UNIVERSITAT POLITÈCNICA DE CATALUNYA

Pràctica 1 - Cerca Local

[Intel·ligència Artificial]

Estudi i optimització d'algorismes de cerca per una empresa subministrament d'electricitat

---

Autors:

Pol Fradera
Jin Haonan
Alex Wickenden

2022-2023

Índex

1 Introducció

Aquesta pràctica té l'objectiu de resoldre un problema de distribució de clients en centrals elèctriques per obtenir el màxim benefici. Per resoldre el problema, implementarem una intel·ligència artificial basada en l'estratègia de cerca local.

Aquesta cerca local es farà sobre l'espai de solucions del problema i ens desplaçarem amb un conjunt d'operadors segons el criteri d'una funció de qualitat.

Per trobar la millor estratègia de resolució d'aquest problema, plantejarem un conjunt d'experiments per determinar els operadors que utilitzarem, la solució inicial generada i els paràmetres de cerca.

2 Descripció del problema

2.1 Descripció general

Amb les noves regulacions europees pel mercat de l'energia sorgeixen noves oportunitats per optimitzar l'oferta i demanda. Cada empresa de generació d'electricitat gestiona un conjunt de centrals elèctriques de diferents tipus, que cada un permet generar una certa quantitat de megawatts diaris. Normalment, la capacitat de producció supera la possible demanda, per tant, s'ha de decidir quines centrals mantenir obertes i quines tancades per tal de maximitzar el benefici, tenint en compte que existeix un cost de posar en marxa i un de parada.

Pel que fa als consumidors, disposem de tres grups, els extra grans (XG), els molt grans (MG), i els grans (G). A més, per cada consumidor hi haurà un contracte que determinarà la prioritat en què ha de ser servit: servei garantit i servei no garantit. Els consumidors que tenen contracte amb servei garantit hauran de ser subministrats i els que tenen un contracte amb servei no garantit, si no se'ls subministra energia, se'ls haurà de pagar una indemnització en funció del seu consum.

2.2 Elements d'entrada

- El problema considera una àrea rectangular de 100x100 km^2.
- Existeixen C centrals ubicades cada una en unes coordenades (x,y) en quilòmetres dins de l'àrea de 100x100 km^2, cadascuna té un tipus, conté la producció que pot generar en un dia, el cost de posar en marxa i el de parada.
- Existeixen N clients ubicats cada un en unes coordenades (x,y) en quilòmetres dins de l'àrea de 100x100 km^2. Amb una prioritat p, que pot ser 0 si el client té un contracte amb servei garantit i 1 si no és garantit. Cada client té un tipus i la seva demanda del dia.

2.3 Càlcul dels elements del problema

- *Cost total marxa d'una central:* producció*cost de producció+cost de posar en marxa.
- *Pagament d'un client:* tarifa*demanda del dia.
- *Indemnització d'un client amb contracte de servei no garantit:* tarifa de la penalització*demanda del dia.
- *Pèrdua de l'electricitat subministrada segons la distància entre el client i la central:*

Distancia	Pérdida
Hasta 10 Km	0
Hasta 25 Km	10 %
Hasta 50 Km	20 %
Hasta 75 Km	40 %
Más de 75 Km	60 %

2.3 Objectiu

Volem resoldre el problema per un dia concret. Per això disposarem dels elements d'entrada mencionats anteriorment, que bàsicament és una llista de centrals i una altra de clients amb les seves dades pertinents.

2.4 Descripció de la solució

Una solució determinarà per cada central quins clients té assignats, l'objectiu és assignar els clients de manera que s'obtingui el benefici màxim.

2.4.1 Criteris de la solució

Per obtenir i avaluar la solució farem servir els següents criteris i restriccions:

1. No es pot assignar a una central més demanda de la que pot produir.
2. Si una central està en marxa, genera tota la seva producció, la que no s'assigna a cap client es perd.
3. Un client només s'assigna a una central.
4. Als clients se'ls serveix sempre completament, és a dir, no és possible fer una assignació si no es pot servir tota la demanda del client.
5. Tots els clients que han contractat servei garantit han de ser servits.
6. S'ha de maximitzar el benefici obtingut.

2.5 Justificació de l'ús de cerca local

Després d'analitzar el problema, ens hem adonat que no es tracta d'un problema que consisteixi en complir unes certes condicions o en trobar una solució vàlida, per resoldre el problema hem de buscar el màxim benefici.

Amb una bona funció heurística i uns operadors mitjanament descents ens permetrà arribar a una solució no tan llunyana de la millor possible dins l'espai de solucions.

Com que l'espai de cerca és inabordable de manera exhaustiva, necessitarem algorismes de cerca local com el Hill Climbing i el Simulated Annealing, que ens permetran limitar l'espai de cerca a un espai de solucions el qual serà totalment explorable. Per tots aquest motius hem conclós la necessitat de fer una búsqueda local.

3 Disseny del projecte

3.1 Representació de l'estat

La nostra classe estat conté els següents atributs:

- *Clientes:* Objecte de la classe Clients que conté un *ArrayList* de la classe client.
- *Centrales:* Objecte de la classe Centrals que conté un *ArrayList* de la classe centrals.
- *Represent: Array* d'*integers* que representa l'estat (les posicions representen els clients i el seu valor la central a la qual estan assignats).
- *produccion*: Array de doubles que conté la producció disponible de cada central.
- *central_activa*: *Array* d'*integers* que conté el nombre de clients que hi ha assignats a cada central.
- *beneficio*: *Double* que conté el benefici total.
- *produccion_disponible: Double* que conté la suma de les produccions disponibles de cada central.

El que realment representa l'estat és l'array Represent, que determina quina és la solució, de tal manera que indica per cada client la central assignada, si el client no té cap central assignada, aleshores té un índex de central no vàlid (-1).

3.2 Anàlisi de la mida de l'espai de cerca

Ara doncs, de manera bastant aproximada, explorarem quina és la mida de l'espai de solucions o de cerca. Els dos factors que intervenen en la representació d'una solució són les centrals i els clients. Si tenim només una central, el nombre possible de solucions és de !nombre_clients, si tenim un nombre de clients normal (1000 clients) com a la majoria d'experiments que s'han realitzat, el nombre de possibles solucions és de l'escala de 10^{2568}, un terabyte de memòria és de l'escala de 10^{12} , així doncs descartem la possibilitat d'explorar tan sols una fracció significativa de totes les possibles solucions. Si afegim més centrals al problema aquest nombre exponencialment i es fa encara més impossible d'explorar al complert.

3.3 Operadors

En un primer moment vam pensar d'utilitzar els següents operadors:

- AfegirClient (index_client, index_central): afegeix el client que correspon a index_client a la central amb index_central si el client no es troba a cap central i si la central pot proporcionar l'energia suficient.
- TreureClient (index_client): si el client es troba a una central i aquest no té contractat un servei garantitzat, el client deixa de tenir assignada la central.

Després de fer les experimentacions vam veure que amb l'algorisme Hill Climbing, l'operador TreureClient gairebé mai generava estats millors que els anteriors, ja que la funció de qualitat ha de maximitzar el benefici.

Aleshores, al no disposar de l'operador de TreureClient, per poder moure els clients per les diferents centrals i per tant, poder abarcar tot l'espai de solucions vam arribar a la conclusió que com a mínim necessitàvem un operador que permetés moure un client d'una central a una altre.

Per això, vam implementar aquests dos operadors:

- MoureClient(index_client, index_central): al client corresponen a l'index_client se li assigna la central amb index_central si aquesta pot servir la demanda del client.
- PermutarClients(index_client1, index_client2): el client corresponen a l'index_client1 se li assigna la central del client amb index_client2, i viceversa, si les dues centrals poden subministrar l'energia necessària al fer el canvi. Si cap dels dos clients està assignat a una central, no es fa cap canvi.

3.3.1 Anàlisi del factor de ramificació dels operadors

El factor de ramificació de l'operador mover és el nombre de clients multiplicat pel nombre de centrals i el de l'operador permutar és el nombre de clients al quadrat, ja que són tots els possibles estats que es poden generar sense tenir en compte les comprovacions.

Per tant, si combinem els operadors mover i permutar, la ramificació total que obtenim és Nclients*Ncentrals + $Nclients^2$, que és la suma dels dos factors de ramificació.

3.4 Solució inicial

En un principi, vam realitzar una solució inicial que consistia en anar assignant els clients amb servei garantit començant per la primera central i continuant per la següent quan no se li pogués assignar el client. El problema que ens vam trobar va ser, que en casos extrems no es podia assignar tot el conjunt de clients amb servei garantit amb la manera que ho fèiem.

Per tant, vam haver de realitzar un *backtracking* perquè anés provant distribucions de clients amb servei garantit fins que tots quedessin assignats. A aquesta solució inicial l'anomenarem solucióIni. Per provar una altra solució inicial, el que vam fer va ser, després d'afegir tots els clients amb servei garantit tal com fem a la *solucióIni*, afegir els clients amb servei no garantit possibles de manera iterativa. Aquesta solució inicial la vam descartar ràpid, ja que empitjorava notòriament la *solucióIni*.

Una altra solució inicial que hem implementat, que anomenem *solucióIni3*, consisteix a realitzar un *backtracking* per assignar tots els clients amb servei garantit, però a diferència de la *solucióIni*, assigna els clients de tal manera que primer hi ha un client a cada central, després dos i així successivament. D'aquesta manera, generalment a totes les centrals hi haurà espai per posteriorment assignar més clients amb els operadors.

3.5 Funcions heurístiques

Factors que intervenen en el problema:

- Cost de les centrals (en marxa i en aturada)
- Producció de les centrals
- Consum dels clients
- Tipus de contracte dels clients
- Distància entre client i central assignada
- Benefici total

Per a la primera funció heurística proposada no hem tingut en compte la producció disponible de les centrals, es basa a trobar el màxim benefici total.

En cap de les dues heurístiques hem tingut en compte els clients disponibles (sense una central assignada) ni el nombre de centrals en marxa, ja que no són un factor que tingui un impacte directe i que es comporti de manera uniforme, és a dir, no ens aporta cap millora real la seva consideració.

A la segona funció heurística, però, entra en l'equació el potencial benefici del total de les centrals (en forma de la producció disponible per vendre), és a dir, al càlcul de la funció heurística a part de buscar el màxim benefici també ens interessa la solució que té més producció per assignar.

Primera funció heurística

Plantegem el problema com a la maximització del benefici total de les centrals. Aquest és causat per diversos paràmetres com els costos de les centrals (engegades o no), els clients als quals no es pot subministrar (en forma d'indemnització) o el cobrament als clients subministrats. Aquesta opció ens garanteix que no es realitzarà cap pas que reduirà el benefici total.

Segona funció heurística

En aquest cas plantegem el problema d'una manera més complexa i no només busquem el benefici més alt sinó que també tenim en compte un altre paràmetre, el benefici potencial. Això ens permet evitar bastants casos d'estancament a punts sense sortida, el que es coneix en castellà com a meseta.

Un dels clars exemples de la necessitat d'aquesta variant és el cas en què tenim dos clients assignats cada un a una central i ambdós produeixen unes pèrdues del 60% (el màxim possible), pot ser que si fem un intercanvi dels clients (els canviem les centrals de l'un a l'altre) un o els dos tingui una reducció en el percentatge de pèrdues que suposa a la central i, per hi haurà més producció disponible per a vendre, cosa que pot comportar un major benefici.

4 Experimentació

Per tal de poder replicar els experiments, s'ha d'obrir el projecte amb IntelliJ o un IDE semblant i anar al document Main.java, allà s'han de fer els canvis pertinents per cada experiment, un cop acabada la configuració dels paràmetres s'ha d'executar el Main.java.

4.1 Entorn d'experimentació

Ordinador d'execució:

- *Processador:* Ryzen 7 4800H 2.9GHz
- *Memòria RAM:* 16GB
- *Sistema Operatiu:* Windows 10 Pro

Altres dades:

- *IDE:* IntelliJ IDEA Community Edition 2022.2.2
- *Memoria asignada:* 8192MB (menys en casos que s'especifica)
- *Càlcul del temps d'execució:* Llibreria de Java System (mitjançant la funció currentTimeMillis())

4.2 Experiment 1: Determinar el conjunt d'operadors

En aquest experiment hem de determinar quin conjunt d'operadors proporciona millors resultats amb un escenari en què el nombre de centrals de cada tipus és 5 (A), 10(B) i 25(C), el nombre de clients són 1000, tenen una proporció de 25% (XG), 30% (MG) i 45% (G) segons el tipus i una proporció del 75% amd subministrament garantit.

Observació	El conjunt d'operadors escollits per a generar successors pot influir en la solució final del Hill Climbing.
Plantejament	Tenim 2 conjunts d'operadors: - Moure - Swap i moure
Hipòtesi	Combinant diferents operadors arribarem a una millor solució.
Mètode	● Escollim 10 llavors aleatòries per a l'estudi.

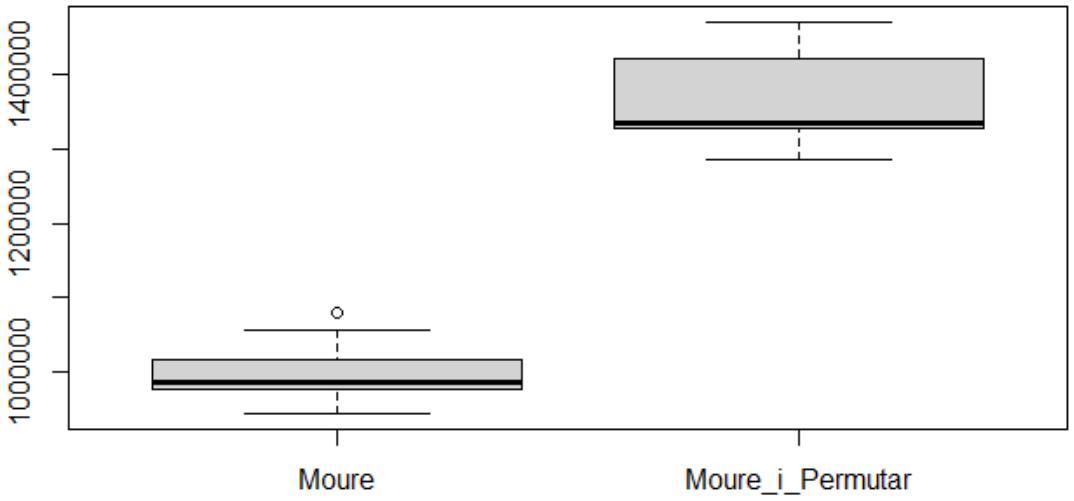
	• A cada llavor realitzarem 10 execucions per cada conjunt d'operadors. • Ho farem per amb la solucióIni. • Realitzarem l'estudi amb 1000 clients i 40 centrals. • Utilitzarem l'algorisme Hill Climbing. • Mesurarem el temps d'execució i anotarem el benefici per cada seed.

Seed	7907	3776	8108	1173	2903	3184	7247	5060	3332	777
Moure	394	381	593	398	687	533	477	494	616	397
Moure i permutar	266984	318116	345193	302471	309323	322532	297840	311500	332893	305424

Temps d'execució (ms)

Seed	7907	3776	8108	1173	2903	3184	7247	5060	3332	777
Moure	990281	979691	1009904	944759	979393	1016982	1055448	977439	958587	1080269
Moure i permutar	1322622	1331632	1393420	1284790	1332446	1420959	1442541	1334932	1328373	1470202

Benefici (euros)



Boxplot comparatiu del benefici final (euros)

Amb els resultats de l'experiment, hem pogut veure que el benefici que aconseguim amb la combinació de l'operador swap i moure és molt superior al que s'aconsegueix només amb l'operador moure.

Degut a aquesta diferència tan gran, ens quedem amb la combinació dels dos operadors, ja que s'aproxima més a complir l'objectiu i, per tant, es compleix la nostre hipòtesi. Com que hi ha una diferència tan gran en l'optimitat de la solució, no ens fixem gaire en la diferència de temps d'execució, ja que l'execució amb els operadors moure i swap expandeix molts més nodes per trobar una solució més òptima, per tant és previsible que trigui més.

4.3 Experiment 2: Determinar la solució inicial

En aquest experiment hem de determinar quina estratègia de solució inicial proporciona millors resultats amb l'escenari de l'apartat anterior i utilitzant l'algorisme Hill Climbing.

Observació	Pot ser que hi hagi mètodes d'implementació que obtenen millor solució.
Plantejament	Escollim diferents mètodes d'implementació i analitzem els resultats.
Hipòtesi	Tots els mètodes són iguals (H0) o n'hi ha de millors.
Mètode	● Escollim 10 llavors aleatòries per a l'estudi. ● A cada llavor realitzarem 10 execucions per cada solució inicial. ● Realitzarem l'estudi amb 1000 clients i 40 centrals. ● Utilitzem l'algorisme Hill Climbing. ● Utilitzem el conjunt d'operadors escollits a l'experiment anterior (mover i permutar).

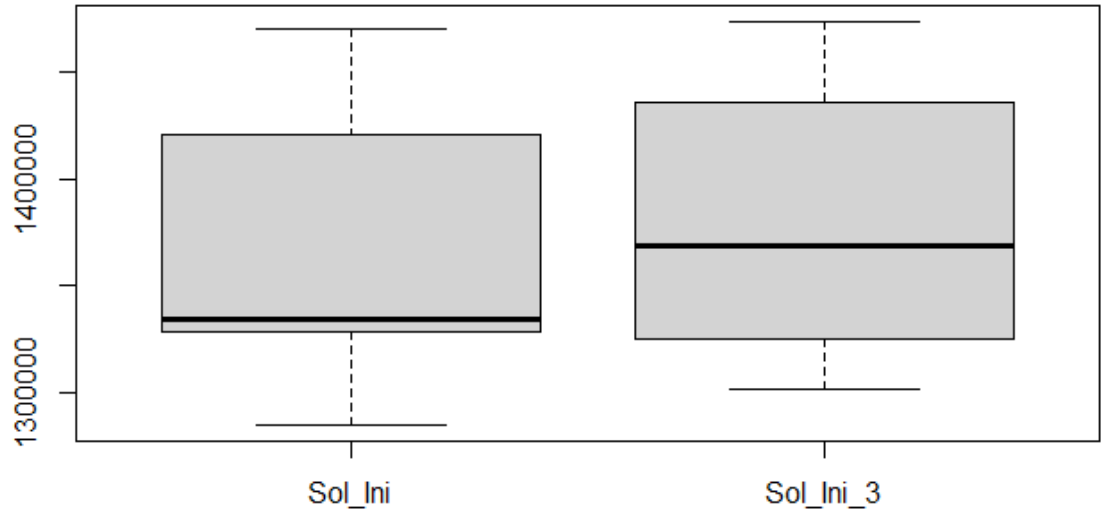
Les dos estratègies que compararem són les següents:

-SolucionIni(): Realitzem un *backtracking* per assignar tots els clients amb servei garantit a les centrals.

-SolucionIni3(): Es fa un *backtracking* però s'assignen els clients de manera que hi hagi un a cada central, després dos, etc.

Seed	7907	3776	8108	1173	2903	3184	7247	5060	3332	777
Sol Ini 3	1388349	1357842	1324611	1333059	1379224	1435745	1454532	1324762	1300864	1473579
Sol Ini	1322622	1331632	1393420	1284790	1332446	1420959	1442541	1334932	1328373	1470202

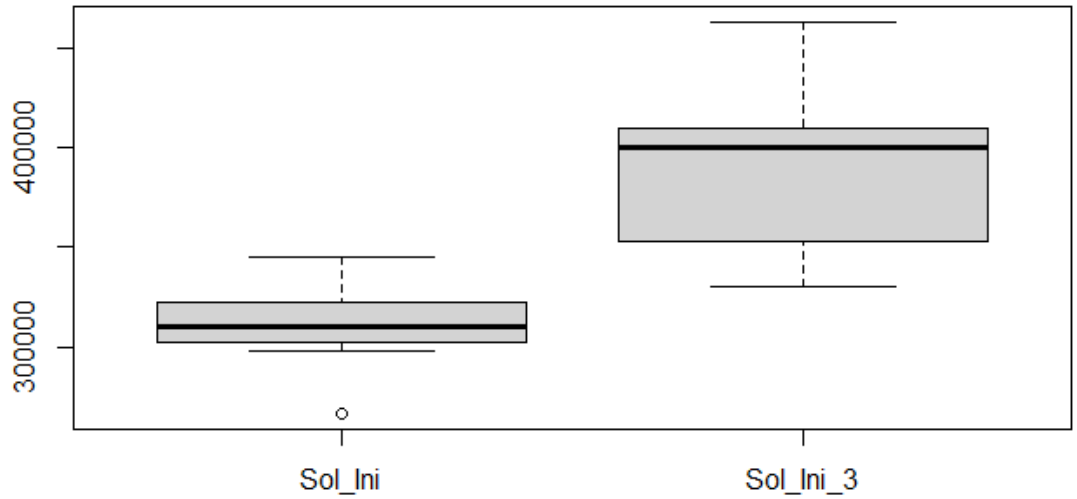
Benefici (euros)



Boxplot comparatiu del benefici final (euros)

Seed	7907	3776	8108	1173	2903	3184	7247	5060	3332	777
Sol Ini 3	352668	330512	427647	372941	402330	347248	397669	462806	409991	404429
Sol Ini	266984	318116	345193	302471	309323	322532	297840	311500	332893	305424

Temps (ms)



Boxplot comparatiu del temps d'execució (ms)

Com podem veure en els boxplots, el benefici que s'obté en les dues solucions inicials és molt semblant i per tant, no podem determinar quina solució inicial genera una solució més òptima.

On sí que podem apreciar una gran diferència és en el temps d'execució per cada generació de solució inicial, podem veure que quan generem la SolucioIni, el temps d'execució total és molt inferior al de SolucioIni3, per tant podem afirmar que no es compleix la hipòtesi nul·la ja que, la SolucioIni és un mètode millor que la SolucioIni3.

4.4 Experiment 3: Determinar els paràmetres en Simulate Annealing

En aquest experiment hem de determina quins paràmetres proporcionen un millor resultat pel Simulated Annealing amb el mateix escenari, fent servir la mateixa funció heurística i els operadors i l'estratègia de generació de la solució inicial escollits en els experiments anteriors.

Observació	El valor que prenguin els paràmetres K i λ influirà en l'estat final que obtingui el Simulated Annealing.
Plantejament	Triem diferents combinacions de valors per a K i λ, i observem el benefici de l'estat final trobat.
Hipòtesi	Per a valors més alts de λ i més petits de K obtindrem

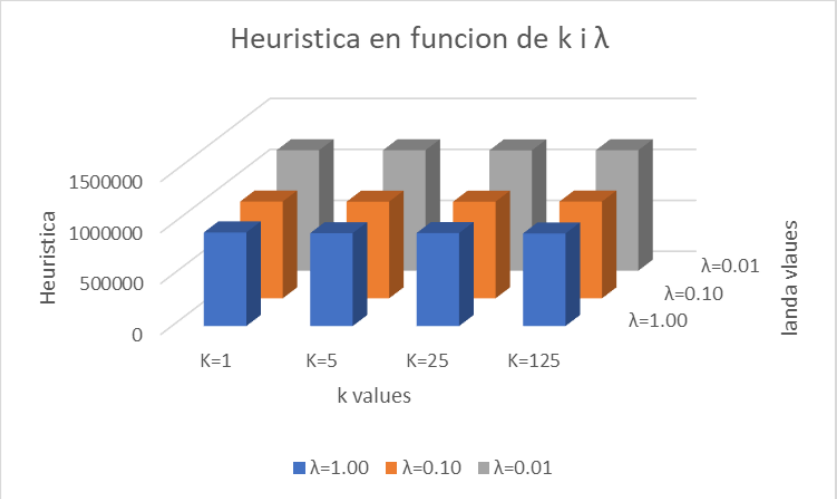
	pitjors resultats.
Mètode	• Generarem 6 llavors aleatòries. • Executarem 10 execucions per cada llavor per a tota combinació possible de K = {1, 5, 25, 125} i λ = {1, 0.1, 0.01}. • Farem les mitjanes dels resultats obtinguts a les 6 execucions de cada combinació. • Compararem els resultats obtinguts.

Degut a l'espai de complexitat de la funció annealing (4 variables), primer de tot haurem de de determinar quins valors són adequats per les variables steps i sitter pel nostre problema. Per aconseguir això, el que farem serà en un principi fixar uns valors fixos per lambda i k, i veurem que SA no expandirà més nodes quan la variable steps és relativament gran, allò que hem fet ha sigut provant cada vegada un valor de step més gran que l'anterior fins que no hem vist diferència entre els beneficis obtinguts. Hem començat amb el valor per defecte de SA amb 10000 steps, i hem anat augmentant el valor fins a 1000000, que es pot tractar com una cota superior relativa (el stiter per defecte els 100, aquesta variable no l'hem tocat).

Amb l'objectiu d'analitzar els paràmetres k i λ per obtenir una millor solució per la cerca local de Simulated Annealing, el que hem fet ha sigut dissenyar un experiment provant diferents valors per k i λ, i analitzant com poden influir aquests 2 paràmetres a la nostra solució òptima de SA.

	K=1	**K=5**	**K=25**	**K=125**
λ=1.00	917974	910964	912307	907856
λ=0.10	949996	948184	947847	949309
λ=0.01	1181397	1181707	1180422	1181427

Benefici (euros)



Un cop obtinguts les dades en fer l'experiment proposat, es pot veure al gràfic que la variable k no influeix molt en el benefici obtingut sinó que és lambda la variable significativa per determinar l'augment d'aquest benefici. Per tant, podem concloure que amb un valor d'steps p de 1000000 i un sitter de 100, mentre que lambda sigui menor o igual que 0,01 aconseguiríem una solució relativament bona, això significa que la hipòtesi plantejada al principi és totalment contrària als resultats aconseguits.

4.5 Experiment 4: Estudi de l'evolució del temps d'execució

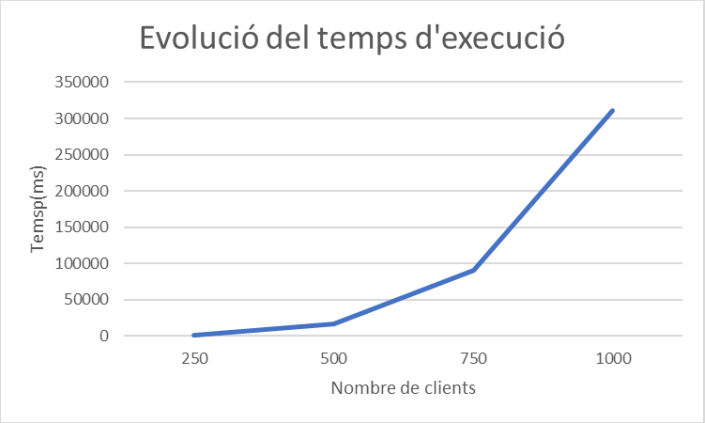
En aquest experiment estudiarem com evoluciona el temps d'execució per trobar la solució per valors creixents dels següents paràmetres.

4.5.1 Augmentar clients mantenint el mateix nombre de centrals

Observació	La seed influencia els clients assignats inicialment.
Plantejament	Augmentar el nombre de clients i observar el seu temps de cerca.
Hipòtesi	Amb menys clients menys temps de cerca amb més clients trigarà més.
Mètode	• Generarem 10 llavors aleatòries per cada problema. • El nombre de centrals per cada problema són 40. • Utilitzem algorisme de HC.

	• Comencem amb 250 clients i augmentem de 250 en 250 fins que s'observa la tendència.

Un cop obtingudes les dades d'aquest experiment hem pogut donar el temps resultant d'execució per uns nombres de clients que són 250, 500,750 i 1000 clients i es pot observar clarament una funció que va augmentant depenent del nombre de clients que tinguem. Però per 1500 clients hem vist que ens proporciona un cas extrem quan la proporció de clients garantits és de 75%. Per 1500 clients hi ha aproximadament 1125 clients garantits i amb 40 centrals elèctriques no acostuma a existir un cas que pugui servir tots els clients.



seed\Client	250	500	750	1000
7907	2130	15443	82735	266984
3776	1556	17904	94556	318116
8108	2205	17229	88054	345193
1173	1448	18631	87718	302471
2903	1442	17939	88381	309323
3184	1572	14962	87109	322532
7247	1677	17093	93469	297840
5060	2113	14973	101496	311500
3332	1670	16263	97287	332893
777	1706	16270	81709	305424

Temps (ms)

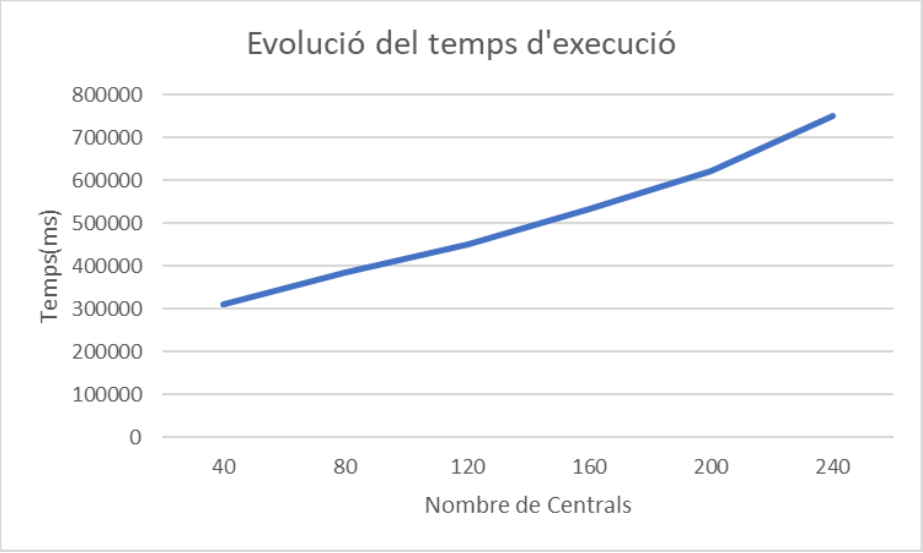
En conclusió per als resultats obtinguts de l'experiment podem afirmar que per un nombre de clients més gran de 1500 no existeix una solució vàlida, però es pot

observar la tendència de l'augment d'execució de temps en funció del nombre de clients.

4.5.2 Augmentar Centrals mantenint el mateix nombre de clients

Observació	La seed influencia els clients assignats inicialment.
Plantejament	Augmentem el nombre de centrals i observem el seu temps de cerca.
Hipòtesi	Amb menys centrals menys temps de cerca i amb més centrals trigarà quasi el mateix o més ràpid.
Mètode	• Generarem 10 llavors aleatòries per cada problema. • El nombre de clients per cada problema són 1000. • Utilitzem algorisme de HC. • Comencem amb 40 centrals i augmentem de 40 en 40 fins que s'observa la tendència.

Per aquest experiment per cada augment de centrals hem decidit augmentar 10 centrals de tipus A, 10 centrals de tipus B i finalment 20 centrals de tipus C.
Com podem veure a la gràfica obtinguda, clarament es veu que té una funció creixent depenent del nombre de centrals que hi hagi.



Seed\Nombre	40	80	120	160	200

de centrals					
7907	266984	342515	423854	480214	582194
3776	318116	420241	423591	557512	612429
8108	345193	381244	482354	572485	642128
1173	302471	364251	442451	512458	601285
2903	309323	372991	449095	532148	625182
3184	322532	415925	491204	568421	652418
7247	297840	388235	442638	521528	632152
5060	311500	401241	471243	561249	642214
3332	332893	392341	425156	525012	624859
777	305424	374523	447551	509274	601749

Temps (ms)

En conclusió, resulta que la nostra hipòtesi plantejat al principi no és del tot certa, com més centrals hi hagi més temps triga a acabar el problema.

4.6 Experiment 5: Determinar la penalització per obtenir una solució vàlida

Enlloc d'utilitzar una generació d'una solució inicial que assigna tots els clients garantitzats i mantenint el compliment de la restricció a la generació de successors, la solució inicial serà buida i a la funció heurística afegirem una penalització a les solucions on aquesta restricció no es compleixi.

Observació	Es generen molt més nodes ja que partim d'una solució buida.
Plantejament	Partim d'una solució buida i mirem si arriba a una solució vàlida.
Hipòtesi	Encara que partim d'una solució no vàlida, l'algorisme Hill Climbing sempre trobarà una solució vàlida però el Simulated Annealing no.
Mètode	• Utilitzem una nova heurística per poder trobar una solució vàlida per l'algorisme de HC i SA.

Per a fer aquest experiment hem creat una nova heurística anomenada *HeuristicaPenalitzacio()* la qual a part de comptabilitzar el benefici total, també suma una constant *bonificacion* multiplicada pel nombre de clients garantitzats als que es subministra energia. D'aquesta manera podem controlar la preferència que li donem als clients garantitzats i a partir d'un cert valor de *bonificacion* assegurar que la solució final serà vàlida. Només podem fer això quan (*bonificacion* >= cost màxim d'obertura d'una central i *bonificacion* > (preu contracte no garantitzat màxim - preu de contracte garantitzat mínim), ja que assignem la màxima prioritat, mitjançant l'heurística a assignar els clients garantitzats i, un cop s'hagi fet, ja anem a buscar el màxim benefici.

Bonificacion > Cost màxim d'obertura d'una central -> Solució final és vàlida

(El cost màxim d'opertura d'una central sempre serà més gran que la diferència màxima de preu de contractes no garantitzats i garantitzats, per s'eliminia perquè és implícit)

El cost màxim d'obertura d'una central es calcula com a:

*Cost màxim = 50*producció màxima + cost d'engegada màxim*

Així doncs el cost màxim és 57500 (50*750 + 20000) i, per tant, el rang de *bonificacion* per assegurar una solució vàlida és [57500,∞)

4.7 Experiment 6: Estudi de la distribució de clients en les centrals

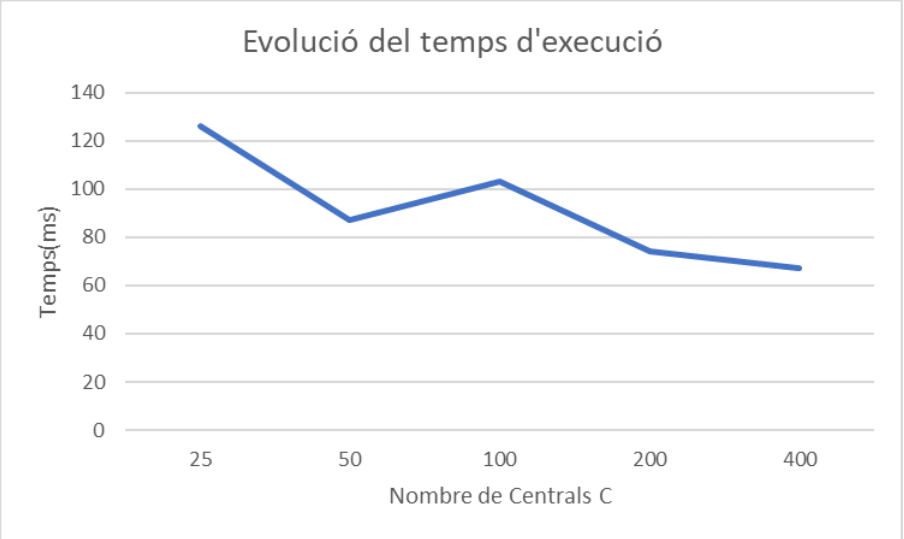
En aquest experiment comprovarem si augmentant el nombre de centrals de tipus C, provoca que les centrals de tipus A i B s'utilitzin menys. Ja que, al tenir més o menys centrals més petites distribuïdes geogràficament pot reduir les pèrdues en el transport i, per tant, subministrar a més gent de manera eficient.

4.7.1 Estudi amb Simulated Annealing

Observació	A l'augmentar el nombre de centrals tipus C farà que les altres centrals s'utilitzen menys.
Plantejament	Doblar el nombre de centrals tipus C cada vegada i observem el seu temps de cerca i el nombre de clients

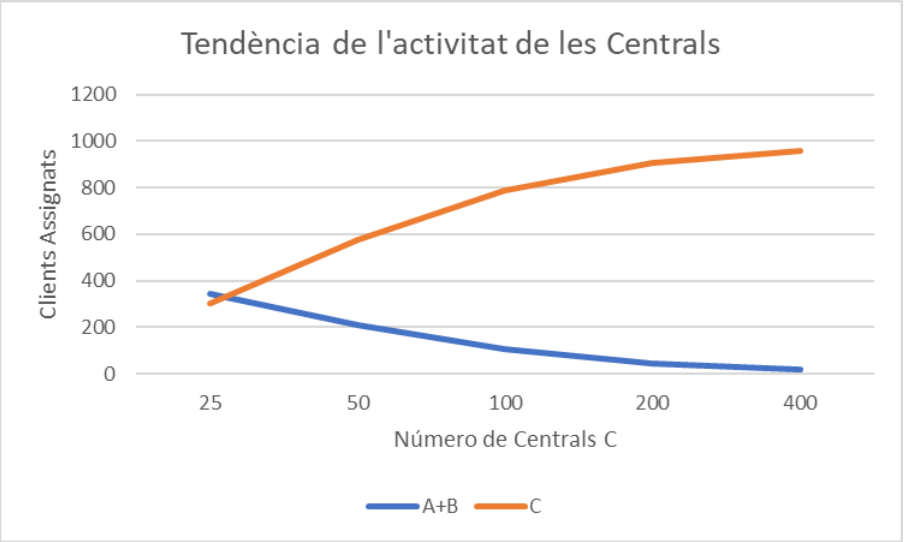
	assignats a cada central.
Hipòtesis	Amb més centrals de tipus C més temps triga a trobar una solució òptima.
Mètode	• Per 1 llavor aleatòria executarem 6 vegades i obtindrem el valor mig del nombre de centrals utilitzats de tipus A i B+C i anirem doblant el nombre de centrals C. • Utilitzarem SA per fer l'experiment.

En el cas de SA, un cop obtingut les dades de l'experiment proposat es pot observar que el temps d'execució en funció de l'augment del nombre de centrals tipus C no varia molt. De fet, una característica que es pot veure a la gràfica que a mesura que anem doblant el nombre de centrals C. Es veu que amb més centrals tipus C el seu temps de cerca és menor, ja que a la gràfica té es veu una tendència decreixent.



Nombre de centrals C	**seed=** 1234	**seed**=2134	**seed**=6666	**seed=**65	**seed=**113	**seed=**521
25	352668	330512	427647	372941	402330	347248
50	472255	428106	439408	412724	429931	439718
100	681320	625140	633283	671332	660123	645723
200	912515	890123	871230	855125	922921	950214
400	1321241	1243293	1213516	1131401	1300421	1077313

En la següent gràfica es pot observar que com més centrals de tipus C n'hi hagi, més clients estaran assignats a una central tipus C, ja que la proporció de centrals tipus A i B distribuïts comparats amb el C és insignificant.

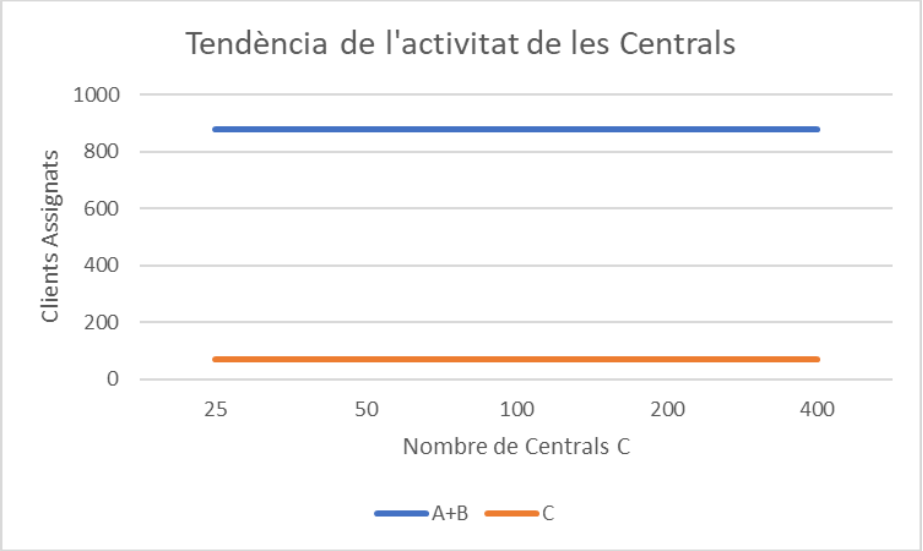


En conclusió, doblant el nombre de centrals C, fa que SA triga menys a trobar la solució òptima cosa que contradiu la hipòtesi plantejada al principi i augmentant un nombre significant de centrals tipus C fa que quasi tots els clients estiguin assignats en centrals C i no en A o B.

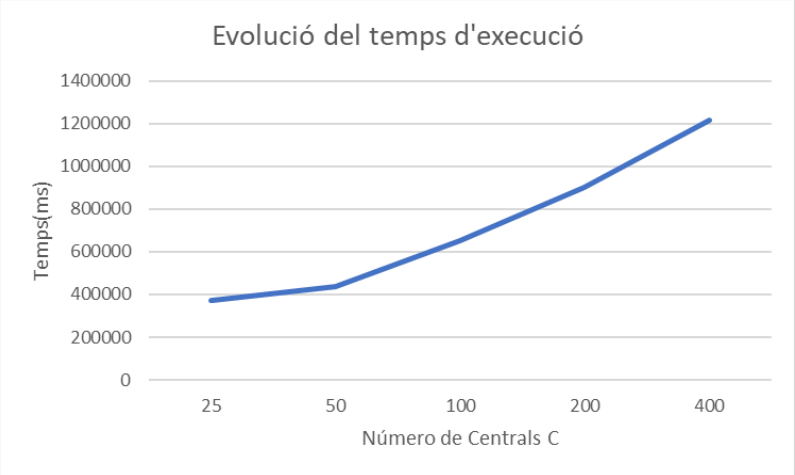
4.7.2 Estudi amb Hill Climbing

Observació	A l'augmentar el nombre de centrals tipus C farà que les altres centrals s'utilitzen menys
Plantejament	Doblarem el nombre de centrals tipus C cada vegada i observem el seu temps de cerca i el nombre de clients assignats a cada central.
Hipòtesis	Amb més centrals de tipus C més temps triga a trobar una solució òptima.
Mètode	• Per 6 llavors aleatòries obtindrem el valor mig del nombre de centrals utilitzats de tipus A i B+C i anirem doblant el nombre de centrals C. • Utilitzarem HC per fer l'experiment.

En el cas de HC, un cop obtingudes les dades veiem que en un principi quasi tots els clients estan assignats a les centrals A, però a mesura que anem augmentant el nombre de centrals C es veu que el nombre de clients assignats a cada tipus de centrals es manté sigui per 25 centrals C o per 400 centrals C. Això és pel fet del propi característica de HC.



Al gràfic de temps d'execució es pot veure que amb més centrals de tipus C hi afegim, més temps tarda l'algoritme HC a trobar una solució òptima pel nostre problema.



En conclusió, per molt que augmentem el nombre de centrals C, sempre mantindrà el mateix nombre de clients assignats a les centrals de cada tipus, ja que HC troba una solució quan pot assignar clients amb n centrals, per molt que augmentem el nombre de centrals sempre es queda amb la primera assignació.