Pràctica de Cerca Local

Distribució de connexions dins d'una Xarxa de Sensors

Grau IA

Albert Pedraza - Miguel Angel Serrat - Sergio Preciado

Curso 2020/2021 2Q
Grau en Enginyeria Informàtica - UPC
Departament de Ciències de la Computació

Índex

Identificació del problema

Descripció del problema

Existeix un recurs distribuït geogràficament el qual volem obtenir. Aquest recurs és obtingut per un conjunt de *Sensors*. Realitzant interconnexions entre ells, aquests necessiten fer arribar el recurs fins als *Centres de Dades* per poder guardar la informació obtinguda.

Els elements que componen la tasca a realitzar són els següents:

Àrea Geogràfica: Correspon a un terreny 100 x 100 km², on tenim 10^4 posicions per distribuir els diferents *Sensors* i *Centres de Dades.*

Sensor: Són elements capturadors d'informació. Es poden distribuir per les diferents posicions que constitueix l'àrea geogràfica. Existeixen 3 tipus diferents de sensors, que només varien en la quantitat d'informació capturada:

- Capturadors d'1 Mb/s
- Capturadors de 2 Mb/s
- Capturadors de 5 Mb/s

A més, un sensor té la capacitat de transmetre informació, mitjançant l'establiment d'una connexió de sortida. Aquesta transmissió es duu a terme a l'instant, és a dir, en el mateix moment en què realitza la captura d'aquesta.

Un sensor també pot rebre connexions d'entrada, tenint 3 com a nombre màxim d'aquestes. S'ha de tenir en compte que la capacitat de transmissió d'un sensor és de com a màxim 3 vegades el volum d'informació que captura.

Si ens fixem en la quantitat de dades que pot rebre i transmetre un sensor, ens dóna origen a la següent taula:

	1 Mb/s	**2 Mb/s**	**5 Mb/s**
Storage (min - max)	0 - 2	0 - 4	0 - 10
Transmissió (min - max)	1 - 3	2 - 6	5 -15

Centre de Dades: Són els elements que guarden la informació capturada i transmesa per part dels sensors. Es poden distribuir per les diferents posicions que constitueix l'àrea geogràfica.

Tenen la capacitat de rebre com a màxim 25 connexions d'entrada, sense superar l'ample de banda de 150 Mb/s.

Anàlisi de les característiques del problema

El problema a resoldre tracta d'aconseguir una distribució de connexions entre sensors i centres on obtenim tota la informació disponible tenint el cost mínim entre connexions.

El cost d'una connexió ve definida per la distància euclídea entre un parell de sensor a sensor/centre connectats, elevada al quadrat, multiplicada per el volum de dades transmès.

$$d\,(x, y) = \sqrt{(xi - yi)^2 + (xj + yj)^2}$$

$$cost(x, y) = d(x, y)^2 * v(x)$$

És important tenir en compte que tota la informació capturada en un segon és transmesa simultàniament per totes les connexions fins arribar al punt final que són els Centres de Dades.

Una possible solució al problema, és a dir, una possible distribució de les connexions ha de contenir que tots els sensors estiguin connectats a tots els Centres de Dades sigui d'una forma directa o indirecta.

A més, tots els sensors transmeten tota la informació capturada i rebuda sense possibilitat d'emmagatzematge per un període de temps. Si la quantitat d'informació que envia un sensor no pot ser transmesa pel sensor/centre posterior estaríem perdent informació que igualment seria transmesa i què en sumaria cost a la nostra solució. Per tant, en la nostra xarxa de connexions, no admetrem aquest tipus d'escenaris.

Descripció dels elements de l'Estat del problema

L'estat del problema a tractar necessita poder guardar informació sobre quins sensors es connecten a quins sensors/centres. És a dir, ha de suposar una representació de la nostra xarxa de connexions, seguint les restriccions esmentades.

Per què fem servir Cerca Local?

La raó per la qual utilitzem cerca local per solucionar aquest problema és per la naturalesa d'aquest.

Donada una distribució aleatòria de sensors i centres, amb les seves característiques i localitzacions, és impossible recórrer totes les possibles distribucions i quedar-nos amb la millor. Això esgotaria la memòria de qualsevol ordinador convencional. Necessitem una tècnica que ens permeti, donada una distribució inicial correcta i uns operadors, moure-n's per dins l'espai de solucions sense recorre'l tot, decidint a cada iteració quin serà el pròxim estat en què visitarem tots els veïns possibles. Aquest procés es repeteix fins arribar a la impossibilitat de millorar la solució actual.

Representació de l'Estat del problema

Descripció de la representació del problema

L'estat és una solució del problema, que no té per què ser la millor. Per tant és una representació de tots els components que podem trobar-nos en el problema, perquè el programa en generi una solució més òptima.

Per tant el nostre estat es basa en les diferents connexions dels sensors i els centres. Per tant repliquem les llistes de sensors i els centres creant una "extensió" de la classe que ens mostri les seves connexions.

Representació del problema

L'estat queda representat pels següents elements:

static Sensores sensores: Objecte de la classe sensores que representa el conjunt de sensors que hi ha a l'àrea, amb la seva posició i capacitat.

static CentrosDatos centros: Objecte de la classe CentrosDatos que representa el conjunt de centres que hi ha a l'àrea, amb la seva posició.

private ArrayList<ConnexSensor> connexSList: Un arraylist de la classe ConnexSensor que agafa les connexions de cada un dels sensors i les relaciona amb altres sensors o altres centres.

public class ConnexSensor: Classe que consta de la següent informació:

private ArrayList<Integer> connectionIn: Un arraylist d'enters de tres posicions on cada un d'aquests enters simbolitza les connexions d'entrada a aquell sensor.

private int connectionOut: És un enter que marca la connexió de sortida del sensor a on si és negatiu representa una connexió a un centre i si és positiva és a un altre sensor.

private Boolean isFree: Un boolean que indica si es pot connectar un altre sensor al sensor actual.

private Double transmission: Un double que guarda la capacitat de transmissió que l'hi arriba pels inputs i que enviarà per l'output.

private ArrayList<ConnexCentro> connexCList: Un arraylist de la classe ConnexCentro que representa totes les connexións del centre i totes les capacitats que rep dels diferents sensors.

public class ConnexCentro: Classe que consta amb la següent informació:

private ArrayList<Integer> connectionIn: Un arraylist que té les connexions d'entrada al centre.
private Boolean isFree: Un boolean que indica si es pot connectar un altre sensor al centre.
private Double reception: Un double que registra tota la capacitat que arriba al centre des de tots els sensors que té connectats.

private Double coste: Un double que enregistra el cost total de la solució proposada.

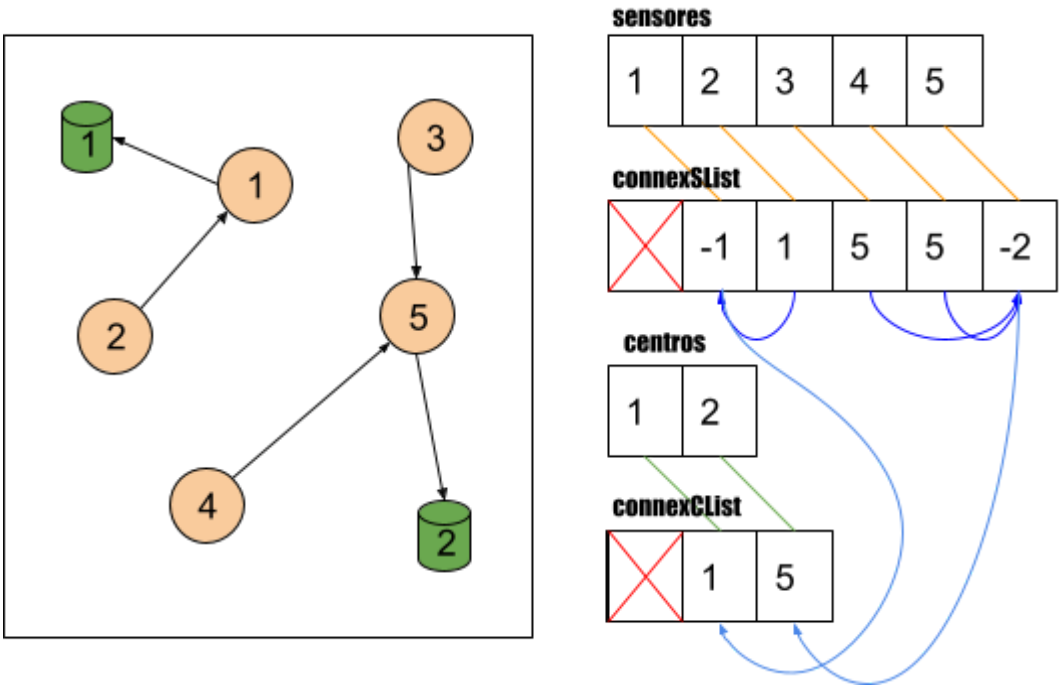

Fig. 1 - Representació a baix nivell de la Xarxa UI, de l'array de Sensors, Centres i les seves connexions

Anàlisi de la mida de l'espai de cerca

La mida de l'espai de cerca és **O((S+C)^S)** on

- S és el nombre de Sensors
- C és el nombre de Centres de Dades

Representació dels operadors

Descripció dels operadors

Un operador agafa l'estat en què es troba la iteració actual de l'algorisme de cerca, i el modifica creant un estat successor.

Els operadors que a primer cop d'ull semblen plausibles són els de modificar una connexió qualsevol, modificar la capacitat d'un sensor, modificar la posició d'un sensor, fer *swap* entre dues connexions actuals, eliminar una connexió, introduir una nova connexió, etc.

Concretament aquests dos últims fàcilment ens traurien i ens tornarien a introduir en l'espai de solucions. No els contemplem i per tant, els descartem, per no complir les restriccions de la nostra solució.

Per tant, escollim l'operador únic **Modificar Connexió de Sortida d'un Sensor.**

Entre els altres, creiem que la generalització de tots ells es tracta d'un operador que elimina la connexió actual d'un sensor i en realitza una de nova amb un sensor/centre. Utilitzarem també diferents limitacions dins l'operador que ens suposaran no sortir del ventall de solucions. Aquestes limitacions es tracten, entre altres, de mantenir la informació màxima transmesa, impossibilitat de què les dades rebudes i generades per qualsevol sensor no arribin a cap centre de dades, i compliment de les diferents mesures imposades per part del problema.

Mitjançant aquest podem explorar qualsevol solució possible dins l'espai de solucions i complir les restriccions i necessitats del nostre problema.

Anàlisi del factor de ramificació

La ramificació de l'operador seleccionat és **O(S+C)** on:

- S és el nombre de Sensors
- C és el nombre de Centres de Dades

Ens indica que per cada node en l'espai de cerca, existeixen O(S+C) nodes successors. És a dir, per un estat actual qualsevol, l'operador modificarà la connexió d'un sensor donant-li la possibilitat de connectar-se amb qualsevol altre sensor o centre.

Justificació dels operadors

El nostre operador tindrà implementades diferents limitacions que faran impossible no complir les diferents restriccions del problema i, a més, realitzaran la impossibilitat de perdre informació, operant sempre amb la màxima possible.

Aquestes restriccions imposades es descriuen per la impossibilitat de pèrdues d'informació, i pel compliment de les restriccions de nre. de connexions d'entrada de cada sensor i centre de dades, com el compliment de les amplades de banda corresponents, i la no-possible connexió d'un sensor amb ell mateix.

Concretament, si existeix la possibilitat que, la nova connexió en potència de ser creada per part de l'operador suposa la pèrdua de dades per part de qualsevol sensor o centre de dades posterior, aquesta modificació en la xarxa de connexions no es realitzarà, és a dir, l'operador fallaria i no s'inclouria l'expansió en l'espai de cerca.

Cal remarcar que per pèrdua de dades, també tenim en compte que tota informació generada pels sensors arribarà a un centre de dades, és a dir, no hi haurà cap illa de sensors en la nostra distribució.

Anàlisis de la funció heurística

Descripció dels factors que intervenen en l'heurística

La funció heurística d'aquest problema ha de suposar la minimització del cost total o sumatori de les diferents connexions que la nostra xarxa distribueix.

$$d\,(x, y) = \sqrt{(xi - yi)^2 + (xj + yj)^2}$$

$$cost(x, y) = d(x, y)^2 * v(x)$$

Segons els nostres criteris aplicats sobre l'espai de solucions, no ens cal realitzar la maximització de la quantitat d'informació transmesa fins als Centres de Dades, ja que utilitzem limitacions en els nostres operadors i en la producció de solucions inicials que realitzen aquesta tasca i mantenen la quantitat d'informació transmesa sempre al màxim.

Per tant, necessitem una heurística que ens retorni el cost total de les connexions que estan distribuïdes en la xarxa.

Justificació de les heurístiques escollides

La suma del cost de les diferents connexions dels diferents sensors de la xarxa, és una heurística correcta per córrer en els nostres algorismes de cerca perquè ens proporcionarà la minimització del resultat d'aquesta.

Descripció dels efectes de l'heurística en la cerca

Els nostres algorismes de cerca, a cada iteració, decidiran sobre el cost total de totes les connexions després d'haver aplicat l'operador corresponent. Concretament, volem minimitzar el resultat de la funció heurística, i per tant, és justament el que es farà.

Anàlisis dels estats inicials

Hem fet servir 3 algoritmes diferents per a la generació de les nostres solucions inicials. Al codi els hem anomenat:

- *solucioInicial1*() `// “Random”`
- *solucioInicial2*() `// Connexions directes a Centres + Dist. // mínima entre S i C + Sensors restants (no // connectats a C) iterant “backwards”`
- *solucioInicial3*() `// Connexions directes a Centres + Dist. // mínima entre S i C + Sensors restants (no // connectats a C) iterant “forwards”`

Tots aquests algorismes tenen en compte les restriccions que planteja el problema per a definir una “solució” (no es perden de dades i tots els sensors arriben a un centre de dades).

Tot i que a simple vista una de les solucions inicials ens pugui semblar millor que la resta, cal experimentar amb aquestes i fer una anàlisi de costos i temps, ja que és possible que, per exemple, la solucioInicial1() “random” (amb més espai i capacitat de millora) ens retorni millors resultats que una solució inicial més tancada i concreta (Amb probabilitats de caure ràpidament a un mínim local).

És per això que cal definir i identificar correctament els estats inicials i la seva generació:

- ***solucioInicial1*:** No ordenem els centres o sensors que es creen a partir de les seeds. Tal com es generen aquestes dues Arrays, s’itera per la dels Sensors (de 0 a nSensors) connectant cada sensor al primer Centre disponible de l’Array de Centres (de 0 a nCentres).

  Els sensors restants que no han pogut connectar-se a un Centre, tornen a iterar per l’Array de Sensors, connectant-se així al primer Sensor lliure que troben.

- ***solucioInicial2*:** Ordenem les Arrays de Sensors i Centres que es generen a partir de les seeds segons la seva distància respecte al punt 0,0 de la Xarxa. Els Sensors i Centres **s’ordenen de menys distància a més distància**, per tant els primers 25 Sensors de l’Array estaran més a prop del primer Centre, els 25 següents a prop del segon centre, i així successivament.

  Els sensors restants que no han pogut connectar-se a un Centre, tornen a iterar per l’Array de Sensors en aquest cas modificant la direcció de la iteració, de nSensors a 0, per tal de connectar els nodes restants als sensors més llunyans al 0,0 (els més propers a ells), connectant-se així al primer Sensor lliure que troben.

- ***solucioInicial3*:** Igual que la solució inicial 2, però els sensors restants que no han pogut connectar-se a un Centre, tornen a iterar per l'Array de Sensors (de 0 a nSensors), connectant-se així al primer Sensor lliure que troben.

A la primera solució inicial, com que els Arrays no estan ordenats, les connexions es fan de manera "random" tot i no ser aleatòria del tot. Simplement es connecten els sensors tal com s'han generat (en posicions aleatòries en la xarxa) als primers centres que troben lliures (també en posicions aleatòries en la xarxa). D'aquesta manera aconseguim una distribució amb més entropia i així a partir d'una solució "aleatòria" podem arribar a un conjunt de solucions que no descobriríem amb una solució inicial més tancada o processada.

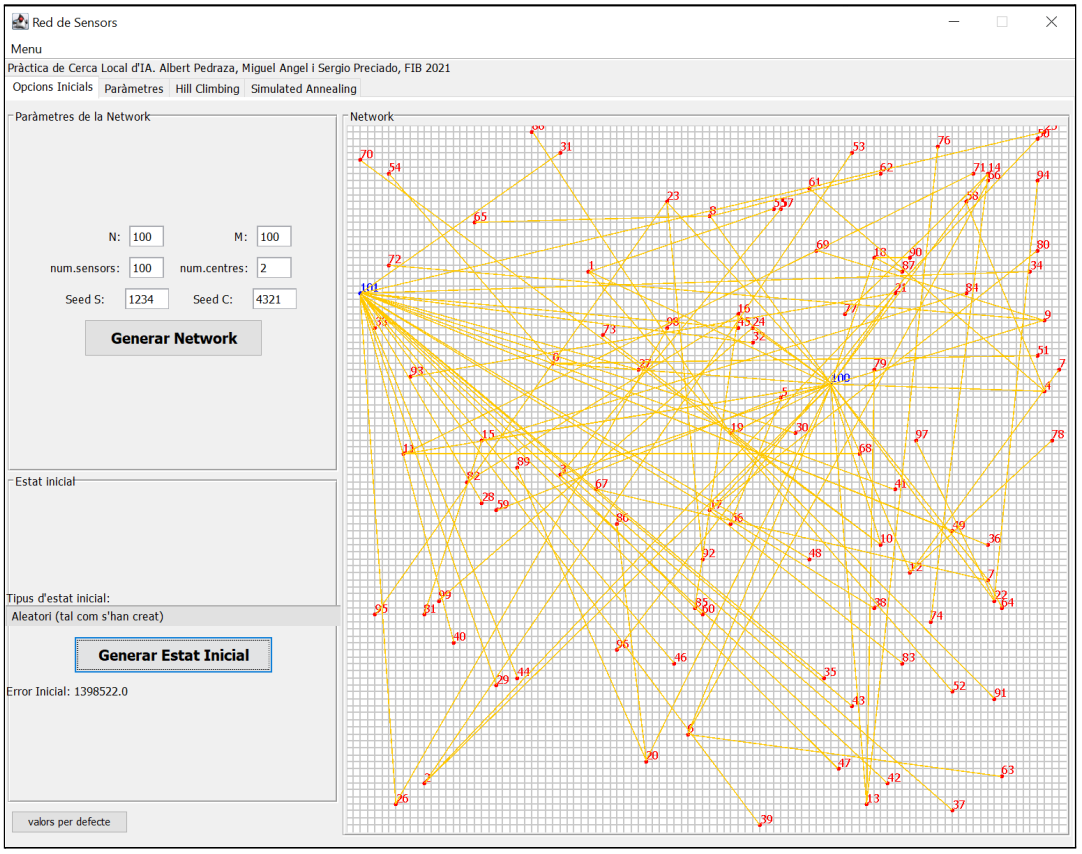

Fig. 2 - Xarxa de la Sol.Inicial 1 amb 100 Sensors i 2 Centres

La segona i tercera solució són semblants, l'única diferència és la distribució de les connexions de Sensors restants. En aquestes dues solucions podem observar una distribució de connexions que a simple vista podríem catalogar com a "millors" solucions i, efectivament, les solucions inicials que generen tenen menys cost, però per a poder acabar de determinar si les nostres solucions són profitoses per a l'algoritme, haurem d'experimentar amb aquestes.

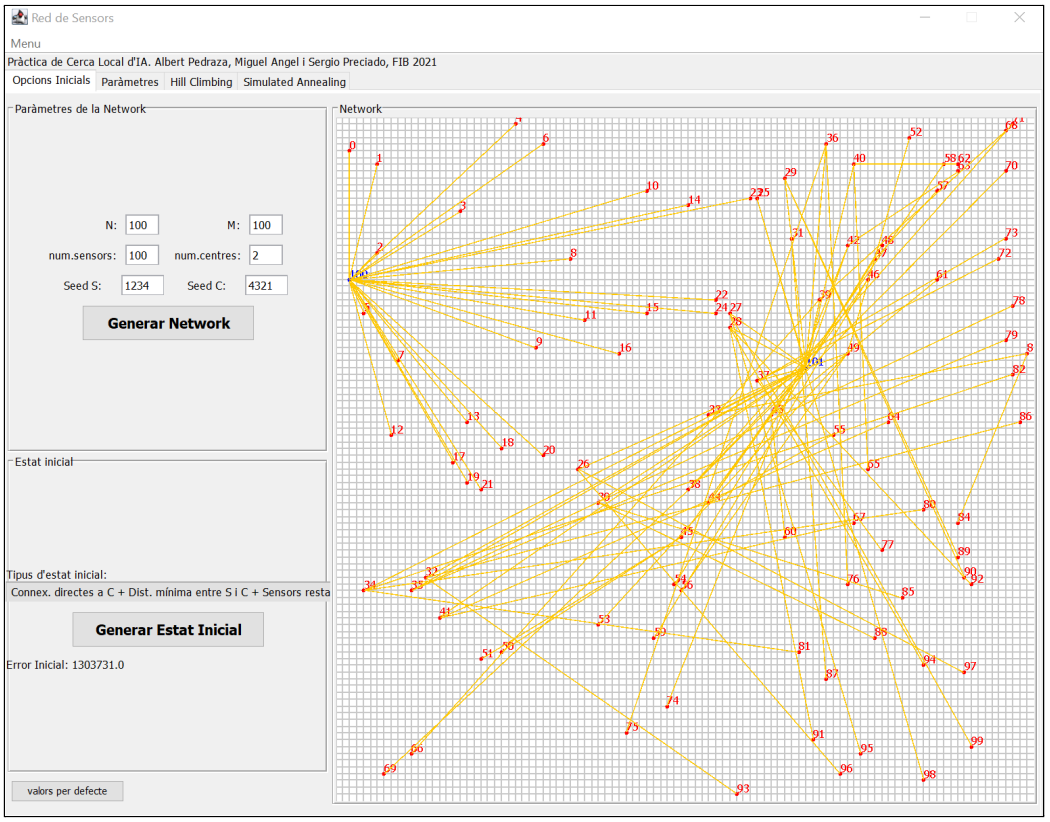

Fig. 3 - Xarxa de la Sol.Inicial 2 amb 100 Sensors i 2 Centres

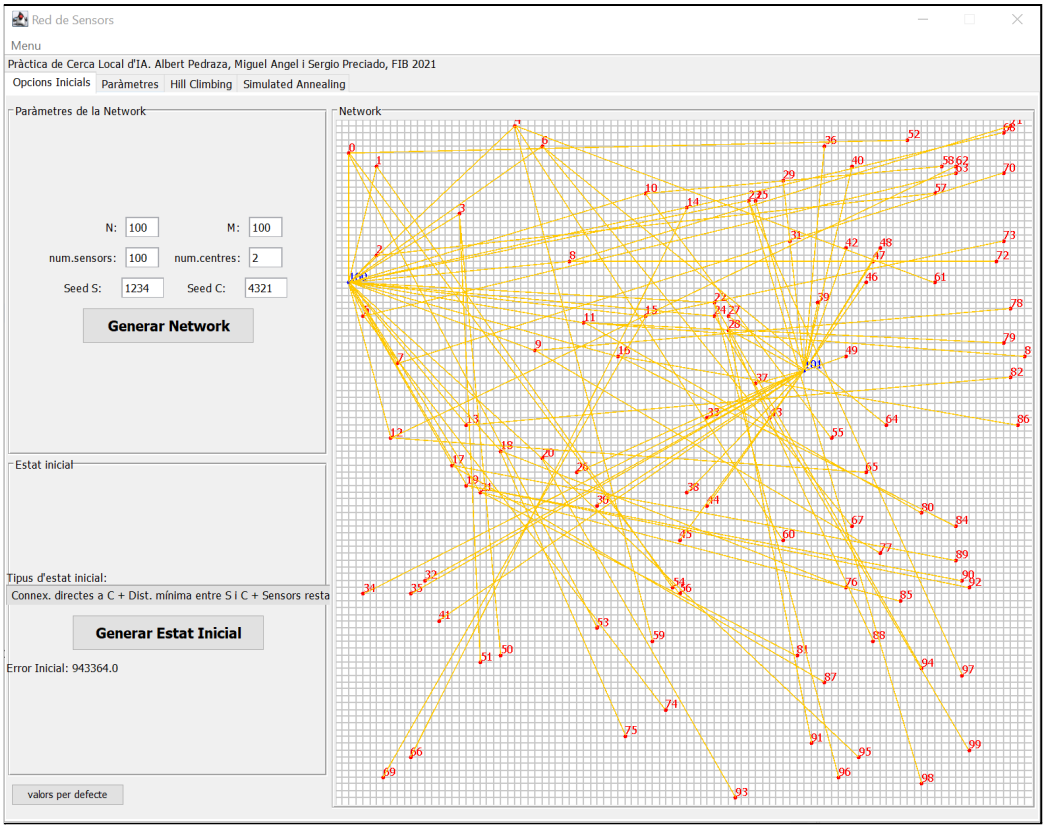

Fig. 4 - Xarxa de la Sol.Inicial 3 amb 100 Sensors i 2 Centres

Experiments

Experiment 1

1. Determinar quin conjunt d'operadors dona millors resultats per una funció heurística que optimitza el criteri de qualitat del problema (3.2) amb un escenari en què el nombre de centres de dades és 4 i el de sensors és 100. Heu de fer servir l'algorisme de Hill Climbing.

En el nostre cas, des d'un punt inicial hem fet l'elecció d'un sol operador que ens permet qualsevol solució possible i, utilitzant limitacions dins d'aquest i amb la funció heurística, optimitzem perfectament els criteris de qualitat de l'enunciat.

Aquest té la possibilitat d'eliminar qualsevol connexió d'un sensor, realitzant-ne llavors una de nova. Aquest canvi només serà efectiu si és òptim, és a dir, si mitjançant la funció heurística, els algorismes de cerca trien aquest estat modificat per l'operador per continuar amb les iteracions posteriors. Dit d'una altra manera, minimitza el cost de les connexions, representat per la funció heurística, i a la vegada ens manté la maximalitat de transmissió de dades, sent capaços d'explorar qualsevol solució possible.

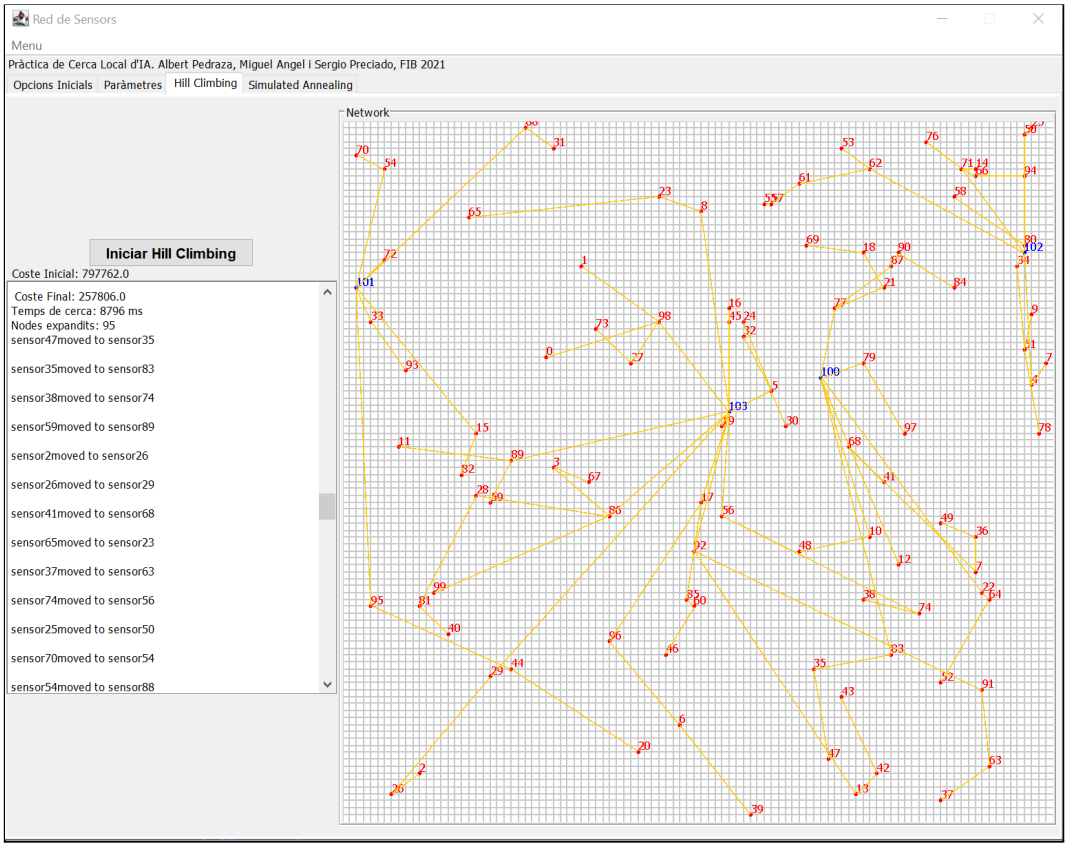

Fig. 5 - Solució del HillClimbing amb Sol.Inicial 1

Experiment 2

2. Determinar quina estratègia de generació de solució inicial dona millors resultats per la funció heurística que hem fet servir en l'apartat anterior, amb l'escenari de l'apartat anterior i fent servir l'algorisme de Hill Climbing. A partir dels resultats heu de fixar també l'estratègia de generació inicial per la resta d'experiments.

Observació: Un dels 3 generadors de solucions inicials pot ser millor que els altres
Plantejament: Generarem solucions amb els 3 i compararem els resultats.
Hipòtesis: Els 3 generadors de solucions són iguals(H0) o un d'ells produeix millors resultats.

Mètode:

- Tenim un conjunt de 10 seeds aleatòries.
- Executem un experiment per cada seed i generador de solucions inicials.
- El nombre de sensors serà 100 per totes les execucions i 4 centres de dades.
- Farem servir l'algorisme de Hill Climbing
- Estudiarem el valor final de la funció heurística i el nombre de passos que ha fet H.C. per arribar a la solució corresponent.

COST INICIAL	**Solució 1**	**Solució 2**	**Solució 3**
	797762	556722	556722
	624484	640790	640790
	781945	418089	418089
	851838	614343	614343
	840493	670536	670536
	852173	650038	650038
	846369	802246	802246
	1013269	701696	701696
	686218	393459	393459
	883416	639879	639879
Mediana	**843431**	**640334,5**	**640334,5**
Desviació Típica	**106748,32**	**124292,42**	**124292,42**

COST FINAL	Solució 1	Solució 2	Solució 3
	257806	226284	226284
	229424	198512	198512
	216862	158078	158078
	319525	297363	297363
	241474	237657	237657
	223973	195796	195796
	432498	393497	393497
	268957	217649	217649
	207129	172257	172257
	352808	296715	296715
Mediana	**249640**	**221966,5**	**221966,5**
Desviació Típica	**72238,59**	**71284,53**	**71284,53**

PASSOS	Solució 1	Solució 2	Solució 3
	95	86	86
	90	86	86
	89	80	80
	87	79	79
	88	81	81
	95	89	89
	93	87	87
	96	78	78
	88	97	97
	92	91	91
Mediana	**91**	**86**	**86**
Desviació Típica	**3,33**	**6,02**	**6,02**

TEMPS	Solució 1	Solució 2	Solució 3
	10913	7763	9790
	7958	7553	7677
	7740	7035	7654
	7700	6952	6713
	7600	7278	7558
	8330	7935	8481
	8153	7738	7558
	8320	6779	6882
	7621	8498	7974
	8050	7971	7580
Mediana	**8004**	**7645,5**	**7617**
Desviació Típica	**978,48**	**537,08**	**861,435114**

Un cop hem recopilat les dades de l'experiment, podem crear gràfiques en format Boxplot que ens permetran poder extreure conclusions més fàcilment i d'una manera molt més visual i entenedora.

Gràfiques Boxplot de l'experiment.

En Blau: Sol.Inicial 1, en Verd: Sol.Inicial 2 i en Groc: Sol.Inicial 3.

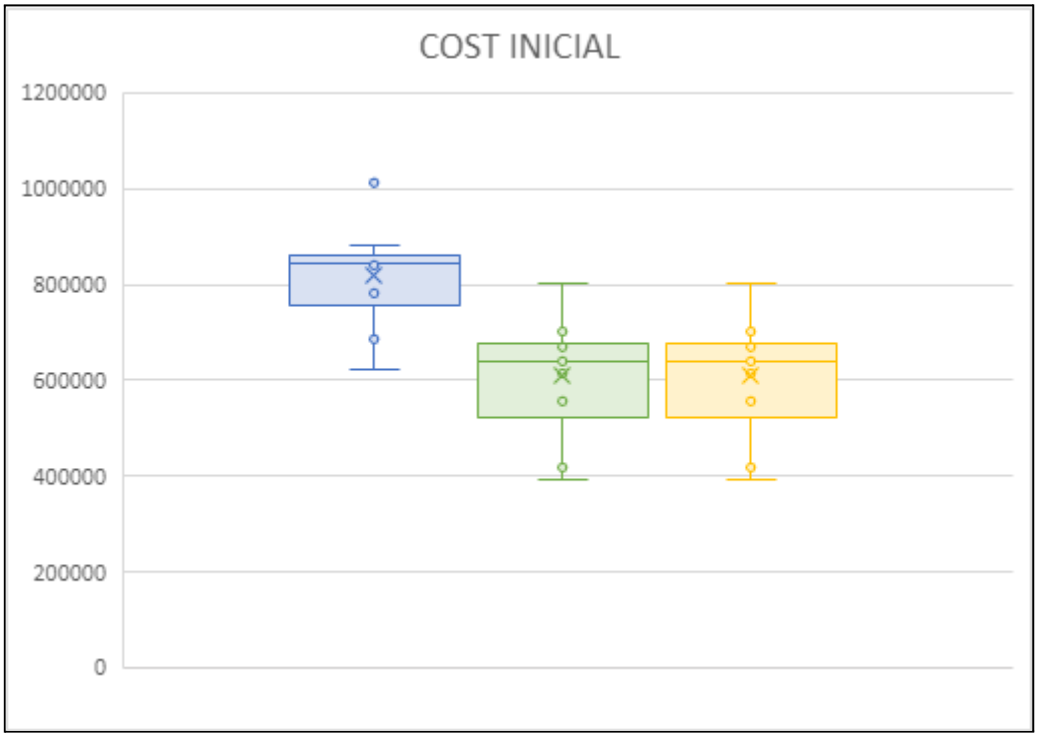

Fig. 6 - Boxplot del Cost Inicial

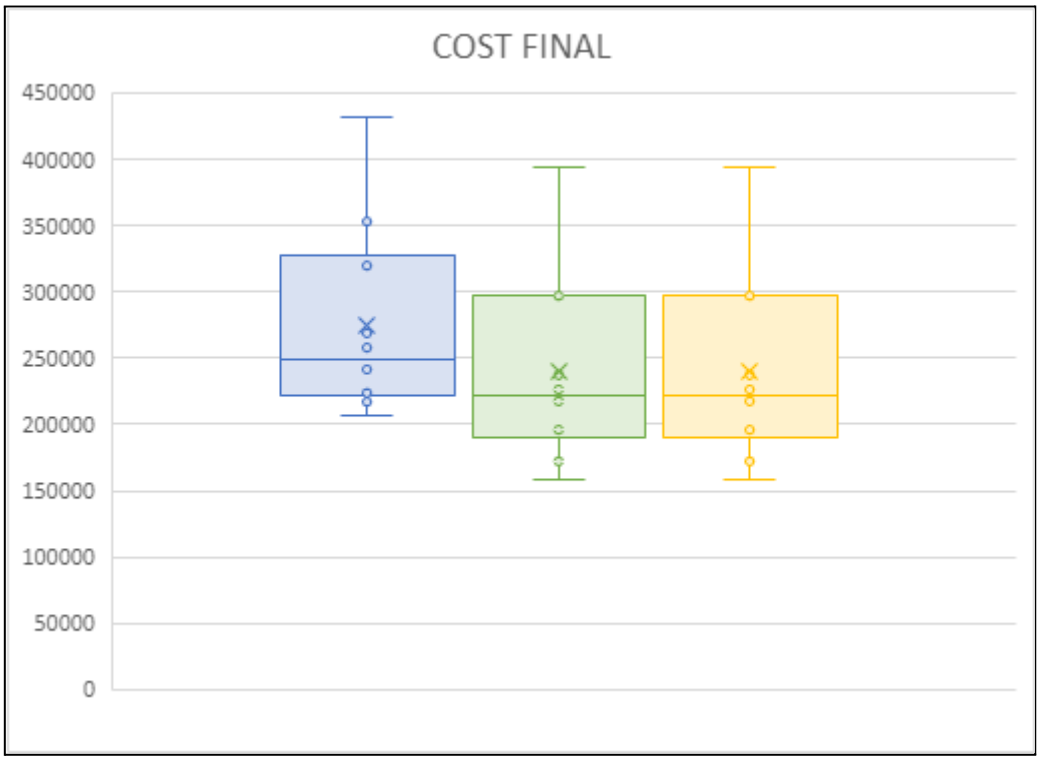

Fig. 7 - Boxplot del Cost Final

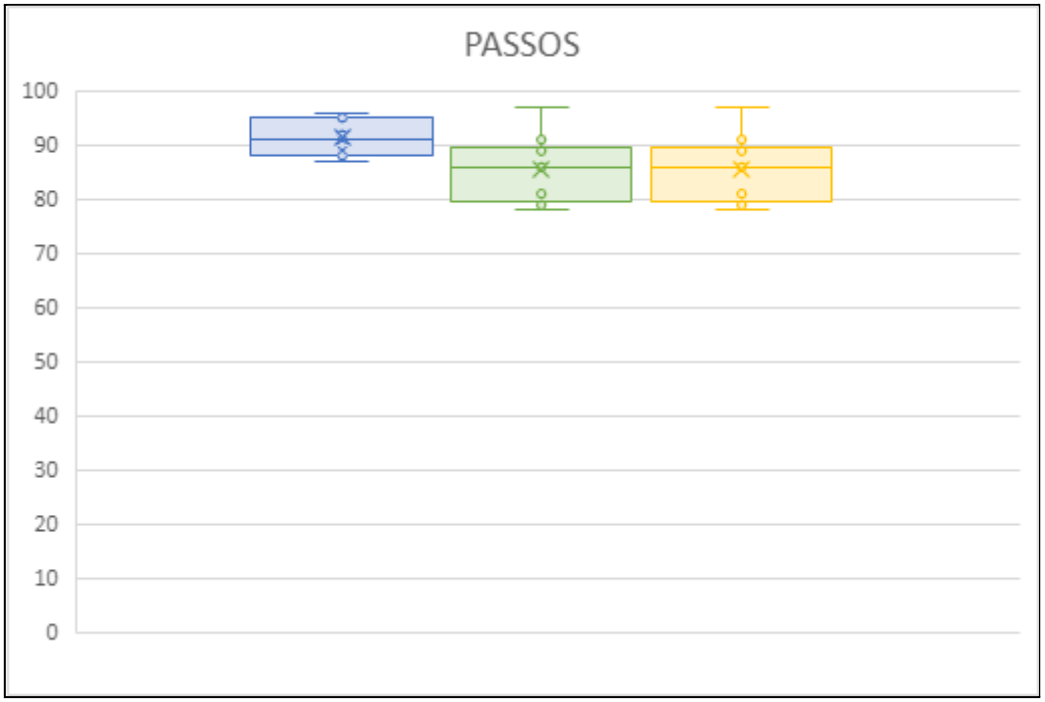

*Fig. 8 - Boxplot dels Passos o els **nodes expandits***

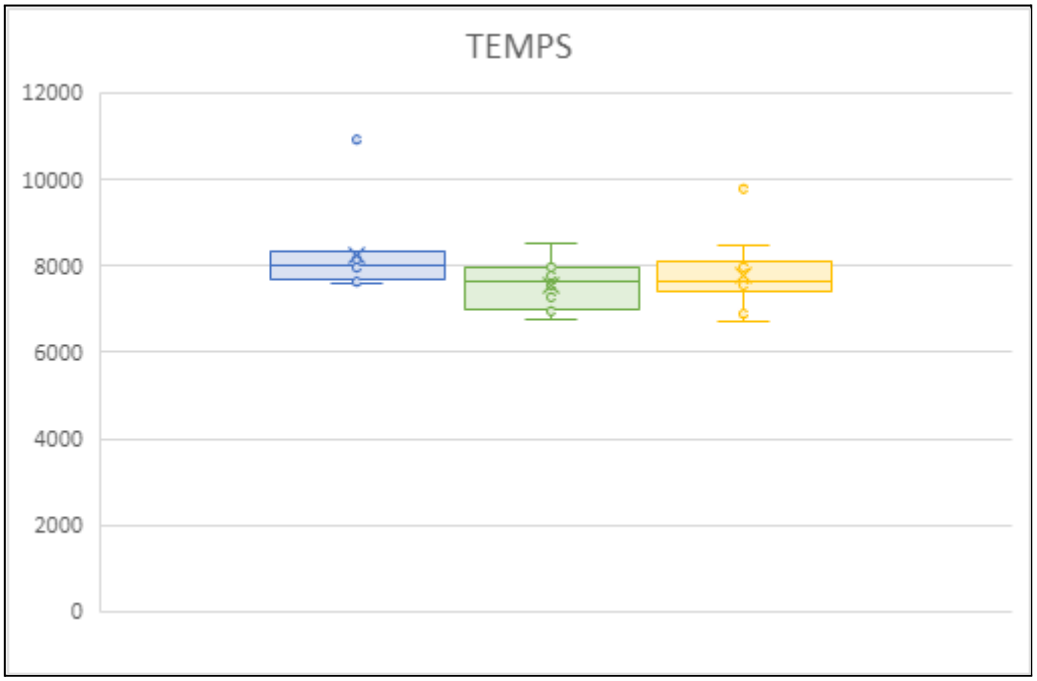

Fig. 9 - Boxplot del Temps total de l'execució

Segons les gràfiques, podem observar que **el segon i tercer generador de solucions** inicials, que connecten inicialment totes les connexions més properes possibles, **és el que ens proporciona el cost més baix en mitjana** dels diferents valors de la funció heurística de l'estat final de les diferents solucions extretes del H.C.

També podem detectar que, **per a solucions inicials amb 100 sensors i 4 centres**, tant les generacions com els resultats de **les dues generacions Sol.Inicial 2 i Sol.Inicial 3 són les mateixes**. En realitat, aquesta casuística ve del fet que la primera part dels dos algoritmes és la mateixa, així que només podrem observar diferencies quan els sensors no es puguin connectar directament a aquests últims. (És a dir, quan quedir sensors restants sense assignar).

Hem experimentat amb diferents paràmetres de HC on es complien que no tots els sensors queden assignats a un centre, p.e. nSens: 100 , nCent 2, i podem concloure que la **Sol.Inicial 2** és millor en termes de cost final i temps d'execució que la **Sol.Inicial 3.**

Tot i que **Sol.Inicial 2 i Sol.Inicial 3**, per la pròpia naturalesa del codi emprat, resulten ser també **els més costos en termes de temps i d'esforç de computació** abans de l'execució, podem argumentar que aquests valors no són gens grans i, per aquesta manera, el temps d'espera és negligible.

Podem observar en la Figura 6 que la **Sol.Inicial 1** és la que comença amb un **Cost Inicial destacablement més elevat** (per tant, amb més capacitat millora) i, així i tot, podem veure que els resultats finals no es troben gaire lluny de les solucions inicials 2 i 3, ni comporta un cost molt més elevat de temps.

Tenint en compte els apartats anteriors, podem concloure que seria convenient **descartar el generador de solucions inicials 1** i el **generador de solucions inicials 3** per ser pitjors en termes de minimització del cost total de totes les transmissions respecte al **generador de solucions inicials 2**.

Experiment 3

3. Determinar els paràmetres que donen millor resultat pel SimulatedAnnealing amb el mateix escenari, fent servir la mateixa funció heurística i els operadors i l'estratègia de generació de la solució inicial triada amb els experiments anteriors.

Observació: Hi ha d'haver una combinació de paràmetres de la funció que calcula el Simultaed Annealing que dóna millors resultats que una altra combinació per una instància de problema concreta (100 sensors i 4 centres de dades).

Plantejament: Generació de diverses solucions amb diferents paràmetres per poder acotar els diferents paràmetres corresponents. És necessari anar experimentant pas a pas i sobre diverses execucions, ja que l'algorisme realitza una tria pseudoaleatòria sobre l'estat successor a qui aplica la funció de probabilitat d'acceptació. Si no es produeix aquesta, aleatòriament tria un altre successor que tindrà la possibilitat de ser acceptat, i així successivament fins a trobar un successor que és acceptat a cada iteració.

Hipòtesis: Es pot trobar una combinació amb què Simulated Annealing mostri els mateixos resultats que Hill Climbing o millors. (H0)

Mètode:

Determinarem els paràmetres òptims mitjançant la realització de 10 execucions amb la seed 1234 pels sensors i 4321 pels centres de dades per cada parell de diferents valors de k i λ. Modificarem també el nombre d'iteracions i les iteracions per produir-se canvi en la temperatura.

Concretament, per fer les proves utilitzarem la funció heurística esmentada anteriorment, la generació de solucions inicials que en l'experiment 1 ens ha suposat ser la millor, i la successor function corresponent, on només s'utilitza l'operador descrit anteriorment.

Per valors establerts de k = 4 i k = 12 i una λ de 0.001, provarem un rang d'iteracions entre [1000 - 8000]. El nombre d'iteracions per cada canvi de temperatura serà n/100, on n és el #d'iteracions de cada execució.

Les gràfiques ens mostren la mediana d'aquestes execucions per cada paràmetre concret.

Es pot observar a partir de les gràfiques incloses, que per un valor de k elevat, les execucions realitzades no són suficients per arribar a un cost assequible. Per k=4, aconseguim un cost amb mitjana aparellat amb l'execució de Hill Climbing millor de l'experiment 1.

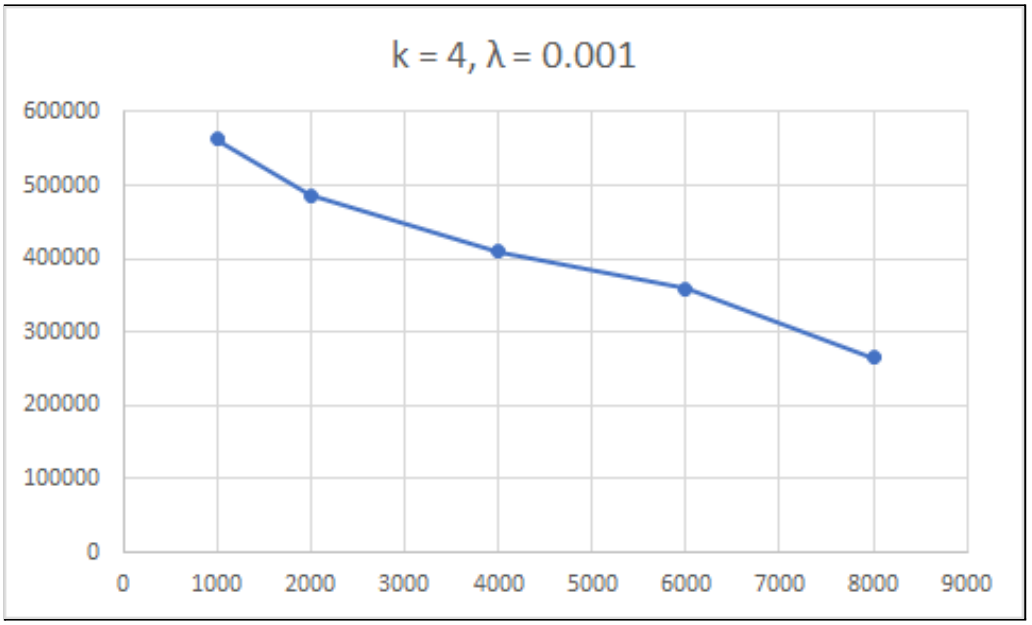

Fig. 9 - Gràfica de costs SA k=4, lambda=0.001

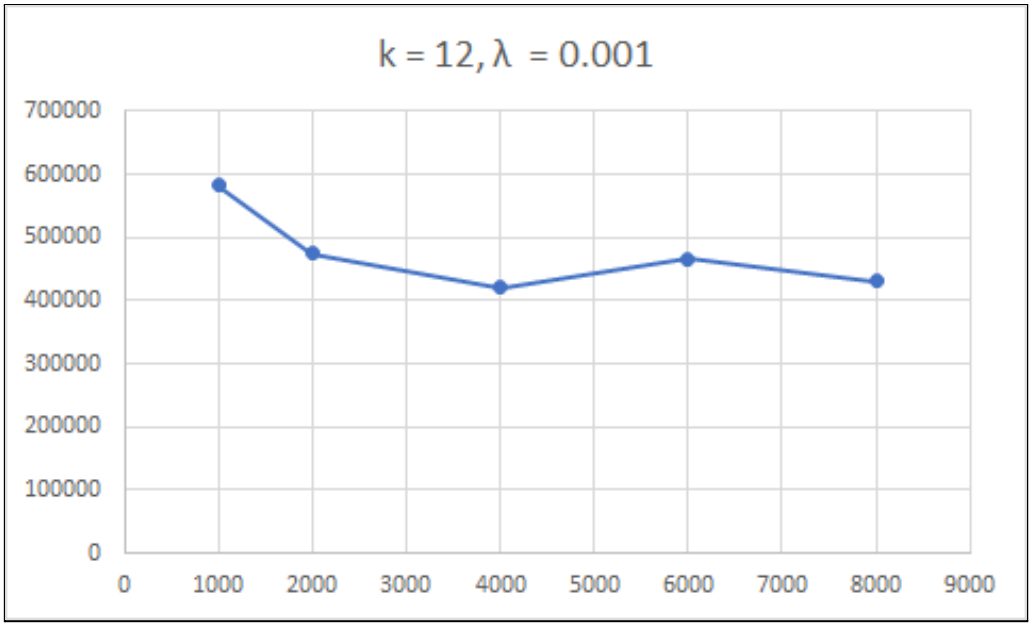

Fig. 10 - Gràfica de costs SA k=12, lambda=0.001

4. Donat l'escenari dels apartats anterior, estudieu com evoluciona el temps d'execució per trobar la solució per valors creixents de paràmetres seguint la proporció 4:100. Comenceu amb 4 centres de dades i incrementeu el nombre de 2 en 2 fins que es vegi la tendència. Feu servir l'algoritme de Hill Climbing i la mateixa funció heurística que abans.

Observació: Hi ha d'haver un nombre de centres K en què es noti la tendència en el temps d'execució en l'escenari [4-K] centres de dades i 100 sensors.

Plantejament: Execució de l'algorisme Hill Climbing per observar la tendència creixent en el temps d'execució amb l'augment de centres de dades.

Hipòtesis: Es pot trobar un nombre de centres K que constitueixi tendència del temps d'execució en un escenari de 100 sensors fixos.

Mètode: Començant amb l'escenari de 4 centres de dades i 100 sensors farem execucions successives augmentant en 2 el nombre de centres fins a trobar un nombre K en trobar una tendència raonable en el temps d'execució. Utilitzarem la seed 1234 pels sensors i 4321 pels centres de dades.

#centres	Cost	**Temps d'execució (ms)**
4 centres	257806.0	8014
6 centres	232876.0	7906
8 centres	93682.0	8593
10 centres	90961.0	8944
12 centres	80751.0	8772
14 centres	72324.0	9601
16 centres	65278.0	9770
18 centres	59462.0	10309
20 centres	48671.0	10620
22 centres	44172.0	11423
24 centres	42225.0	12184
26 centres	40467.0	12357

28 centres	40257.0	12048
30 centres	38207.0	14344
32 centres	36905.0	15158
34 centres	36770.0	15259
36 centres	31574.0	16683
38 centres	28053.0	15930
40 centres	28002.0	16217
42 centres	27921.0	16783
44 centres	25687.0	17387
46 centres	24551.0	18230
48 centres	22907.0	19049
50 centres	22587.0	19744
52 centres	22472.0	20578
54 centres	22338.0	26333
56 centres	22026.0	**27131**
58 centres	21888.0	26909
60 centres	21754.0	26988

Per K >= 56, trobem un valor en el temps d'execució en ms que fa tendència i és 27000ms. Les següents execucions augmentant K paren d'augmentar el seu temps d'execució quedant-se en el valor corresponent.

A més, en aquestes execucions on el #centres de dades supera o és igual a 56, el cost aproximat tampoc no disminueix.

Experiment 5

5. Les proporcions del primer escenari fan que la capacitat de rebre dades sigui molt més gran que la captura, A partir de tots els experiments realitzats, hi ha resultats en els que no tots els centres de dades són utilitzats?

Sí. Aquesta casuística es donarà en aquelles generacions de Sensors i Centres de dades on un dels centres està situat geogràficament a llocs molt poc accessibles (amb una distància relativament gran) per a qualsevol altre sensor i, a més a més, hi hagi un excedent de "centres".

Respectant les restriccions de la Xarxa, aquest sensor mai es connectarà a un centre llunyà, sempre que tingui una altra connexió vàlida amb un altre centre o se

Experiment 6

6. Suposant que els centres de dades no són costosos, podríem estimar com afecta poder afegir més centres de dades a la xarxa. Fixant el nombre de sensors a 100, realitzeu experiments augmentant el nombre de centres de dades de dos en dos fins a 10 i mesureu el cost de la xarxa de connexió, el nombre de centres de dades utilitzats i el cost temporal per trobar la solució. (Hill Climbing i Simulated Annealing)

Observació: Afegir centres de dades, hipotèticament no-costosos, disminueixen el cost de la xarxa de connexió.

Plantejament: Mitjançant Hill Climbing i Simulated Annealing mesurarem la implicació de l'increment de centres de dades en la nostra xarxa.

Hipòtesis: Els 3 generadors de solucions són iguals(H0) o un d'ells produeix millors resultats.

Mètode:

- Tenim un conjunt de 10 seeds aleatòries.
- Executem un experiment per cada seed i generador de solucions inicials.
- El nombre de sensors serà 100 per totes les execucions i 4 centres de dades.
- Farem servir l'algoritme de Hill Climbing
- Estudiarem el valor final de la funció heurística i el nombre de passos que ha fet H.C. per arribar a la solució corresponent.
- Paràmetres SA: k=4 lambda=0.001 iteracions=8000 passos per iteració=80

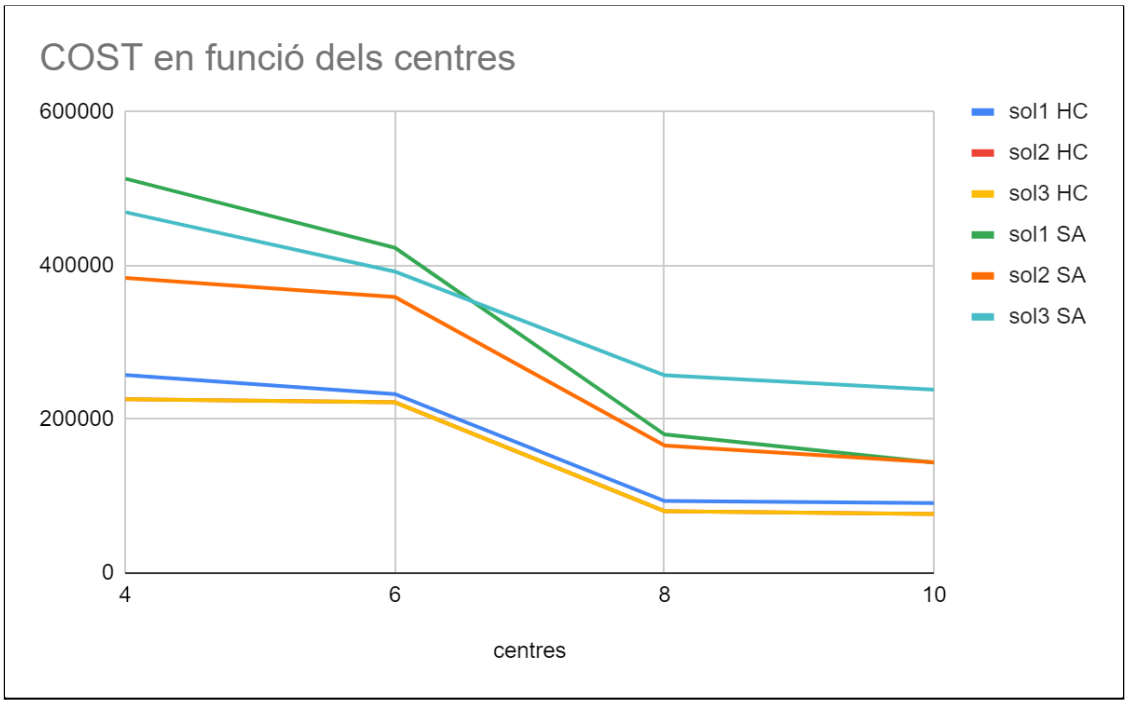

Fig. 11 - Gràfica de costs comparativa de HC i SA amb nCentres variant

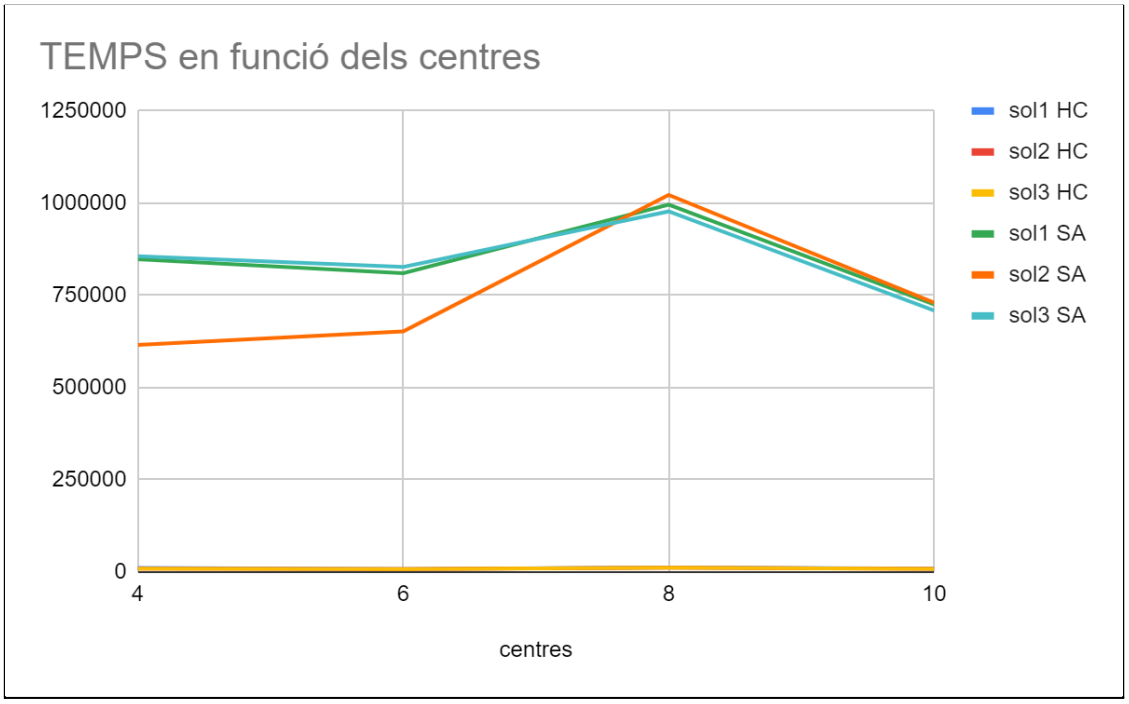

Fig. 12 - Gràfica de temps comparativa de HC i SA amb nCentres variant

La disminució del nombre de centres de dades suposa, com esperàvem, una disminució en el cost total de les connexions de la xarxa.

Les gràfiques de la Fig.11 i Fig.12 ens proporcionen informació sobre la millor manera d'afrontar el problema en qüestió. Aquesta és mitjançant Hill Climbing amb la generació inicial que suposa un ordre en les connexions (2 i 3). Concretament, aquestes van a la par en la gràfica de costos en funció dels centres, atribuint la similitud entre elles.

Tot i que Simulated Annealing ens suposa unes solucions plausibles al nostre problema, el seu cost amb els paràmetres amb els quals hem experimentat, segueix sent elevat. Seria necessària una recerca més incisiva sobre els diferents paràmetres òptims sobre els quals l'algorisme hauria de córrer en aquesta instància del problema concreta.

Igualment, a causa del seu cost temporal elevat, encara que s'aproximés a les dades de Hill Climbing, el refutaríem. Només si suposes una excel·lència en comparació a les execucions fetes per Hill Climbing, donaríem Simulated Annealing com l'algorisme indicat per córrer el nostre problema de cerca local.

Experiment 7

7. En l'escenari que heu explorat està pràcticament assegurat transmetre totes les dades, el que fa que el factor de la funció heurística que maximitza les dades transmeses no tingui quasi efecte durant la cerca (la majoria del temps és constant) i que només es tingui en compte el cost de la xarxa de distribució. Ara canviarem l'escenari de manera que hi hagi 2 centres de dades i 100 sensors.

> **-S'ha reduït el temps a trobar una solució?**
> **-Si fem servir una ponderació que mesura la quantitat d'informació enviada igual que als primers escenaris, proporciona un augment en el cost de la xarxa?**
> **-Com canvia el cost de la xarxa en funció de la ponderació que es dóna en enviar les dades? Hi ha alguna ponderació a partir de la que el cost de la xarxa ja no augmenta?**

Segons el nostre criteri d'espai de solucions, mai deixem que els operadors ni la generació de solucions inicials ens porti fora de transmetre la quantitat màxima d'informació sempre.

Per tant, cap ponderació és necessària en el nostre cas per poder representar el criteri de qualitat que maximitza la informació transmesa. Sent el volum de transmissió constant, està present dins la funció heurística, però és totalment constant i sense ponderacions. Això ens permet minimitzar el cost de les diferents connexions de la xarxa, sense preocupar-nos de cap ponderació.