

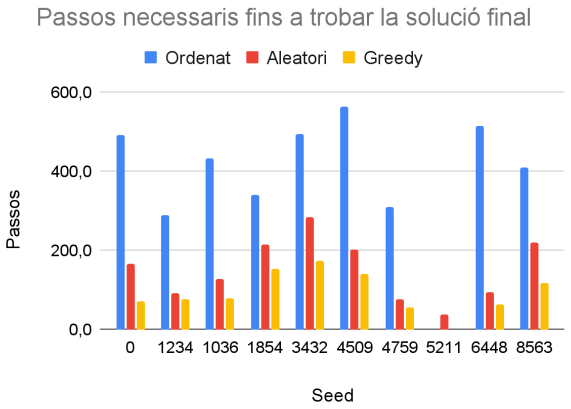
Mètode

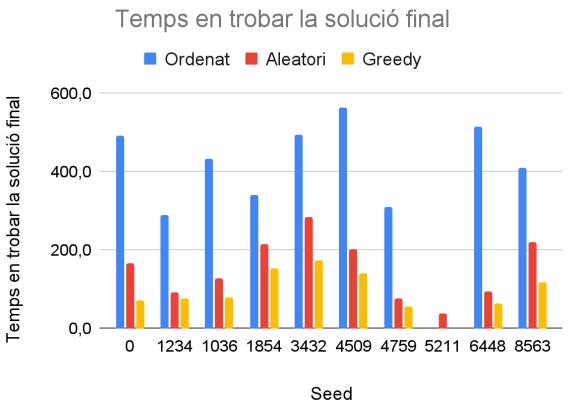
Per això mesurarem el benefici final, els passos que triga fins a arribar a aquest i el temps que triga en executar-se.

Per a que les 3 estratègies tinguin les mateixes possibilitats mesurarem 10 "seeds" diferents i per la estratègia aleatoria executarem tres vegades el codi per cada "seed" i calcularem la mitjana de les tres.

Resultats







Per la "seed" 5211 la solució inicial aleatòria era la única capaç de trobar una solució vàlida, les altres dos començaven des d'un estat no solució. (Veure *Figura 1* annex per valors concrets)

Conclusions

En aquests 3 gràfics podem veure millor les dades de la taula anterior. S'observa que, tot i que la solució basada en ordenar és sempre molt pitjor, les altres dues acaben aconseguint gairebé el mateix benefici sempre, però la solució greedy en un menor nombre de passos i temps. Això es pot explicar assumint que la solució greedy comença més a prop de la solució final. D'aquesta forma, confirmem la nostra hipòtesi inicial.

Per la resta d'experiments utilitzarem la solució inicial greedy ja que, encara que la aleatòria acaba trobant normalment un benefici lleugerament superior, és la que menys temps i passos necessita per acabar, i no depèn del factor aleatori.

2.3 Experiment 3

Utilitzar l'algorisme de Simulated Annealing amb la mateixa configuració per determinar quins són els paràmetres amb millors resultats.

L'algorisme de Simulated Annealing busca evitar quedar-se en un mínim local, acceptant solucions pitjors a l'inici i poc a poc anant acceptant cada vegada menys solucions pitjors. Per fer això existeixen els valors de k i λ, que descriuen la funció d'acceptació, el nombre d'iteracions totals i les iteracions per cada canvi de temperatura.

La funció que determina la probabilitat d'acceptació d'un estat pitjor és: $P = e^{\frac{\Delta E}{k \cdot e^{-\lambda \cdot T}}}$ on T és la iteració actual (Temperatura) i ΔE la diferència de l'heurística de l'estat anterior i el nou (Diferència d'energia). Podem adaptar els valors de k i λ per permetre moure'ns al principi de l'espai de solucions, acceptant-ne de pitjors i intentar sortir d'un possible mínim local.

Observació

Depenent dels paràmetres del Simulated Annealing, podem obtenir millors o pitjors resultats.

Plantejament

Escollim diferents valors k , λ , nombre d'iteracions per pas i observem les solucions.

Hipòtesi

El Hill Climbing acaba en un mínim local. El Simulated Annealing ens donarà una solució més bona.

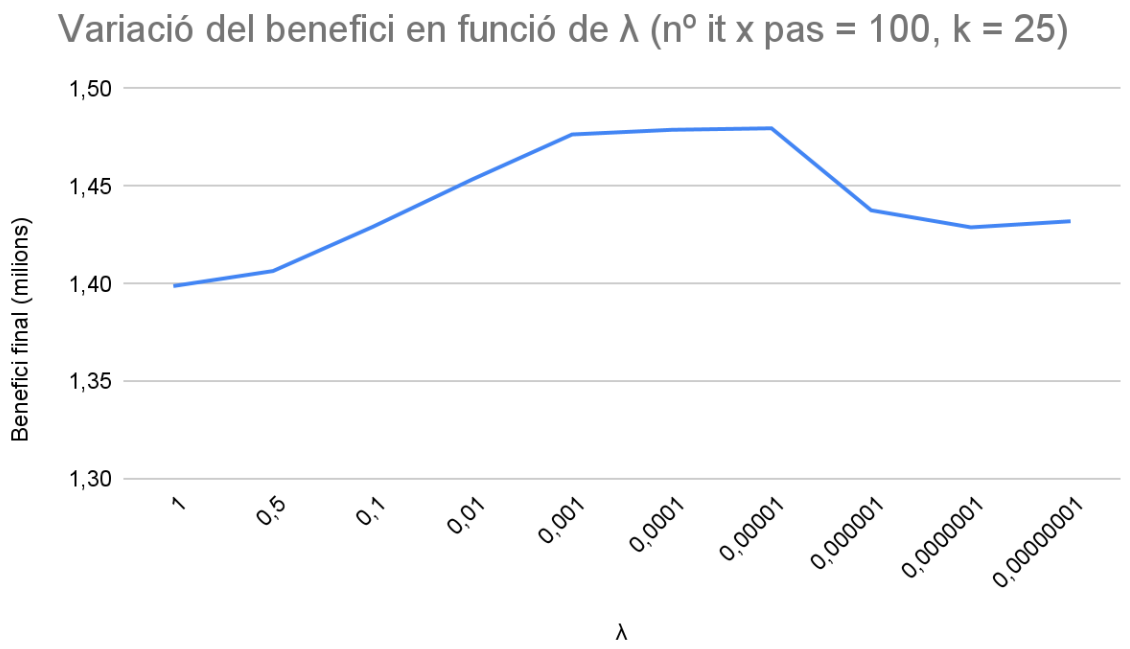
Mètode

Provarem diferents combinacions de valors de k, λ i iteracions per pas que trobin una solució final millor que el Hill Climbing.

Per trobar aquests valors només ho podem fer experimentalment. Utilitzarem un nombre d'iteracions bastant gran per assegurar-nos que acaba convergint.

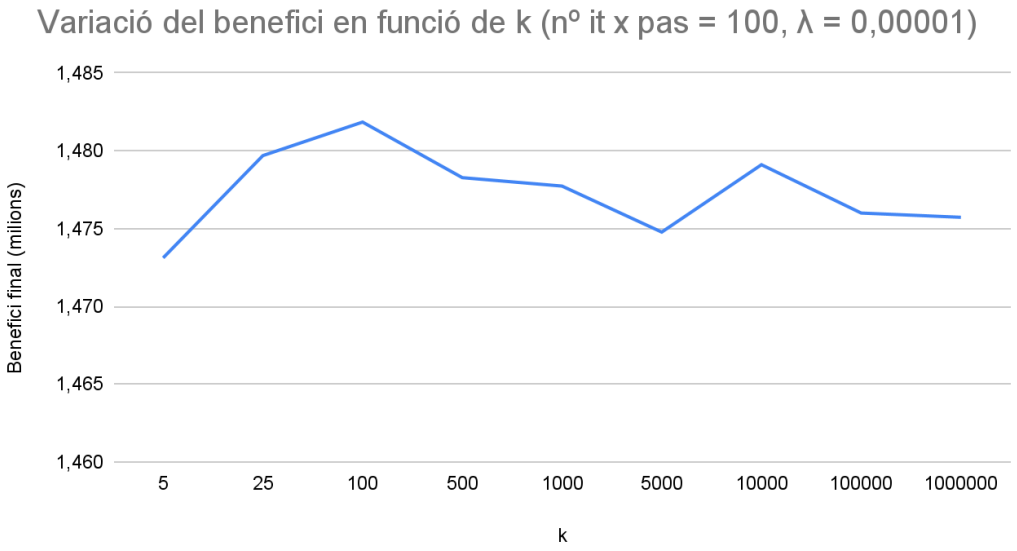
Resultats

Comencem per buscar el valor òptim de λ. Fixarem k a 25 i el nombre d'iteracions per pas de temperatura a 100.



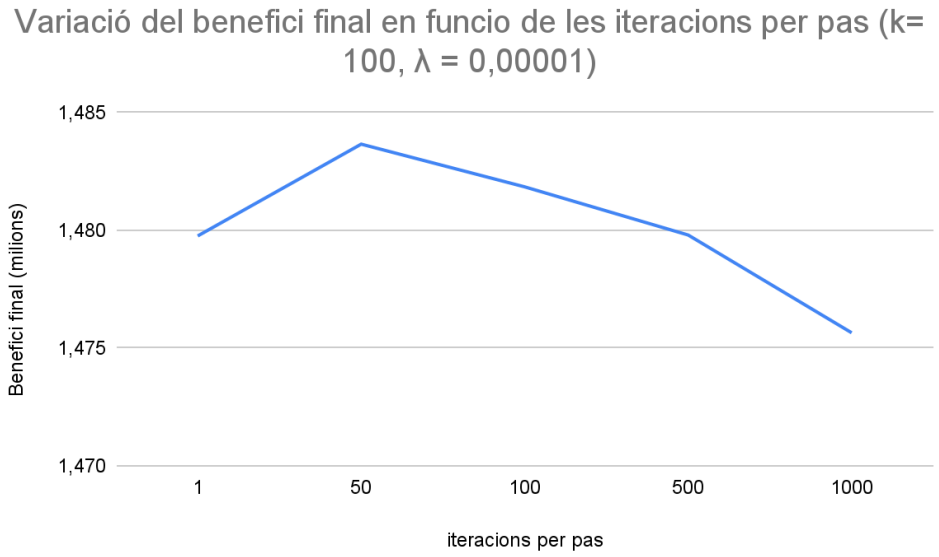
En aquest gràfic podem veure la variació del benefici final en funció de λ. Com més disminuim aquest valor durant més temps s'accepten estats pitjors, i per tant més temps triga en trobar la solució final. Veiem que el valor de λ òptim és 0,00001, que és el que dóna un millor resultat. (Veure annex pels valors concrets)

Seguidament buscarem el valor òptim de k. Fixarem λ al valor que acabem d'obtenir, és a dir, λ = 0,00001 i el nombre d'iteracions per pas a 100.



Aquí veiem la variació del benefici final en funció del valor de k. Com més augmentem el valor de k més endarrerim el punt on la funció d'acceptació arriba a 0 (on ja només acceptarem estats millors), acceptant durant més temps estats pitjors. Observem que el valor òptim de k es troba al voltant de 100.

Ara intentarem trobar el valor òptim del nombre d'iteracions per pas. Fixarem k i λ als valors que hem trobat abans.



En aquest gràfic podem veure clarament com el valor òptim pel nombre d'iteracions és 50, ja que és el punt més alt de tots els valors que hem provat.

Finalment intentarem acabar d'ajustar els valors dels 3 paràmetres per aquesta solució inicial, plantejament i heurística, buscant al voltant dels valors que hem trobat anteriorment.

Nº iteracions	**Nº it per pas**	k	λ	**Benefici final**
50.000.000	50	100	0,00001	1.483.655,9
50.000.000	100	50	0,00001	1.481.255,9
50.000.000	100	150	0,00001	1.481.394,9
50.000.000	50	100	0,00005	1.481.365,4
50.000.000	50	50	0,00001	1.481.720,4

Acabem veient que modificant algun dels paràmetres, el benefici final empitjora, per tant ens quedem amb els que ja teníem: k = 100; λ = 0,00001; nº it per pas = 50.

Conclusions

La configuració que ens dóna millors resultats és k = 100; λ = 0,00001; nº it per pas = 50; amb els que obtenim un benefici final de 1483655,9 €. A més, confirmem la nostra hipòtesi inicial, ja que obtenim un benefici major que amb el Hill Climbing.

2.4 Experiment 4

Utilitzar l'algorisme de Hill Climbing per estudiar el temps d'execució per trobar la solució, incrementant els següents paràmetres:

- Número de centrals, amb les mateixes proporcions. Per 1000 clients, anar augmentant de 40 en 40 el nombre de centrals.
- Número de clients, amb les mateixes proporcions. Per 40 centrals, anar incrementant de 250 en 250 el nombre de clients.*

*A l'enunciat se'ns indica que hem d'augmentar el nombre de clients de 500 en 500 però a l'hora de fer els experiments, a partir dels 1500 ja no ens donava resultats, per això, vam decidir augmentar de 250 en 250 per poder recollir més dades

Observació

A mesura que afegim centrals i clients, augmenta el factor de ramificació i, per tant, hauria d'augmentar el temps d'execució.

Plantejament

Anem augmentant el nombre de centrals o de clients per veure el temps que es necessita.

Hipòtesi

El temps d'execució anirà augmentant a mesura que afegim clients.
El temps d'execució anirà augmentant a mesura que afegim centrals.

Mètode

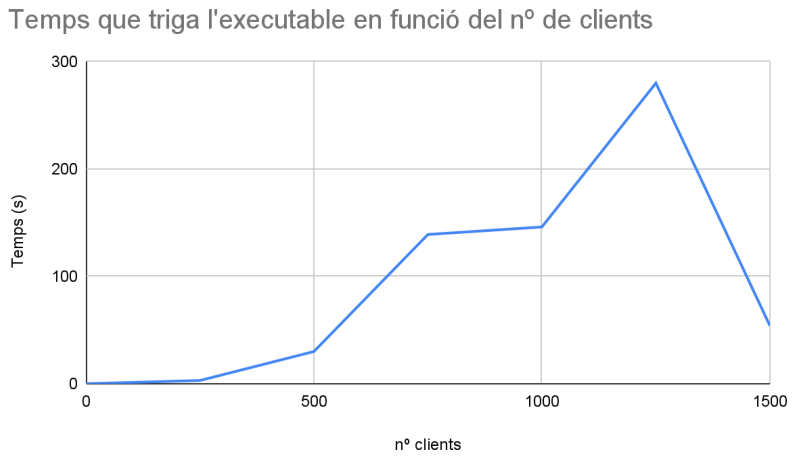
Anem augmentant el nombre de clients de 250 en 250.

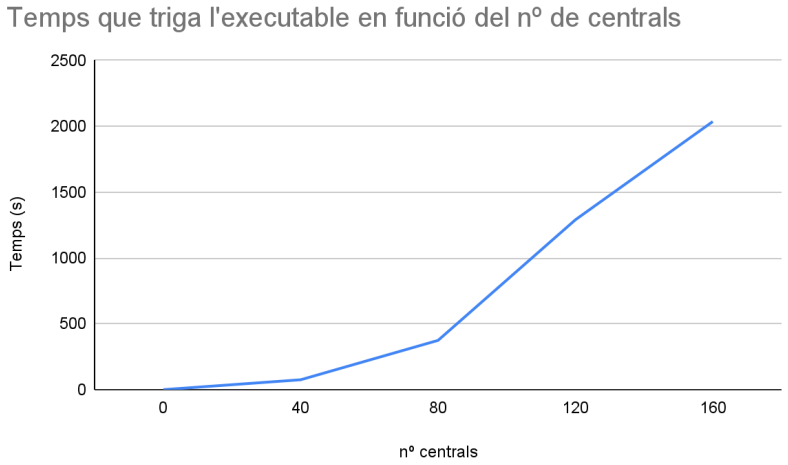
Utilitzarem l'algorisme de Hill Climbing

Mesurarem el temps que triga a trobar cada solució

Fem el mateix amb el nombre de centrals i anem augmentant de 40 en 40.

Resultats





Conclusions

Per la part dels clients, vam haver de provar diferents seeds ja que moltes no ens donaven una solució vàlida quan arribàvem a 1500 clients. Fins que vam trobar la seed 1309 que ens donava una solució. De fet, és l'única seed que vam trobar :'c

A la gràfica dels clients veiem com augmenta el temps a mesura que augmenten els clients, fins que arribem als 1500 clients, que fa una baixada molt dràstica. Per tant, no podem confirmar la nostra hipòtesi inicial. Analitzant l'execució dels 1500 clients es veu com els operadors que s'apliquen molt rarament és el de *buidar*, que té un temps d'execució superior a la resta per tota la feina que ha de realitzar. Tot i que expandeix més nodes que les execucions amb menys clients, com que els operadors que

compleixen les condicions d'aplicabilitat són força ràpids ens permet expandir nodes més ràpidament.

Podem observar clarament com augmenta el temps a mesura que augmentem el nombre de centrals. Per tant, podem confirmar la nostra hipòtesi inicial.

2.5 Experiment 5

Utilitzar l'algorisme de Hill Climbing i Simulated Annealing per trobar un rang de penalització d'una heurística, que partint d'una solució inicial buida, ens doni una solució vàlida.

Observació

A l'heurística no tenim en compte els clients garantitzats. Una penalització per no tenir clients garantits assignats ens podria estalviar comprovacions en altres parts del codi.

Plantejament

Escollim diferents penalitzacions per veure el rang per el qual ens dóna solucions vàlides.

Hipòtesi

Existeix un rang que ens assegura una solució vàlida.

Mètode

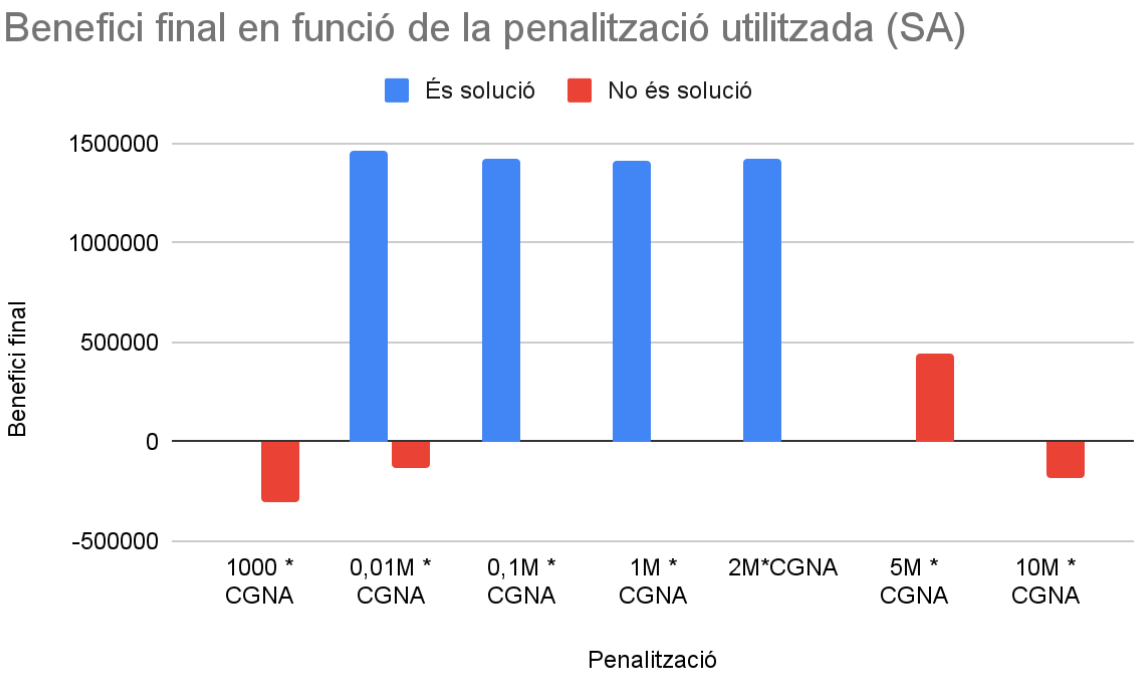
Prendrem com a solució inicial una en la que cap client està assignat i la mateixa heurística que abans amb un sumand més (la penalització), que serà de la forma K*CGNA.

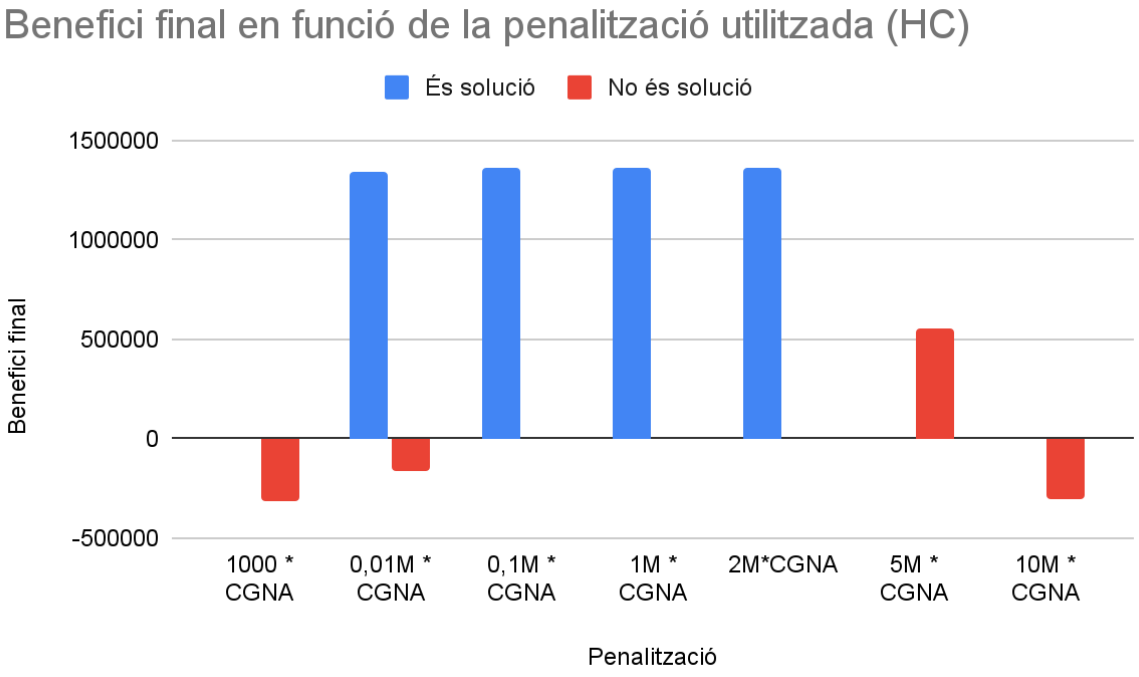
CGNA = clients garantits no assignats.

Utilitzarem 3 "seeds" diferents i agafarem la mitjana.

Mesurarem i compararem els diferents beneficis finals i si són una solució vàlida.

Resultats


Benefici final en funció de la penalització utilitzada (SA)
És solució
No és solució
1500000
1000000
500000
0
-500000
Benefici final
1000 * CGNA
0,01M * CGNA
0,1M * CGNA
1M * CGNA
2M*CGNA
5M * CGNA
10M * CGNA
Penalització


Benefici final en funció de la penalització utilitzada (HC)
És solució
No és solució
1500000
1000000
500000
0
-500000
Benefici final
1000 * CGNA
0,01M * CGNA
0,1M * CGNA
1M * CGNA
2M*CGNA
5M * CGNA
10M * CGNA
Penalització

Conclusions

Podem veure que el rang de 0,1M - 2M ens genera en totes les seeds solucions vàlides amb el mateix benefici final. I per tant, comprovem la nostra hipòtesi inicial. Pel valor 0,01M hem vist que en una de les 3 “seeds” que hem utilitzat acaba en una solució no vàlida, per tant no la inclourem dins del rang.

Amb una penalització petita no acaba en una solució vàlida perquè no té prou pes respecte els altres valors de l’heurística. En el cas de les penalitzacions molt grans creiem que es deu a l’arrodoniment que es fa amb els doubles, és a dir, 10000000 i 10000001 s’acaben arrodonint al mateix valor, i per tant el valor de l’heurística no canvia entre estats i com que no troba un valor millor (son tots iguals) acaba l’execució.

2.6 Experiment 6

Utilitzar l'algorisme de Hill Climbing i Simulated Annealing per veure si duplicant i triplicant les centrals de tipus C, disminueix l'ús de les A i B.

Observació

Utilitzant diferents proporcions, podrem assignar els clients de diferents formes i per tant, potser trobar una configuració que ens doni millor benefici. Els clients segurament tindran una central C més a prop i no perdrem tanta energia anant a una central més gran però més llunyana.

Plantejament

Mirarem diferents proporcions i analitzarem quines ens donen millors resultats.

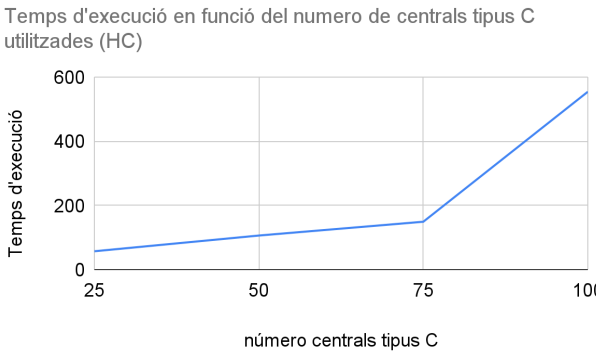
Hipòtesi

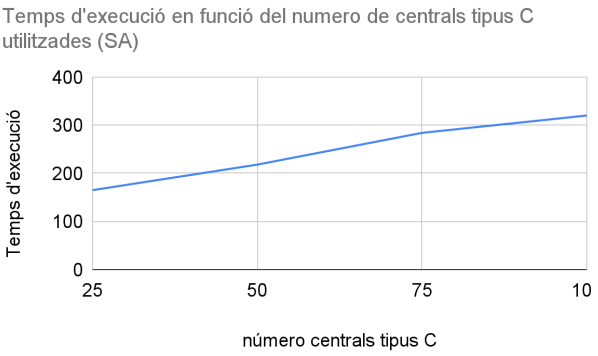
Si augmentem el nombre de centrals de tipus C, s'utilitzaran més que les de tipus A i B.

Mètode

Utilitzarem els algorismes de Hill Climbing i Simulated Annealing. Deixarem el nombre de centrals de tipus A i B com al principi, A = 5, B = 10, i anirem agafant valors entre el 25 i 100 per les centrals tipus C.

Resultats





En aquests dos gràfics podem veure que pel cas de Hill Climbing el temps d'execució augmenta considerablement al augmentar el nombre de centrals de tipus C. Això es pot deure a que aplicarà o intentarà aplicar l'operador Buidar a moltes centrals petites (tipus C), que és bastant lent. En canvi en el cas de Simulated Annealing veiem que el temps augmenta de forma bastant menys significativa, gràcies a que aquest mètode no aplica tots els operadors possibles a cada estat.

		Num centrals utilitzades		
HC/SA	**nº centrals tipus C**	A	B	C
HC	25	5	10	19
HC	50	3	9	42
HC	75	0	8	66
HC	100	0	4	77
SA	25	5	10	20
SA	50	4	8	30
SA	75	4	7	28
SA	100	3	8	36

En aquesta taula podem veure el nombre de centrals utilitzades a la solució en funció del nombre de centrals de tipus C que hi ha. Recordem que sempre hi ha 5 centrals tipus A i 10 centrals tipus B.

Conclusions

Si mirem els resultats de la taula veiem que sí que redueix el nombre de centrals A i B quan augmentem el nombre de centrals C. En el cas de Hill Climbing les redueix molt significativament i en el cas del Simulated Annealing menys, però en els dos casos les redueix. De manera que podem confirmar la nostra hipòtesi inicial.

En el cas de Hill Climbing el temps d'execució augmenta significativament, ja que ha d'aplicar moltes més vegades l'operador *Buidar*, que és bastant lent.

3 Annex

Seed	Solució			Passos			Temps (s)		
	Ordenat	Aleatori	Greedy	Ordenat	Aleatori	Greedy	Ordenat	Aleatori	Greedy
0	1402834,6	1495187,2	1491633,1	824,0	261,3	187,0	492,0	165,3	71,0
1234	1413162,3	1475102,9	1475169,9	785,0	384,3	274,0	289,0	91,7	77,0
1036	1386361,1	1417508,9	1414637,4	783,0	228,7	158,0	433,0	127,7	79,0
1854	1270163,1	1447091,6	1451194,3	742,0	373,3	340,0	342,0	214,7	153,0
3432	1168966,3	1382863,8	1383058,8	737,0	319,3	220,0	495,0	284,7	174,0
4509	1338696,4	1485849,3	1472115,3	759,0	275,0	213,0	564,0	201,7	140,0
4759	1283694,2	1304209,0	1311119,2	751,0	450,7	393,0	309,0	76,0	57,0
5211	NO	1208151,1	NO	NO	398,0	NO	NO	37,3	NO
6448	1362124,3	1405677,5	1409699,2	775,0	323,7	260,0	516,0	94,7	65,0
8563	1295237,2	1402631,2	1400403,7	750,0	250,0	203,0	409,0	220,3	119,0

Figura 1: Taula amb el benefici final, els passos i el temps utilitzat per cada execució.

Nº iteracions	Nº it per pas	k	λ	Benefici final
1000000	100	25	1	1.398.883,4
1000000	100	25	0,5	1.406.593,9
1000000	100	25	0,1	1.429.227,4
1000000	100	25	0,01	1.453.553,4
1000000	100	25	0,001	1.476.548,4
5000000	100	25	0,0001	1.478.920,4
5000000	100	25	0,00001	1.479.688,4
5000000	100	25	0,000001	1.437.676,9
5000000	100	25	0,0000001	1.428.972,4
5000000	100	25	0,00000001	1.432.079,9
Nº iteracions	**Nº it per pas**	**k**	**λ**	**Benefici final**
10000000	100	1	0,00001	1.469.557,4
10000000	100	5	0,00001	1.473.116,9
50000000	100	25	0,00001	1.479.688,4
50000000	100	100	0,00001	1.481.842,9
50000000	100	500	0,00001	1.478.271,4

50000000	100	1000	0,00001	1.477.730,4
50000000	100	5000	0,00001	1.474.772,4
50000000	100	10000	0,00001	1.479.103,4
50000000	100	100000	0,00001	1.475.997,4
50000000	100	1000000	0,00001	1.475.723,4
Nº iteracions	**Nº it per pas**	**k**	**λ**	**Benefici final**
50000000	1	100	0,00001	1.479.758,9
50000000	50	100	0,00001	1.483.655,9
50000000	100	100	0,00001	1.481.842,9
50000000	500	100	0,00001	1.479.798,4
50000000	1000	100	0,00001	1.475.647,9

Figura 2: Taula amb els valors concrets per trobar els paràmetres del Simulated Annealing