Intel·ligència Artificial

Pràctica de Cerca Local

Cristian Planas i Pere Joan Martorell

Index

1. Introducció

En aquesta pràctica es planteja optimitzar l'organització dels plans de rescat que s'han d'establir al rescatar un gran nombre de persones d'una àrea de cert tamany. L'àrea en qüestió, per simplificar el problema, s'estudiarà com a quadrada, i d'unes mesures de 50x50, el que ens dóna 50^2=2500 possibles posicions.

La principal limitació del problema és la següent: els helicòpters no tindran capacitat infinita, i així doncs, hauran de tornar a la base per a poder recollir més grups de persones. La capacitat dels helicòpters serà exactament de 15 persones. A més, mai podrem tenir a un helicòpter més de 3 grups diferents de persones.

De cara a resoldre el problema, hem de tenir en compte les caraterístiques següents: la cantonada superior esquerra serà la base dels helicòpters, és a dir, la posició de la qual sortiran al seu primer viatge, i a la qual han de tornar per descarregar els grups rescatats. Aquest punt serà el (0,0) dins del nostre mapa. Aquesta posició ha resultat ser de certa importància, doncs els resultats variarien molt si, per exemple, la base estigués al mig del mapa (25,25); en aquest cas, la base estaria més propera a la majoria dels punts, i tornar a la base resultaria menys costós. Sempre que un grup es carregui a un helicòpter, aquest tardarà 5 minuts en tornar a enlairar-se; en el cas que haguem de descarregar un grup a la base, aquest tardarà 10 minuts en tornar a sortir.

Els criteris a optimitzar (en aquest cas, minimitzar) són els següents:

- Temps total necessari per a rescatar a tots els grups
- Temps total necessari per a rescatar a tots els grups, però minimitzant el necessari per a rescatar els grups de prioritat 1.

En aquests criteris hi ha un punt que no es comenta directament a l'enunciat, però que és una qüestió principal quan hem de cerca un heurístic adequat a aquests criteris: no només hem de tenir en compte el temps en el qual hauran estat rescatats tots els grups (és a dir, l'helicòpter amb un temps més elevat), sino les maneres de retallar el temps de tots els helicòpters: de totes les sortides de tots els helicòpters; a aquest segon factor l'hem anomenat "benzina". Si només tenim en compte el primer factor, les millores estaran molt limitades; si només tinguessin en compte el segon, encara que millorariem, l'algoritme ens generaria helicòpters buits (doncs per ell seria el mateix un helicòpter

fent quatre sortides que dos helicòpters fent-ne dos i dos). L'explicació de com hem resolt aquest problema es troba al capítol cinc, que està dedicat als heurístics que fem servir per generar les solucions.

Els algoritmes de cerca local que usarem per realitzar aquests càlculs són el Hill Climbing i el Simulated Annealing. Amb ells haurem d'experimentar modificant paràmetres, usant diferents configuracions inicials, aplicant o traient restriccions i provant els diferents heurístics per veure quins són els que ofereixen millors resultats.

2. Representació de l'estat

El primer a plantejar-se és què és el que considerem estat en el context del nostre problema.

Un estat és qualsevol configuració vàlida (o solució candidata) de rutes, recordem que qualsevol problema de cerca local es mou sempre dins l'espai de solucions i no en tot l'espai de camins, per tant, tot estat és solució.

Definirem com a solució vàlida aquella que compleixi amb les restriccions de solució marcades per l'enunciat, és a dir, aquella en la que tots els grups siguin rescatats; a més, en cap moment un helicòpter podrà contenir més de 15 persones. Més enllà d'aquestes restriccions fortes, tota configuració serà solució.

0.1. Espai de Cerca

L'espai de cerca és molt gran tenint en compte la quantitat de combinacions possibles que es poden donar. Precisament per això, s'encara el problema mitjançant una estratègia de cerca local, que evita visitar gran part de l'espai de cerca. En general, l'espai de cerca s'obté recorrent totes les possibles permutacions de grups dins de la trajectoria d'un helicòpter. Això és així evidentment, sempre que la nova configuració sigui un estat vàlid, és a dir, una solució. Perquè sigui possible haurem de partir d'una solució inicial, que anirem millorant mitjançant el nostre algoritme de cerca.

Així, l'espai de cerca serà aproximadament de ng*ns*nh.

0.2. Estructura de dades

Per emmagatzemar l'estat hem dissenyat les següents estructures de dades.

2.2.1 Grup de persones

Un grup de persones estarà definit pels següents atributs: un boolean que indicarà si el grup és prioritari o no (aquest boolean tindrà 1/3 de possibilitats de ser true); un enter que serà igual a la quantitat de persones que hi ha dins d'aquest grup; dos enters pos_x i

pos_y que ens indicaran la posició del grup dins del mapa 50x50 del territori; a més, tindrem un enter id que ens servirà per identificar el grup a mesura que anem operant amb ell. Aquest darrer atribut podria ser eliminable, però a més de ser més senzill d'aquesta manera, es dóna que el seu sustitut natural, fer servir la posició, és inacceptable; doncs es pot donar que hi hagin dos grups en la mateixa posició. La classe que conté aquesta informació s'anomenarà ginfo.

El grup de persones és la unitat bàsica del nostre problema, doncs sobre ell executarem totes les operacions: els helicòpters seran una mena de caixes on ordenarem els grups de persones.

2.2.2 Vectors de grups de persones

Tots els grups de persones -objectes de tipus ginfo- estaran al vector grups, on l'ordre coincidirà amb la id de l'objecte. La classe *GInfo* conté la informació relacionada amb els grups a rescatar: posició en coordenades *x* i *y*, el nombre de persoens que el formen, el seu tipus de prioritat i un identificador.

2.2.5 Vector de sortides d'un helicòpter

Tindrem un vector anomenat *sortides* de mida igual al nombre d'helicòpters on tindrem LinkedLists (tipus propi de Java) de sortides, concretament una per cada helicòpter.

2.2.5.1 Sortida

L'helicòpter, en si, no ens resulta útil com a estructura de dades; és molt millor convertir-lo en el conjunt de sortides que fa un helicòpter. Així doncs, entendrem una sortida com una LinkedList d'enters, on aquests seràn els identificadors dels grups destí.

2.2.6 Vector de nombre de persones

Per agilitzar el recompte de persones a cada moviment de grups tindrem un vector que ens indicarà per cada sortida quin total de persones conté.

2.2.6 Vector de distàncies

Com a dada auxiliar, tindrem un vector de floats anomenat dist que ens indicarà la distància entre dos grups. Per exemple, dist[3][6]=dist[6][3] serà la distància entre el grup 3 i el grup 6. Ho calculem amb la següent operació:

```
double aux_x=grups[i].pos_x-grups[j].pos_x;
double aux_y=grups[i].pos_y-grups[j].pos_y;
dist[i][j]=java.lang.Math.hypot(aux_x, aux_y);
```

Per convenció, decidim que la distància entre un grup i si mateix (per exemple dist[3][3]) serà igual a zero.

3. Generació de la solució inicial

3.1. Solució inicial aleatòria

Aquesta solució està pensada per donar-nos una solució (és a dir, una opció que compleixi tots els requeriments forts) ràpidament, per a executar l'algoritme de cerca local sobre ella. El cost d'aquesta solució és de $\theta(ng)$.

3.1.1 Funcionament

Això ho implementarem de la següent manera: cadascun dels helicòpters tindran un torn per moure's, i aniran a un grup no rescatat escollit de manera aleatòria. En el cas de que aquest grup resulti ser més gran del que pot carregar l'helicòpter (carrega actual+persones del grup > 15), l'helicòpter es dirigirà a la base.

3.2 Solució inicial de qualitat (per proximitat)

La nostra solució inicial de qualitat millorarà la aleatòria optimitzant un dels factors a tenir en compte (el temps) fent que els helicòpters es dirigeixin al grup més proper a la posició on es troben en aquell moment. El cost d'aquesta solució és de $\theta(ng * ng)$.

3.2.1 Funcionament

Això ho implementarem de la següent manera: cadascun dels helicòpters tindran un torn per moure's, i aniran al grup no rescatat més proper respecta a la seva posició actual. En el cas de que aquest grup resulti ser més gran del que pot carregar l'helicòpter (carrega actual+persones del grup > 15), l'helicòpter es dirigirà a la base.

3.2.2 Exemple

Donem un exemple explicat del funcionament d'aquest algoritme de solució inicial.

3.2.2.1 Posicions

Al nostre exemple, els grups seran definits de la següent manera:

Grup 0: (16,7) (4 persones)

Grup 1: (19,29) (7 persones)

Grup 2: (38,16) (8 persones)

Grup 3: (15,2) (11 persones)

Grup 4: (49,48) (11 persones)

Grup 5: (26,43) (9 persones)

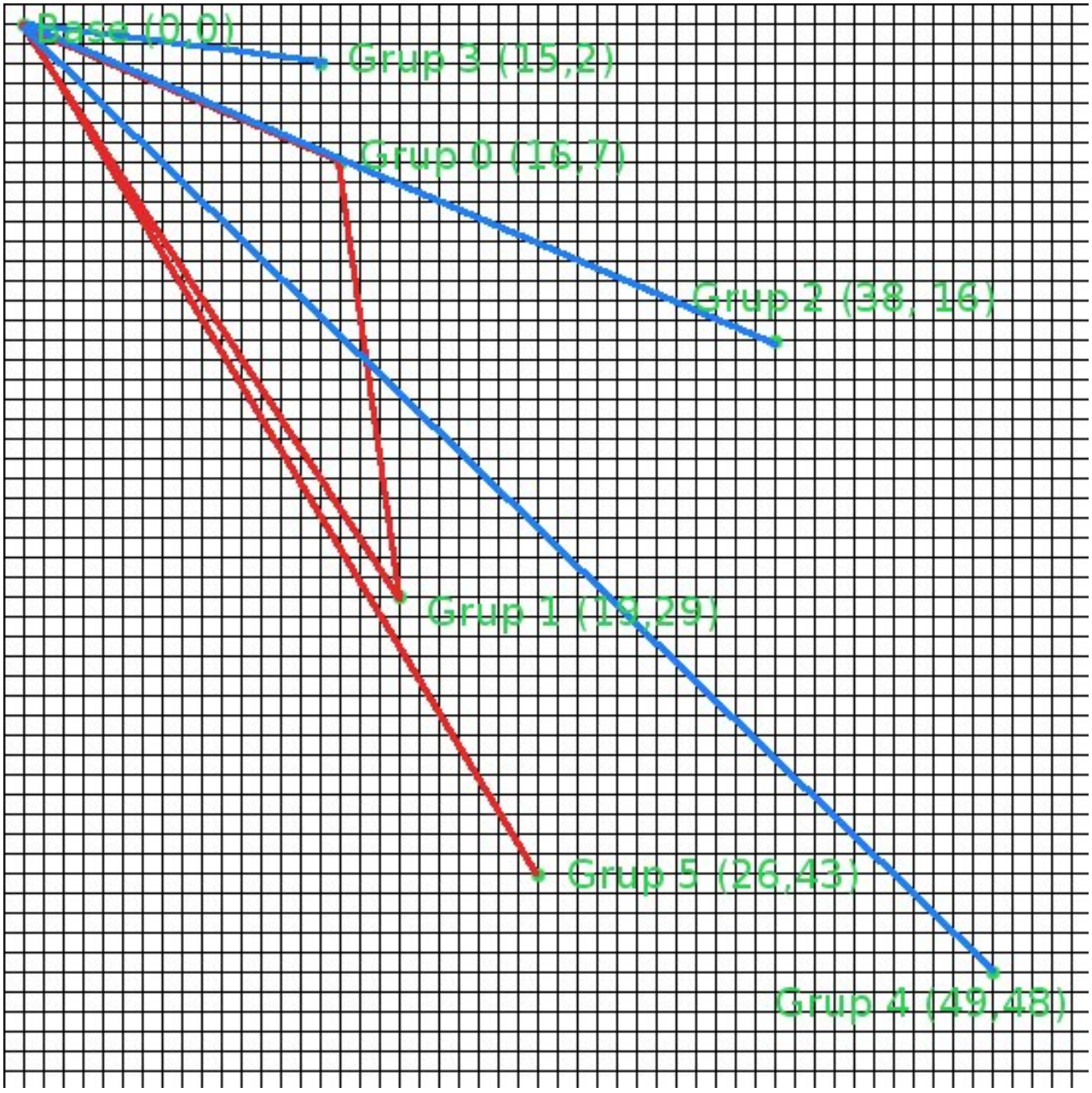
La prioritat de cara a aquest exemple no importa, doncs l'algoritme no la té en compte.

La base, com sempre, serà a la posició (0,0).

Podem calcular fàcilment que les distàncies seran:

	Base	Grup 0	Grup 1	Grup 2	Grup 3	Grup 4	Grup 5
Base	X	17.5	34.7	41.2	15.1	68.6	50.2
Grup 0	17.5	X	22.2	23.7	5.1	52.6	37.4
Grup 1	34.7	22.2	X	23	27.3	35.5	15.6
Grup 2	41.2	23.7	23	X	27	33.8	29.5
Grup 3	15.1	5.1	27.3	27	X	57.2	34.5
Grup 4	68.6	52.6	35.5	33.8	57.2	X	23.5
Grup 5	50.2	37.4	15.6	29.5	34.5	23.5	X

3.2.2.2 Resultat



Helicòpter 0	Helicòpter 1
De base a grup 3 (11 persones) 15, 1 km	De base a grup 0 (4 persones) 17,5 km
De grup 3 a base 15,1 km	De grup 0 a grup 1 (7 persones) 22,2 km
De base a grup 2 (8 persones) 41,2 km	De grup 1 a base 34,7 km
De grup 2 a base 41,2 km	De base a grup 5 (9 persones) 50,2 km
De base a grup 4 (11 persones) 68,6 km	De grup 5 a base 50,2 km
De grup 4 a base 68,6 km	
Distància: 249,8 km	Distància: 174,8 km

Inicialment, tots dos helicòpters surten de la base. El dos grups més propers a la base són el 3 i el 0, en aquest ordre; així, a aquests dos grups hi aniran l'helicòpter 0 i l'helicòpter 1, respectivament. Al següent pas, helicòpter 0 tindrà com a grup més proper i no rescatat el grup 2, però com que 11+8>15, tornarà a la base. En canvi, l'helicòpter 1 té com a grup més proper i no rescatat el grup 1, i 7+4<=15. Així doncs, l'helicòpter 1 anirà a rescatar al grup 1. Al següent torn, L'helicòpter 0 anirà a buscar el grup més proper a la base que queda, el grup 2; l'helicòpter 1 tornarà a la base. A partir d'aquest moment, cada helicòpter recollirà un grup i tornarà a la base, ja que mai es donarà que grup rescatat+grup més proper <= 15.

3.3 Solució inicial d'alta qualitat

A més de les demanades per l'enunciat, hem desenvolupat un algorisme de solució inicial alternatiu, que ens dóna una solució amb un heurístic inicial més baix. No obstant, el principal aventatge d'aquesta solució és que redueix lleugerament el temps de procés del problema. El cost d'aquesta solució és de l'ordre de $\theta(ng * (ng + nh))$.

3.3.1 Funcionament

Comptarem el temps que tarda cada helicòpter; cada vegada escollirem per a "sortir" a l'helicòpter que hagi estat menys temps a l'aire -així reduim el màxim-. Agafarem el grup més proper que aquest helicòpter pugui agafar; és a dir, aquell que compleixi que:

Persones a l'helicòpter+Persones del grup <= Màxim de persones a l'helicòpter

Si no trobem cap grup amb aquesta característica o ja tenim tres grups a l'helicòpter, tornarem a la base, sumant al temps de l'helicòpter el que tardem en arribar a la base.

1. Operadors

Per aconseguir que l'algorisme de cerca local en qüestió sigui capac de moure's per l'espai de solucions és necessari implementar operadors. Aquests operadors tenen com a finalitat crear un estat diferent (on els paràmetres de qualitat tenen valors potencialment diferents) veí a l'estat al que se li aplica l'operador. Això signica que aquests ens permeten obtenir un estat contigu a l'original. D'aquesta manera, l'algorisme de cerca local pot consultar tots els estats propers per veure quin es el que promet millors resultats.

El mecanisme utilitzat per la implementacio d'AIMA es sol·licitar per un estat quins son tots els possibles successors partint d'ell. Per això, generem tots els possibles estats que sorgeixen d'aplicar un operador (en un sol pas) i retornar-los a l'algorisme perque segueixi amb la cerca.

En el problema que tractem, hem reduit qualsevol possible canvi d'estat en l'aplicacio d'un d'aquests dos operadors:

- Moure un grup.
- Intercanviar la posició de dos grups.

Només amb el primer operador ja podem moure'ns per tot l'espai se solucions; però, donades les limitacions que ens imposa el problema (màxim de 3 grups i 15 persones per sortida) i tenint en compte les característiques del hill climbing -que mai agafarà una solució pitjor, encara que aquesta el porti finalment a una solució millor; és a dir, que mai superarà el màxim local-, hem decidir afegir un altre operador, el d'intercanviar grups, que ens permet superar aquesta situació. No obstant, s'ha de tenir en compte que, encara que aquests dos operadors ens donen les solucions més òptimes, consumeixen una gran quantitat de recursos, i necessiten una important quantitat de temps per donar una solució definitiva en Hill Climbing. Ens vam plantejar limitar als operadors d'alguna manera -per exemple, només fent servir el moure-, però no obstant aconseguir un heurístic millor ens ha semblat més important que la velocitat de l'algorisme. Només hem afegit una limitació: el moure només afegirà grups a la primera posició de la sortida; hem demostrat que els efectes d'aquesta limitación són molt menors per l'heurístic i molt importants en el temps necessari per produir una solució a l'experiment extra (6.9).

El factor de ramificació serà quadràtic: qualsevol factor de ramificació que no sigui d'aquest ordre serà profundament ineficient.

4.1 Primer operador: moure un grup per les sortides

El nostre primer operador és el de moure grups.per totes les sortides que hi ha al problema. Així, tindrem que el cost d'aquest operador és igual al nombre de grups multiplicat pel nombre de sortides: (ng*ns), on ns serà com a màxim igual a ng i com a mínim ng/3.

4.1.1 Limitacions

Mourem grups per totes les sortides de tots els helicòpters, inclosa una sortida "fantasma" al final de cada helicòpter que en servirà per a generar noves sortides; l'única limitació serà per la sortida propia. Per a reordenar sortides, farem servir intercanviar. Així, tenim que el cost definitiu serà: (ng*(ns-1)). A més, només afegirem el grup mogut a la primera posició de la sortida destí.

4.2 Segon operador: intercanviar grups

El segon operador mourà a la vegada el grup origen a la posició destí i el destí a la posició origen.

4.2.1 Limitacions

Intercanviarem tots els grups. Per evitar fer intercanvis simètrics (canvia la posició del grup 5 amb la del 7 és el mateix que la del 7 amb la del 5), farem que l'algoritme deixi de tenir en compte els grups que han sigut intercanviats com a "origen" com a possible "destí". Així, el cost és de (ng*ng)/2.

Hem preparat un experiment extra (6.9) on comparem els nostres operadors amb els "totals", i demostrem que els nostres tenen aproximadament la mateixa potència mantenint el *elapsed time* molt més baix.

5. Funció de qualitat

5.1 Definició

Per tal d'avaluar la bondat d'una solució i guiar la búqueda pel conjunt d'espai d'estats farem ús de la funció de qualitat. Cal dir que aquesta funció no ens indica quan falta per arribar a trobar la solució.

Tal com se'ns demana a l'enunciat cal que les nostres solucions minimitzin 2 factors:

1. Temps total necessari per rescatar tots els grups
2. Temps total necessari per rescatar tots els grups per minimitzant prioritariament el temps necessari per rescatar els grups de prioritat 1 i a igual valor d'aquest paràmetre és millor la que menys tarda en rescatar tots els grups.

5.2 Heurístics

5.3.1 Heurístic 1: Temps de rescat total

A la hora de dissenyar aquest heurístic hem tingut en compte els següents factors ordenats per prioritat:

- Minimitzar el temps total del rescat
- Minimitzar la suma total dels recorreguts dels helicòpters que es correspondria amb el consum total de benzina

Se'ns demana que l'heurístic minimitzi el temps total per rescatar tots els grups. En primera instància lo més lògic seria pensar que per minimitzar el temps total caldria obtenir simplement el màxim dels recorreguts fets pels helicòpters, però això no ens permet discriminar dues solucions en que el conjunt dels recorreguts d'un d'ells és globalment millor.

Per altra banda tenir en compte només la suma del recorreguts dels helicòpters fa que ens allunyem de l'objectiu principal del problema. Això es pot veure gràficament en la comparativa dels dos escenaris següents amb una disposició diferent de les sortides:

5.3.1.1 Exemple aclaratori

Cas 1:

H1 → S1 → S2 → S3 → S4

H2

Cas 2:

H1 → S1 → S2

H2 → S3 → S4

En ambdos casos la suma total dels recorreguts dels helicòpters és la mateixa, però en canvi no s'està complint l'objectiu principal del problema, que és que el temps de rescat sigui el mínim possible. Seria molt millor el cas 2 que no pas el cas 1, però aquest heurístic no ens ajuda a distingir aquests casos.

Per aquest motiu, em valorat que és millor un heurístic que sigui combinació del dos casos esmentats. És a dir:

$$h_1 = \forall x \in H \mid \max(temps\_helicopter(x)) * numhelic + tempstotal$$

On *H* és el conjunt d'helicòpters de rescat, *temps_helicopter* el temps que tarda l'helicòpter *x* en efectuar el rescat, *num_helic* el nombre d'helicòpters de l'escenari i *tempstotal* la suma de tots els temps de rescat dels helicòpters.

5.3.2 Heurístic 2: Temps total prioritzant els grups de ferits

En aquest cas se'ns demana minimitzar el temps total de rescat de tots els grups però minimitzant prioritàriament el temps de rescat dels grups de prioritat 1. A igual valor d'aquest paràmetre serà millor la solució que menys tarda en rescatar tots els grups.

A la hora de dissenyar aquest heurístic hem seguit la mateixa estratègia que l'heurístic anterior, però ara tenint en compte el temps de prioritat 1. Així doncs, aquest heurístic respondrà als següents factors ordenats per prioritat:

- Minimitzar el temps total del rescat
- Minimitzar el temps de prioritat 1
- Minimitzar la suma total dels recorreguts dels helicòpters que es correspondria amb el consum total de benzina.

En conclusió, tenim en compte tant el màxim dels recorreguts com la suma dels recorreguts de tots els helicopters, però em donat un pes molt més rellevant al temps de prioritat 1 que no pas a la resta de temps.

Formalment l'heurístic es podria descriure com:

$$h_2 = \forall x \in H \mid \max(temps\_helicopter(x)) * numhelic + totalprio$$
$$totalprio = temps\_prioritat\_1 * 3 + temps\_prioritat\_2$$

On *totalprio* és igual al temps de prioritat 1 muliplicat per una constant més el temps de prioritat 2. Com es pot comprovar donem el triple de pes al temps de prioritat 1 que al 2.

Entenem per *temps_prioritat_1* com el temps global des de que s'inicia el rescat fins que els grups de ferits arriben a la base, i ho calculem de la següent manera: quan trobem una sortida que conté un grup de ferits, comptabilitzem el temps d'aquesta sortida i el multipliquem pels temps de les sortides que la precedien, ja que aquestes sortides anteriors són part integrant del recorreguts que ha calgut fer per rescatar-los. **D'aquesta manera propiciem que els grups prioritaris s'avancin en el temps, és a dir, que s'apropin a les primeres sortides dels helicòpters.**

6. Experiments

Per tal de justificar decisions preses durant el transcurs de la pràctica, o bé per comprovar-ne l'eficàcia, s'han anat duent a terme experiments que a continuació llistem i expliquem en profunditat.

El primer que es demanava era determinar quin era l'operador o conjunt d'operador que funciona millor en un escenari donat i amb algoritme Hill Climbing; això s'ha desenvolupat a la secció 6.1. Després, a la secció 6.2 s'han testejat les dues solucions inicials que hem desenvolupat, per descobrir quina d'elles donava un heurístic millor.

A l'apartat 6.3 fem un experiment per optimitzar els paràmetres que fem servir al Simulated Annealing.

Al 6.4 observarem com varia el temps total de rescat a mesura que augmentem el nombre d'helicòpters.

Al 6.5, observarem com es comporta el temps total de rescat i el d'execució depenent de l'heurístic i el nombre d'helicòpters.

Finalment, a 6.6 i per acabar amb els experiments, compararem els dos heurístics amb Hill Climbing i Simulated Annealing.

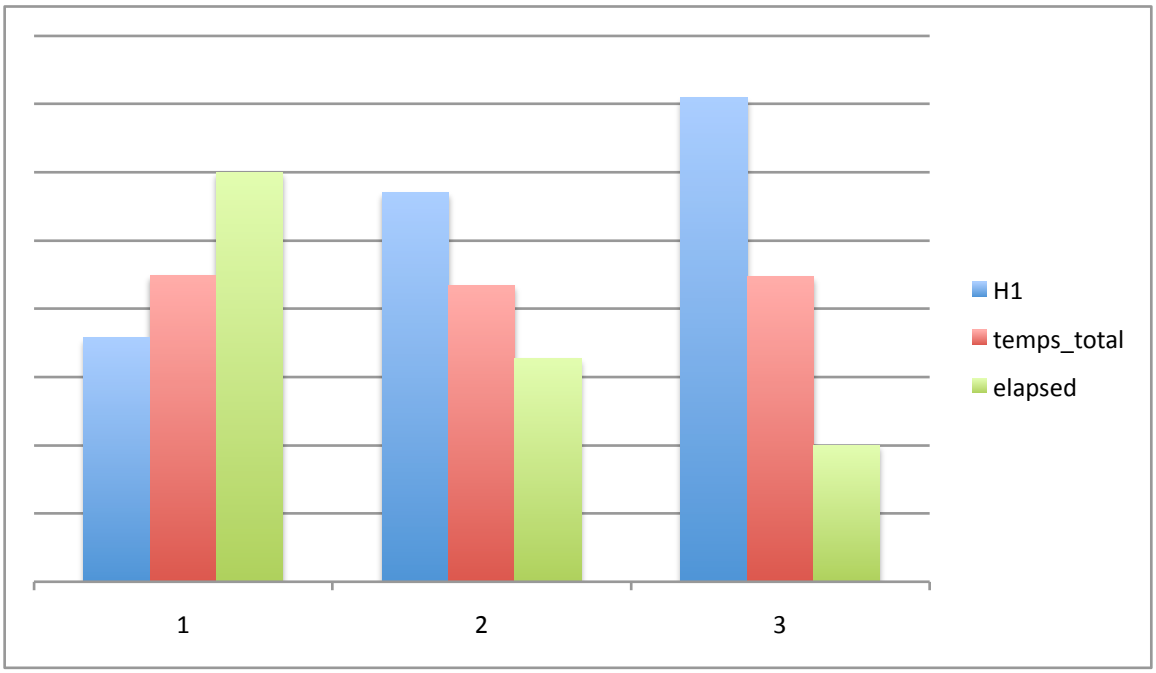
Després comentarem breument la solució al dos problemes plantejats per l'anunciat:

1. El valor p tal que reduïm el màxim el temps de rescat amb el següent heurístic h=p*(temps_prio_1)+(1-p)*(temps_prio_2) (6.7).
2. El tamany de l'helicòpter a partir del qual no es millora de forma clara les solucions (6.8).

Al final d'aquesta secció hem afegit un experiment extra, amb la idea de demostrar que el conjunt d'operadors que hem seleccionat per resoldre el problema són prou potents i prou barats (6.9).

6.1 Comparativa entre conjunts d'operadors

Cal determinar quin conjunt d'operadors dóna millors resultats amb la primera funció heurística i un escenari de 100 grups i 10 helicopters usant l'algoritme de Hill Climbing.



Gràfic 1: Comparativa de conjunts d'operadors

On *(1)* és el conjunt de 2 operadors moure i intercanviar,

(2) és únicament l'operador de moure,

i *(3)* és únicament l'operador d'intercanviar.

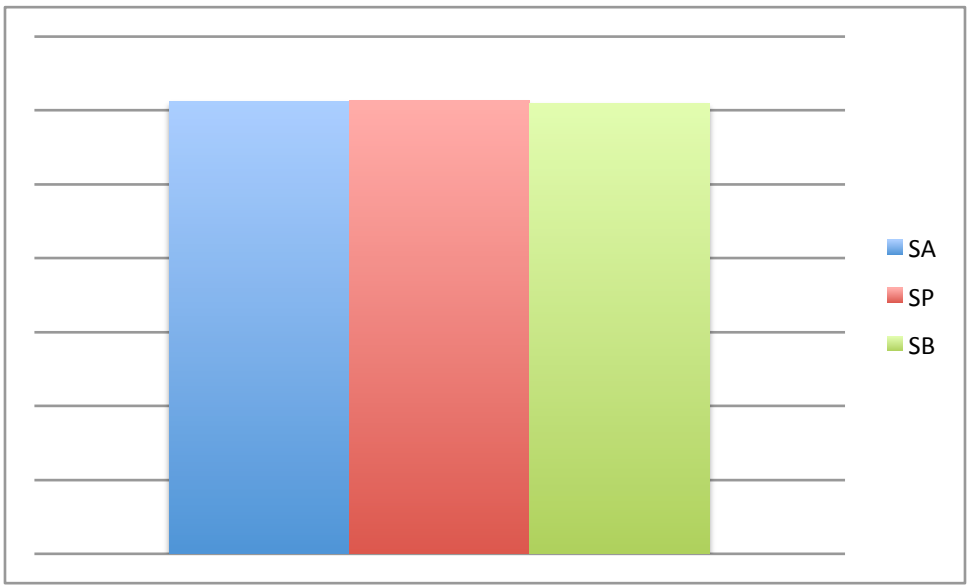
A la gràfica podem observar com, en termes generals, el conjunt d'operadors moure i intercanviar junts es comporten molt millor que cadascun d'ells de forma separada. En **l'experiment 6.9** provem que realment aquest operador és el millor.

En quant al temps total del rescat tots tres es comporten més o menys igual, però en canvi, pel que fa al valor del heurístic 1, que té tant en compte el temps total del rescat com la **suma de tots els temps dels helicòpters**, és clarament millor en el cas en que fem ús dels dos operadors alhora. El nostre heurístic te tant en compte el consum global de benzina dels helicòpters com el temps invertit en el rescat.

En quant al temps de càlcul resulta evident que el tot és la suma de les parts, i per tant el fet d'utilitzar moure i intercanviar alhora suposarà un increment de temps molt més gran que no pas moure i intercanviar per separat.

6.2 Estrategia de generació de solucions inicials

En aquest experiment posarem a prova les solucions inicials en un escenari idèntic al de l'apartat anterior usant Hill Climbing altre cop. Tal com se'ns ha explicat, la generació de solucions a partir d'una solució inicial bona teòricament desemboca en una solució final bona. En el nostre cas, després de fer 20 experiments amb cada tipus de solució inicial, hem arribat a la conclusió que agafar un tipus o un altre no és determinant en quant al valor de l'heurístic 1. En la següent gràfica es pot veure el resultat fruït de les mitjanes de cadascuna de les 20 experimentacions. *SA* és l'acrònim de solució aleatòria, que assigna grups aleatòriament als helicòpters i a les sortides; *SP* és l'acrònim de solució amb proximitat, que assigna els grups als helicòpters segons la proximitat amb el grup que ha recollit anteriorment cada helicòpter i *SB* és l'acronim de SolucióBEST, un tipus de solució que ens dóna una estat inicial amb una heurística molt bona respecte a les altres dues. Aquesta última, tot i ser una bona solució inicial des d'un principi, no acaba desembocant a solucions finals que difereixin massa en quant al valor de l'heurístic.



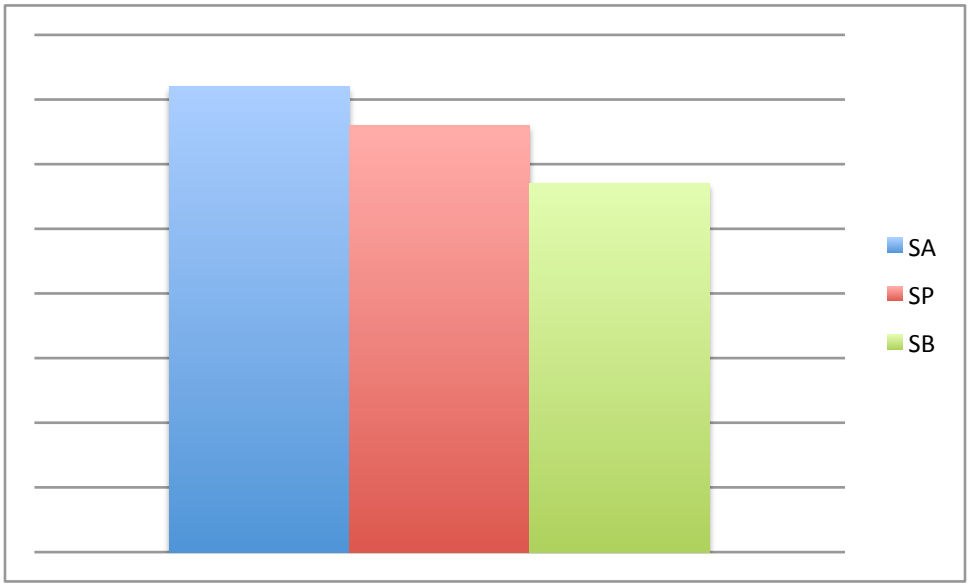
Gràfic 2: Comparativa del valor de l'heurístic 1

Valor de l'heurístic:

SA: 3063.5

SP: 3064

SB: 3050



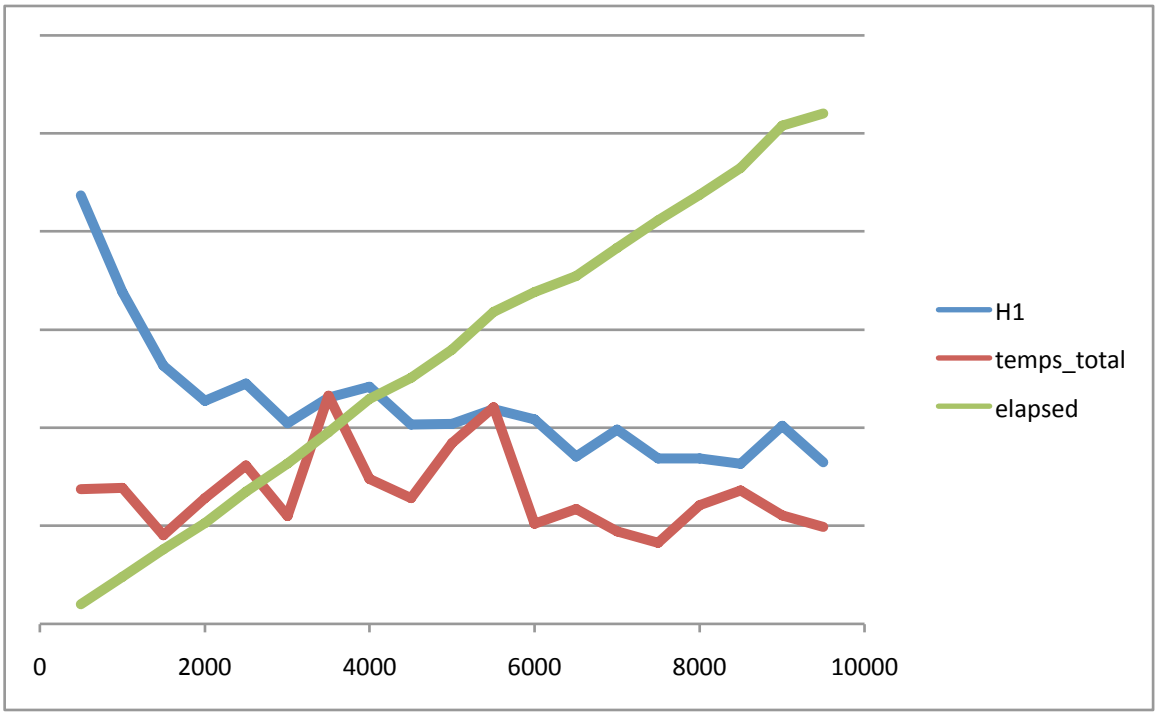
Gràfic 3: Comparativa dels temps d'execució

D'aquesta gràfica s'en pot extreure que si bé no hi ha diferències apreciables pel que fa al valor de l'heurístic, sí que s'observa una millora important en quant al temps de generació de la solució inicial *SolucioBest*. Per tant, serà aquesta la que utilitzarem d'ara endavant en la resta d'experiments.

6.3 Experiment de calibrat dels paràmetres de Simulated Annealing

6.3.1 Iteracions Màximes

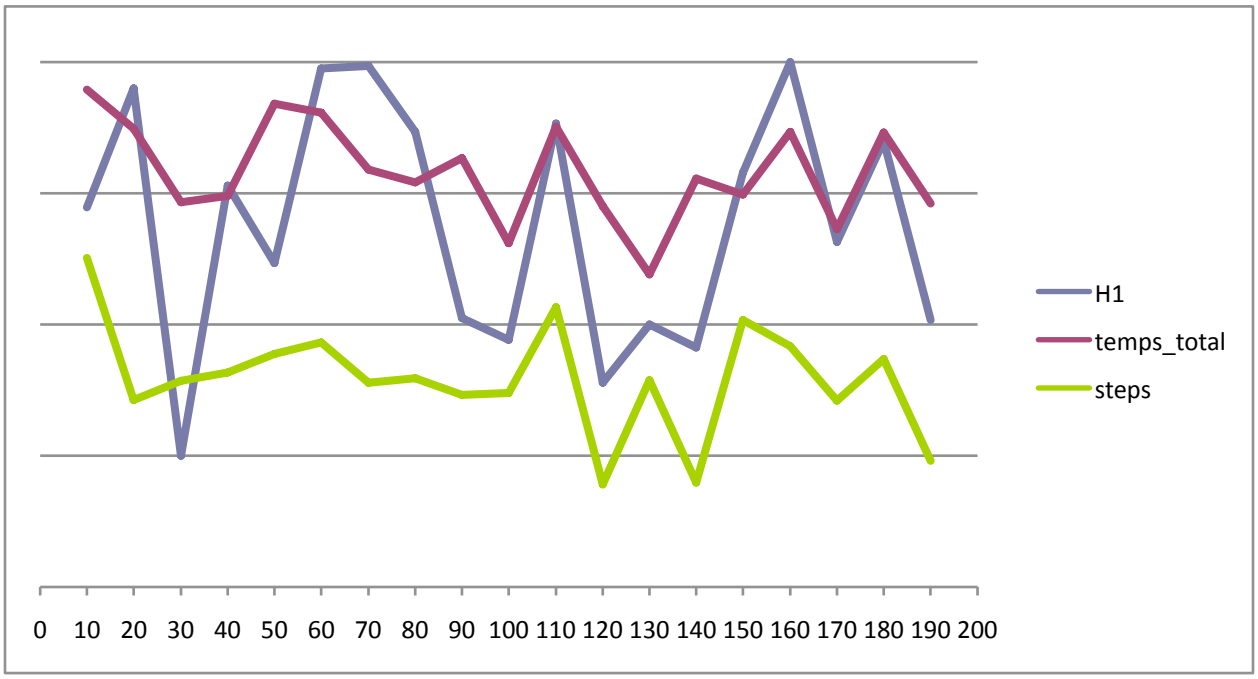
Amb aquest paràmetre definim el número màxim d'iteracions, de certa manera limitant tota la resta de factors obtinguts per l'algorisme perquè no desemboqui en una càrrega excessiva de treball.



Hem experimentat deu cops amb tots els valors entre 500 i 10.000 amb intervals de 500. Observem que l'heurístic va reduint-se lleugerament a partir d'un valor mínim -aproximadament 2.000-. Els millors valors els troben a 6.000 i 8.000; però hem de tenir el compte que el temps creix de forma lineal, és a dir, que **agafarem 6.000**.

6.3.2 Iteracions per cada pas de temperatura

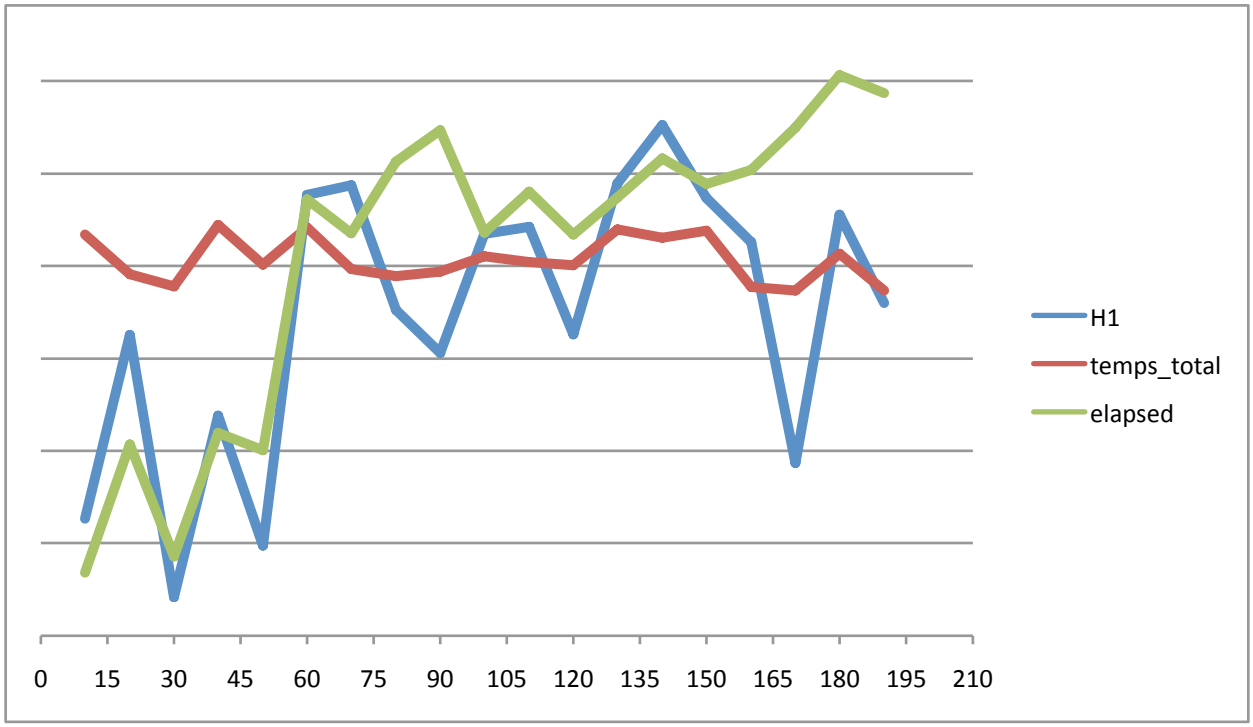
L'algorisme de Simulated Annealing va variant el valor de la temperatura que empra per guiar-se, i en cada temperatura diferent realitza un número d'iteracions. Aquest és el paràmetre que estudiem ara. Hem decidit provar els valors entre 20 i 200 amb increments de 10 per escollir-ne l'òptim.



Observem que els millors valors de l'heurístic els trobem a 30, però també es dóna que entre 120 i 140 tenim bons valors d'heurístic i molt bons de temps_total (el que tarda el darrer grup en ser rescatat). Així doncs, **escollim 130**.

6.3.3 Paràmetre K

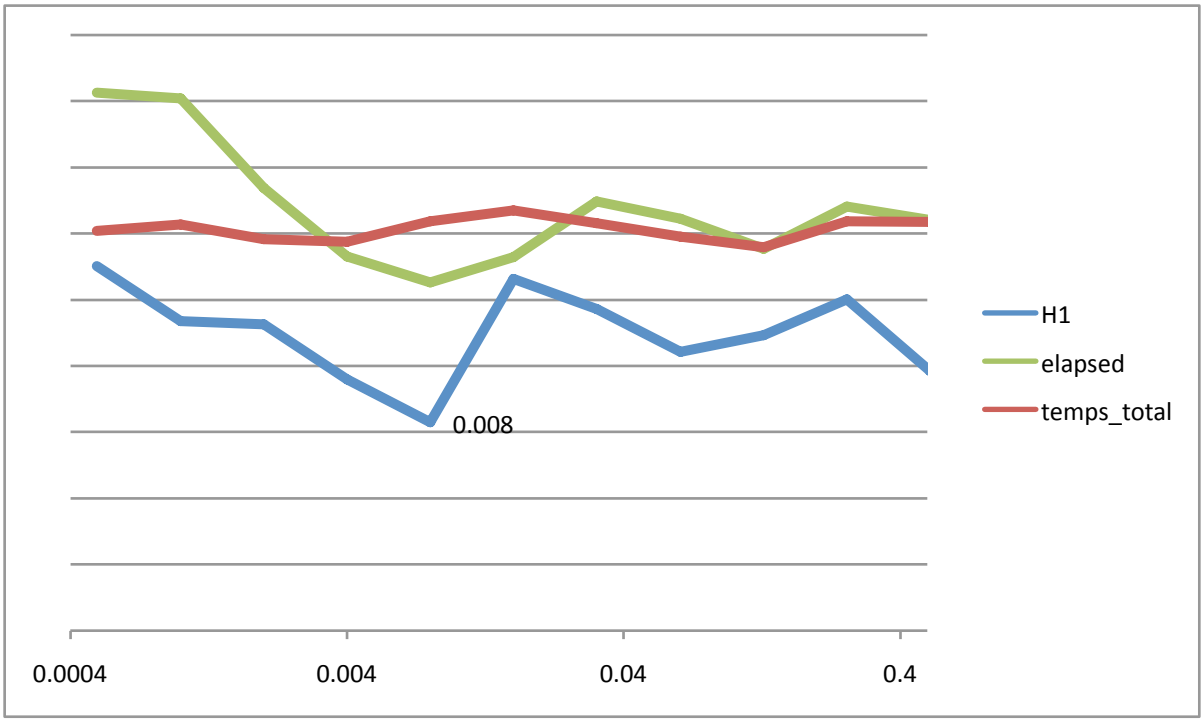
El valor de K determina quant triga la temperatura en començar a descendir. Nosaltres hem provat totes les possibilitats entre 10 i 200 de 10 en 10.



Es mantenen valors bastant estables en el temps_total (el que tarda el darrer grup en ser rescatat), però s'observa clarament com l'heurístic i l'elapsed time augmenten després de k=30. Així doncs, escollim **k=30**.

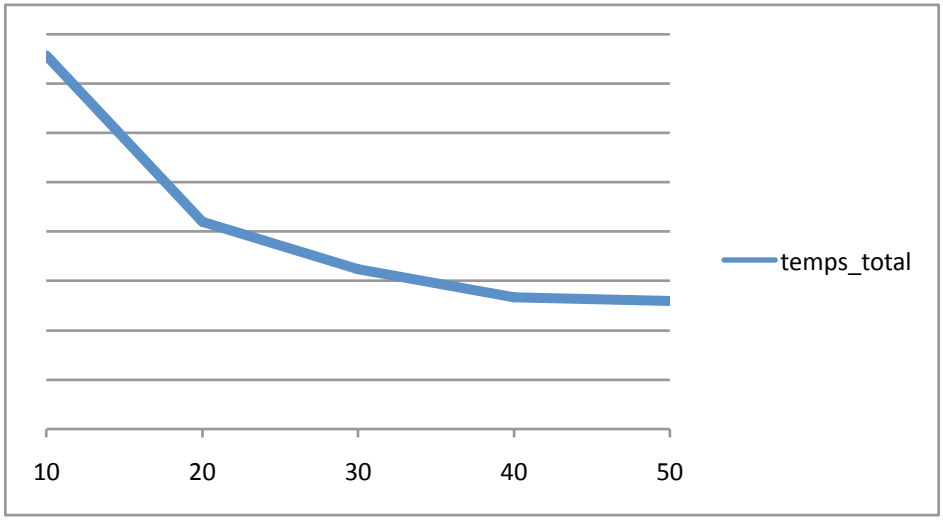
6.3.4 Paràmetre λ

El paràmetre λ és el que marca la pendent del refredament de la funció, o sigui com n'és de ràpid aquest refredament un cop comença. A partir dels gràfics observats a les transparències sobre AIMA i el valor per defecte que aquesta classe defineix per λ, decidim provar els possibles valors entre 0.0005 i 0.512 incrementant a base de multiplicar per dos. D'aquesta manera podem contemplar el comportament de la cerca amb un ampli ventall de valors però sense haver de provar-ne milers. Els resultats són bastant clars en el següent gràfic:

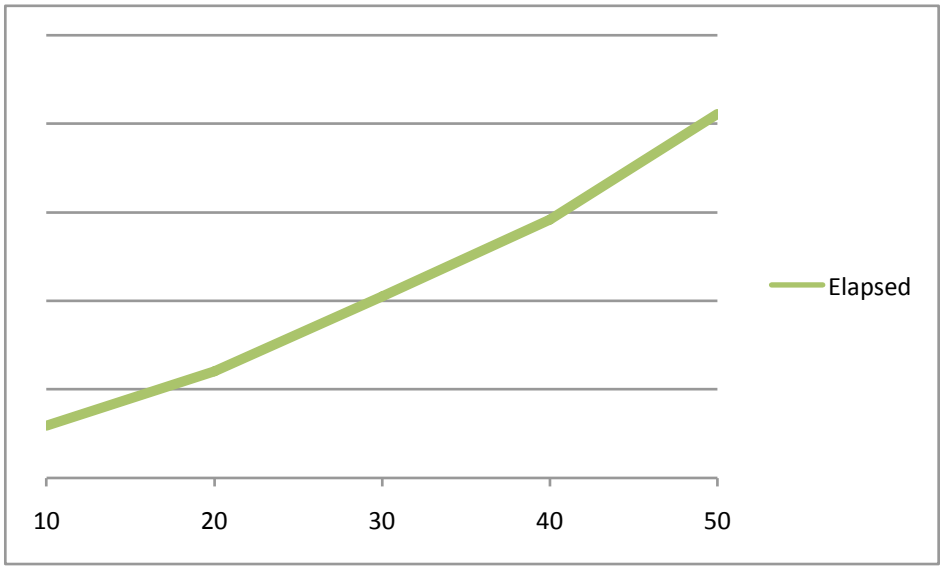


Temps_total es manté molt estable durant tot l'experiment. En canvi, el millor pic tant de temps d'execució com de l'heurística és a 0.008. Per tant, **agafem 0.008**.

6.4 Comportament de l'algorisme amb Hill Climbing i heurístic 1 a mesura que augmenta el nombre d'helicòpters



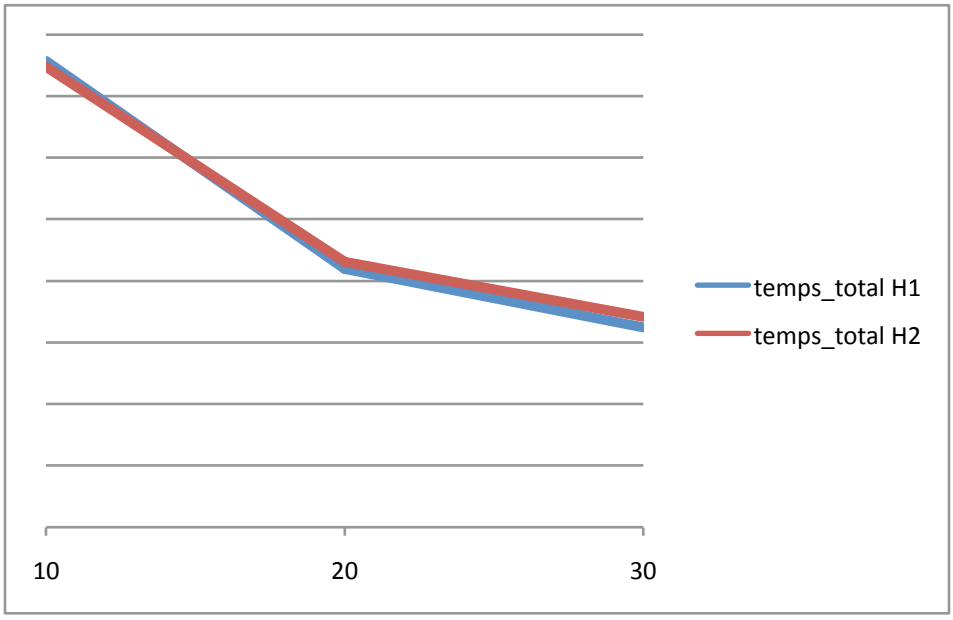
Gràfic 4: Progessió del temps total a mesura que augmentem els helicòpters



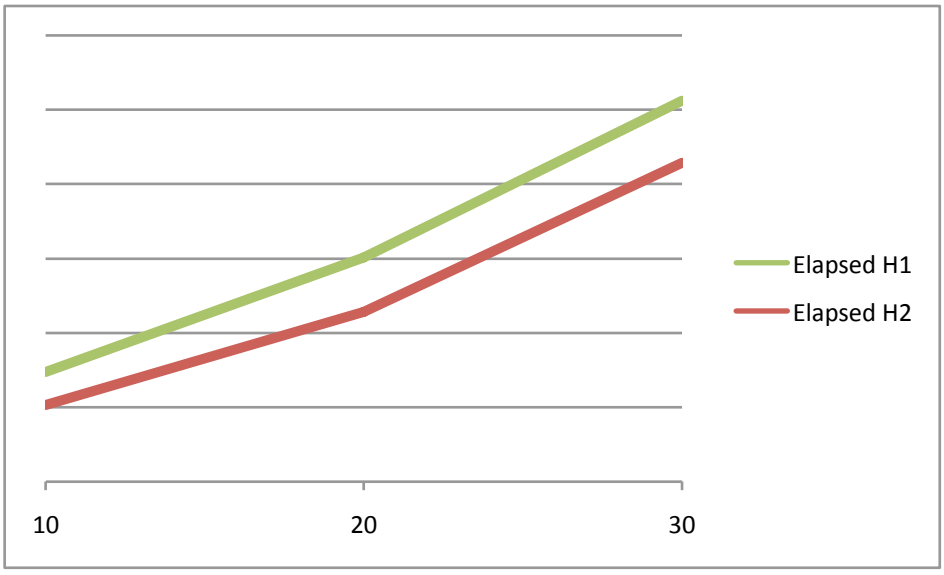
Gràfic 5: Progressió dels temps d'execució a mesura que augmentem els helicòpters

Com és lògic, l'*elapsed time* creix a mesura que augmentem els helicòpters. També de manera normal, el temps_total (el que tarda el darrer grup en ser rescatat) es redueix: no obstant, veiem com cada vegada ho fa de manera més lleugera, especialment a partir de 20 helicòpters.

6.5 Comportament de l'algorisme amb Hill Climbing: heurístic 1 i 2 a mesura que augmenta el nombre d'helicòpters



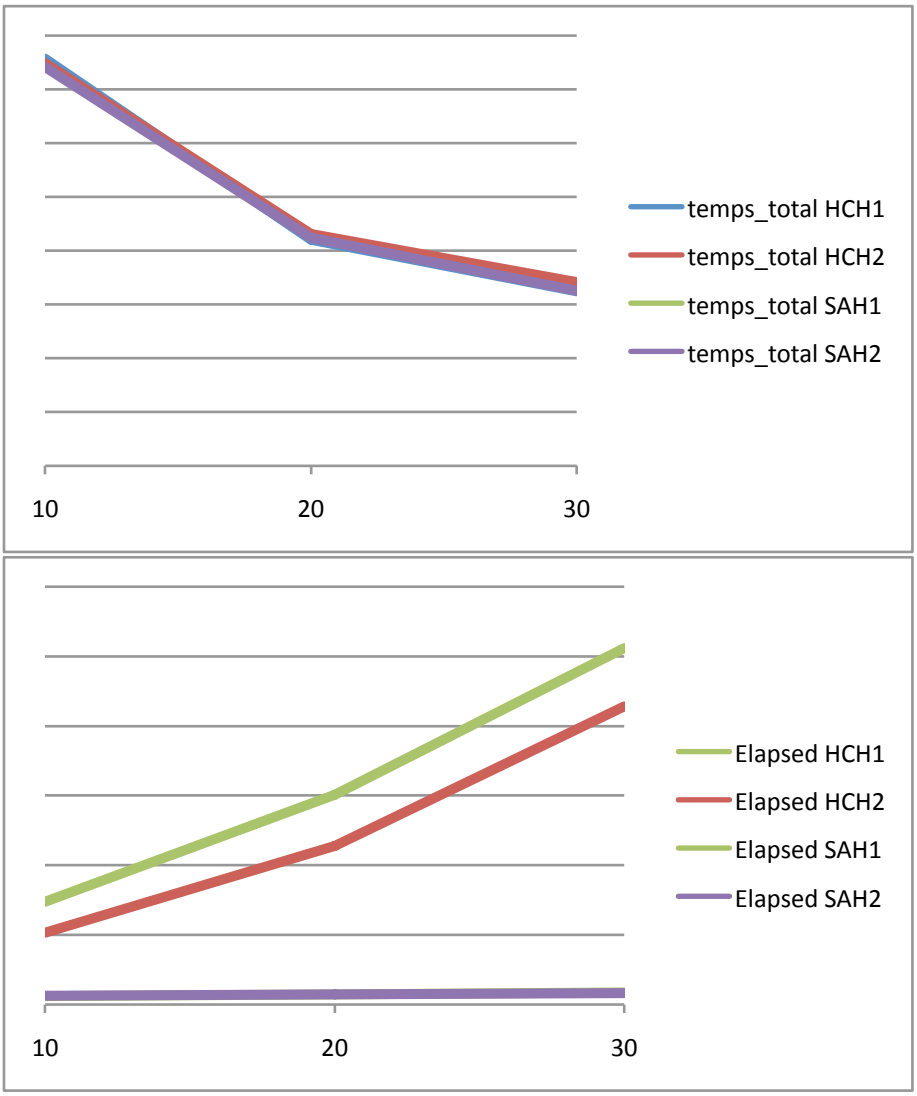
Gràfic 6: Comparativa dels temps totals amb ambdos heurístics



Gràfic 7: Comparativa dels temps d'execució amb ambdos heurístics

Aquest és l'experiment que ens ha donat resultats més sorprenents: esperàvem que el elapsed time al executat l'algorisme amb tots dos heurístics fos molt similar; i encara que el de l'heurístic 2 és lleugerament inferior, el resultat és més o menys el que esperàvem. No obstant, la sorpresa ha vingut quan observem que temps_total és exactament igual per a heurístic 1 i heurístic 2.

6.6 Comportament amb HC i SA: heurístic 1 i 2 a mesura que augmentem els helicòpters



Gràfic 8: Comparativa dels temps d'execució i temps total amb ambdos algortimes i funcions heurístiques

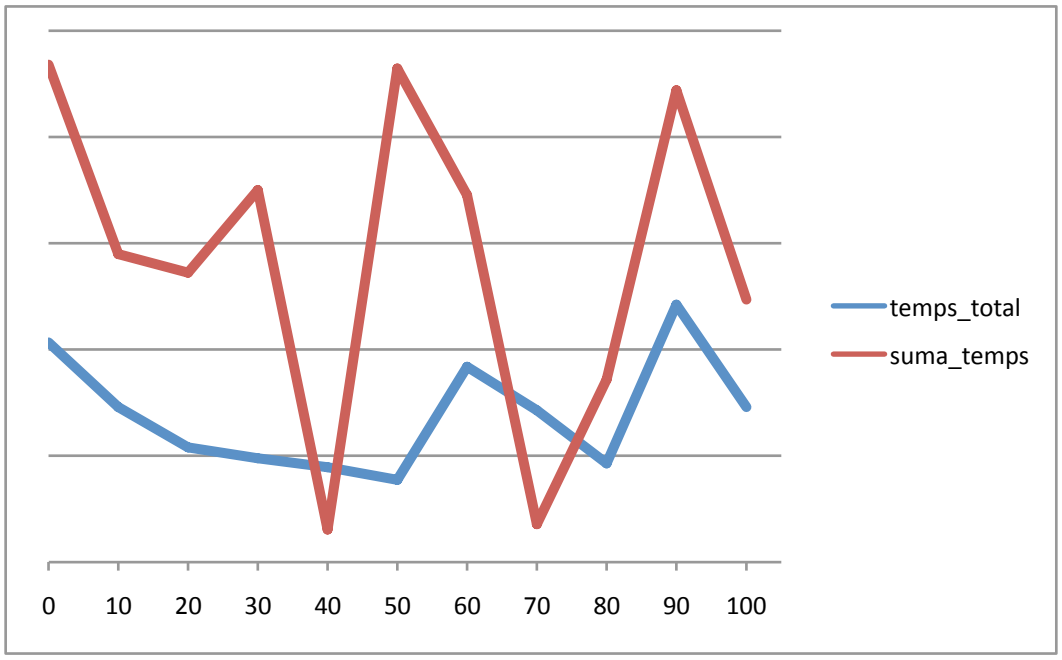
Els resultats amb temps_total de tots dos algoritmes i tots dos heurístics són molt similars; això probablement es deu a que els nostres operadors són molt potents, i poden igualar el funcionament del SA en HC. No obstant, el elapsed time de HC és molt pitjor que el de SA: i a més, la tendència és que a mesura que augmentem el nombre d'helicòpters, la distància es fa més gran. En aquest sentit, SA és molt preferible.

Experiments adicionals

6.7 Nou heurístic

Estimar el millor valor de p [0,1] amb 200 grups i 10 helicòpters i Hill Climbing on:

$$h(x) = (p \times Tiempo\_prioridad\_1) + ((1 - p) \times Tiempo\_prioridad\_2)$$



Gràfic 9: Progressió del temps total i la suma dels temps a mesura que augmenta el valor p

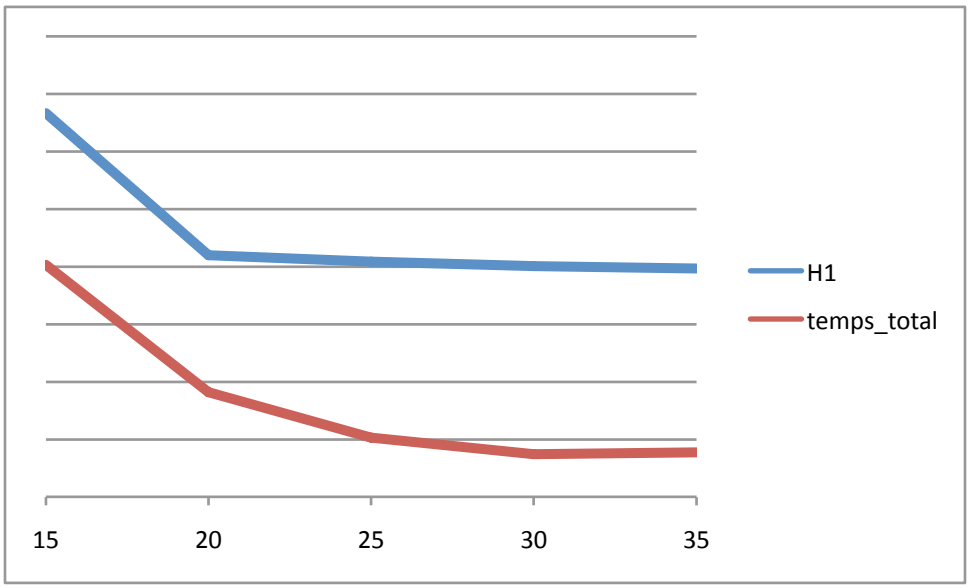
temps_total és el temps total de rescat, és a dir, el màxim dels temps de cadascun dels helicòpters.

suma_temps és la suma dels temps parcials de cada helicòpters, que permet fer-nos una idea de la quantitat de "benzina" que s'ha consumit en el rescat, és a dir, la distància recorreguda per tots els helicòpters, factor que també optimitza l'heurístic 1.

El temps total de rescat disminueix de manera significativa quan el valor de *p* es troba entre **0.2** i **0.5**. Cal tenir en compte que el factor d'aleatorietat en els resultats ja que comptabilitzem el *temps_prioritat_1* com la suma dels temps de les sortides on hi ha un grup de *prioritat 1*.

6.8 Estimar a partir de quin valor d'helicòpters no obtenim millora significativa del H1

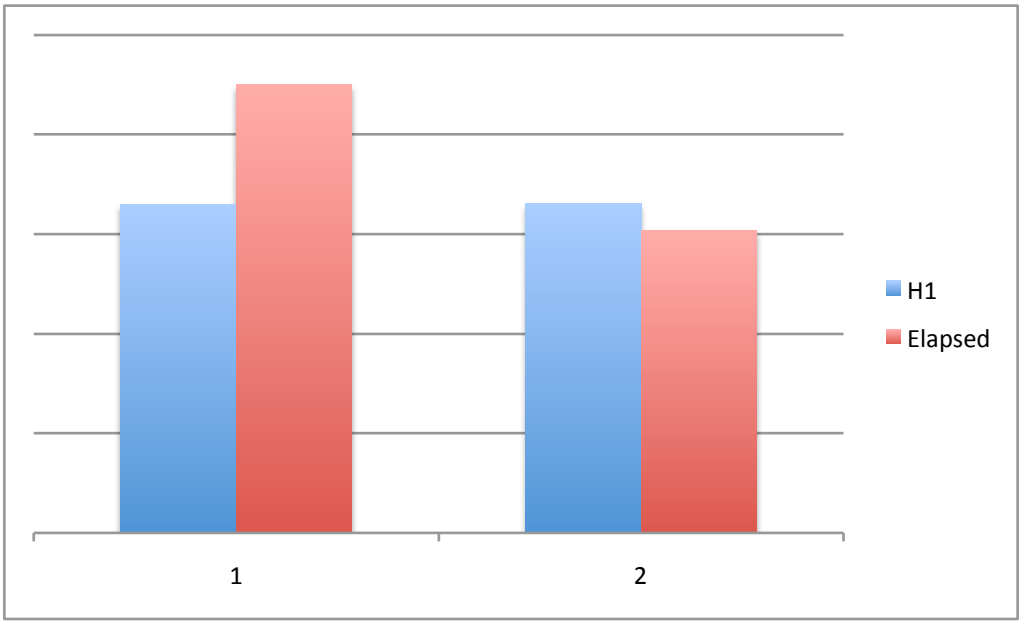
Per dur a terme aquest experiment hem anat augmentant de 5 en 5 la capacitat de l'helicopter tal com s'indicava a l'enunciat.



Gràfic 10: Progressió dels temps total a mesura que augmenta la capacitat dels helicòpters

A partir d'una capacitat 20 persones per helicòpter no s'obté una millora significativa de l'heurístic 1. Pel que fa al temps de rescat total, segueix la mateixa tendència. A partir de 30 persones ambdos factors tendeixen a mantenir-se constants.

6.9 Experiment complementari d'operadors



Gràfic 11: Comparativa d'operadors

(1) Moure - Intercanvi tot a tot

(2) Moure - Intercanvi refinat (escollit)

Pel que fa al cost de l'heurístic els 2 es comporten pràcticament igual, però en canvi, si ens fixem en el temps d'execució és demostra clarament que l'operador refinat és molt més ràipid que l'altre, és a dir, per una mateixa bondat de l'heurístic obtenim una millora en temps molt significativa. És per aquest motiu, que hem decidit escollir l'operador *Moure-Intercanvi refinat* com l'operador a utilitzar en tots els experiments.

Hem de recordar que el conjunt d'operadors escollits no podrà superar certs màxims locals -en el nostre cas, el problema serà per moure grups a posicions que no siguin les primeres d'una sortida-; però, tot i així, creiem que val la pena perdre aquesta petita millora de l'heurístic per reduir els temps.

7. Conclusions

Després de tota la feina que s'ha fet per implementar la cerca, calibrar els diferents paràmetres i observar el comportament de tots els escenaris proposats, hem pogut extreure les següents conclusions:

Creiem que els nostres operadors, tot i no ser perfectes de cara al Hill Climbing -en el cas de moure també hauríem d'afegir la possiblitat de posar un grup enmig d'una sortida i no només al inici- són sobradament potents per donar-nos solucions molt òptimes. A la secció d'experiments, a més dels demanats per l'enunciat, hem afegit un **experiment extra** que demostra que el nostre conjunt d'operadors és prou òptim.

Pel que fa a l'algorisme, entre Hill Climbing i Simulated Annealing ha sortit guanyador l'algorisme de Hill Climbing, tot i que amb resultats molt similars als de Simulated Annealing. Això segurament es deu a que tenim una heurística bastant potent, a més de comptar amb uns operadors que minimitzen els màxims locals; no obstant, recomanariem fer servir Simulated Annealing amb valors molt alts -a partir de 100 grups i 10 helicòpters- perquè el *elapsed time* és molt inferior.

Dins de la disputa entre algorismes, la recerca de paràmetres òptims per l'algorisme de Simulated Annealing ha estat costosa, tot i que concloent. Aquest fet pot fer descartar aquest algorisme quan no es t´ la capacitat de fer un an`àisi acurat com el que s'ha dut a terme en aquest document.

Experimentant amb els heurístics ens hem trobat amb la principal sorpresa de tot aquest procés d'experimentació: que el temps màxim de tot dos heurístics sigui molt similar, comportant-se tots dos de manera similar. Probablement això és conseqüència de dues coses: en primer lloc, el més important per a tots dos heurístics és el temps màxim -que tots els grups estiguin rescatats com abans millor-; en segon lloc, té a veure que al marcar com a prioritaria tota sortida que inclogui un sol grup prioritari, marquem un nombre de grups molt superior al terç inicial.

Pel que fa a les solucions inicials hem observat que la seva importància no és gaire gran pel que fa al resultat; el nostre algorisme de Hill Climbing és molt potent -té com a "local" gairebé tot l'espai de solucions- i per tant no és gaire important on ens col·loca d'inici; en canvi sí hem pogut observar que ens pot servir per accelerar lleugerament l'algorisme de cerca.

8. Funcionament de la pràctica

A l'executar el programa es mostra un menú on haurem d'escollir entre quatre tipus de demostracions. Seguidament haurem d'introduir el valor de la demostració a realitzar, el nombre d'helicòpters, el nombre de grups a rescatar, i per últim se'ns donarà la opció de que el programa ens mostri el "rastre" del conjunt de moviments que s'han efectuat des del *board* inicial.

```
┌──────────────────────────────────────────────────────────────────────────────┐
│                          Selecció de Demostracions                           │
├──────────────────────────────────────────────────────────────────────────────┤
│  Demostració 1: Hill Climbing amb Heurístic 1                                │
│  Demostració 2: Hill Climbing amb Heurístic 2                                │
│  Demostració 3: Simulated Annealing amb Heurístic 1                          │
│  Demostració 4: Simulated Annealing amb Heurístic 2                          │
└──────────────────────────────────────────────────────────────────────────────┘
Introdueix el valor de la demostració a realitzar (entre 1 i 4)
1
Introdueix el nombre d'helicòpters de rescat (entre 1 i 10)
2
Introdueix el nombre de grups a rescatar (entre 1 i 100)
10
Voleu veure el rastre de moviments de l'algoritme? (S/N)
S
```

Un cop finalitzi la demostració es mostrarà per pantalla una taula amb el conjunt de paràmetres que ve a continuació:

Demostració	HF	Temps_total	Suma_temps	Elapsed	Steps
HC(10,2) HF1	748,87	241,997417	409,163530	74	3

Si em decidit que se'ns mostri el rastre de moviments se'ns demanarà que premem qualsevol tecla i a continuació es mostrarà el resultat. Primerament s'imprimirà per pantalla l'estat inicial i seguidament la resta de moviments.

Com es pot comprovar en la imatge següent, es mostra el tipus de moviment que s'ha efectuat, ja sigui intercanvi o moviment, i després el conjunt d'helicòpters, cadascun amb la seva llista de sortides. Cada sortida està formada per una consecució de grups del tipus *id(num_pers),* on *id* és l'identificador del grup en qüestió i *num_pers* el nombre de persones que el comformen.

Just després de les sortides es mostra el nombre de persones per cadascuna de les sortides de manera que es pugui comprovar ràpidament que en cap moment s'estan violant les restriccions imposades per l'enunciat. Per acabar es mostren els nodes expandits per l'algoritme.

```
Premeu una tecla per mostrar el rastre de moviments..

Helic.0: => 7(10) => 2(4) => 8(1) => Base => 9(6) => 4(9) => Base => 5(9) => Base
Helic.1: => 6(5) => 0(4) => 1(6) => Base => 3(11) => Base
Sortides H[0]: 15;15;9;
Sortides H[1]: 15;11;

Intercanvi del grup(0) de la sortida(0) del h(0) amb el grup(0) de la sortida(2) del h(0) | Coste(803.0842368062463)
Helic.0: => 5(9) => 2(4) => 8(1) => Base => 9(6) => 4(9) => Base => 7(10) => Base
Helic.1: => 6(5) => 0(4) => 1(6) => Base => 3(11) => Base
Sortides H[0]: 14;15;10;
Sortides H[1]: 15;11;

Moviment del grup(2) de la sortida(0) del h(0) al grup (0) de la sortida(1) del h(1) | Coste(750.1879443191866)
Helic.0: => 5(9) => 2(4) => Base => 9(6) => 4(9) => Base => 7(10) => Base
Helic.1: => 6(5) => 0(4) => 1(6) => Base => 8(1) => 3(11) => Base
Sortides H[0]: 13;15;10;
Sortides H[1]: 15;12;

Intercanvi del grup(1) de la sortida(0) del h(0) amb el grup(0) de la sortida(0) del h(1) | Coste(748.8727632632633)
Helic.0: => 5(9) => 6(5) => Base => 9(6) => 4(9) => Base => 7(10) => Base
Helic.1: => 2(4) => 0(4) => 1(6) => Base => 8(1) => 3(11) => Base
Sortides H[0]: 14;15;10;
Sortides H[1]: 14;12;

nodesExpanded : 4
```