VRP enriquit aplicat al sector del transport d'hidrocarburs

Gabriel Alemany Giménez
Enginyeria d'Informàtica (2on Cicle)

Consultora**: Jésica de Armas Adrián**

22 de gener de 2015

Aquesta obra està subjecta a una llicència de
Reconeixement-NoComercial-SenseObraDerivada
3.0 Espanya de Creative Commons

FITXA DEL TREBALL FINAL DE CARRERA

Títol del treball:	VRP enriquit aplicat al sector del transport d'hidrocarburs
Nom de l'autor:	Gabriel Alemany Giménez
Nom del consultor:	Jésica de Armas Adrián
Data de lliurament:	01/2016
Àrea del Treball Final:	Applications of Simulation in Industry & Services
Titulació:	Enginyeria d'Informàtica (2on cicle)

Resum del treball:

En aquest treball combinem una sèrie de tècniques per a resoldre un problema de rutes de vehicles amb capacitats limitades (CVRP). El plantejament és el més obert possible, acceptant un nombre variable de productes, de bases logístiques i de tipus de vehicle per a cada base (flota heterogènia, que caracteritza cada tipus de vehicle pel nombre de compartiments i la capacitat de cadascun d'ells). El disseny també contempla el fet de que un mateix vehicle realitzi més d'una ruta, sempre que el servei encaixi en un interval de temps determinat, i que, en cas de manca de producte a la base, la ruta es pugui iniciar en un magatzem.

Tot i la complexitat del problema, veurem que només ens cal un reduït nombre de paràmetres d'ajust de l'algorisme. L'estratègia bàsica és utilitzar una heurística eficient (com pot ser CWS). Atès que aquesta no garanteix la millor solució, el procés es repeteix tantes vegades com sigui possible, en un interval prefixat de temps, triant un paràmetre com a aleatori esbiaixat. És a dir, que varia aleatòriament però afavoreix els valors suggerits per l'heurística. Aquesta tècnica veurem que és aplicada en vàries de les etapes de resolució del problema. Tècniques de cerca local i de memòria cau són aplicades per a la millora de les solucions.

Abstract:

In this work a number of techniques to solve the Capacitated Vehicle Routing Problem (CVRP) are combined. The approach is as open as possible, accepting a variable number of products, depots and vehicle types for each depot (heterogeneous fleet, which characterizes each vehicle type by its number of compartments and the capacity of each one of them). The design also includes the possibility for a vehicle to operate more than one route, provided the service fits in a given time and, in case of stock out, the route can be started from a warehouse.

Despite the complexity of the problem, we'll see that only a small number of

parameters are needed for algorithm tuning. The basic strategy is to use an efficient heuristic (such as CWS). Since this does not guarantee the best solution, the process is repeated as many times as possible in a fixed time interval, choosing a parameter to be biased randomized. That is, it varies randomly but promotes the values suggested by heuristics. We'll see that this technique is applied in several stages of problem resolution.
To improve solutions, local search and caching techniques are applied.

Paraules clau

CVRP, Heurística, Multiarrencada, Optimització, Ruta, Vehicle Compartimentat, Transport d'Hidrocarburs, Aleatorització Esbiaixada.

Contingut

1. Introducció 1
   1.1 Context i justificació del treball 1
   1.2 Objectius del treball 1
   1.3 Enfocament del mètode seguit 2
   1.4 Planificació del treball 3
   1.5 Productes obtinguts 3
   1.6 Descripció dels altres capítols de la memòria 3
2. Esbós històric del problema 3
3. Descripció 4
4. Tècniques de resolució emprades 6
   4.1 Mètodes heurístics, mètodes aproximats i mètodes exactes 6
   4.2 El mètode Multiarrencada i la tècnica de l'aleatorització esbiaixada 6
   4.3 L'heurística Clarke & Wright Savings aplicada al CVRP 8
   4.4 Una heurística basada en estalvis aplicada a l'assignació de nodes a bases 10
   4.5 El mètode d'aproximacions successives per a solucions amb flotes heterogènies 11
   4.6 Les tècniques de memòria cau 12
   4.7 Algorisme 2-opt de millora de rutes 13
5. Descripció detallada de l'algorisme 15
   5.1 El conjunt de dades d'entrada 15
      5.1.1 Paràmetres de l'algorisme 15
      5.1.2 Instància 16
   5.2 L'estructura de dades bàsica 17
   5.3 L'algorisme a un primer nivell 20
   5.4 La generació de submapes 21
   5.5 La solució del submapa 23
      5.5.1 La reducció al problema de flota homogènia 23
      5.5.2 La generació de subclients 25
      5.5.3 Les rutes de provisió 25
      5.5.4 El multitrajecte 27
      5.5.5 La classificació de les rutes 27
      5.5.6 L'aplicació del mètode CWS 29
      5.5.7 El càlcul del temps total de la ruta 31
      5.5.8 La distribució de càrrega dels vehicles 31

5.5.9 La disponibilitat de producte a la base 32
5.6 L'optimització de les rutes 35
6. Proves 36
6.1 Cas teòric simplificat 36
6.2 Cas real 37
6.2.1 Solució implementada per l'empresa del sector 38
6.2.2 Solució calculada per l'algorisme 41
6.3 Instància de dimensió més alta 45
7. Conclusions i proposta de treballs futurs 48
Agraïments 50
Glossari 51
Bibliografia 51
Annexos 54
Annex 1. Solució cas real 54

1. Introducció

1.1 Context i justificació del treball

Aquest treball es centra en un problema logístic que apareix habitualment en empreses del sector de transport per carretera, particularment en els casos en que la comanda del client es composa de diversos productes que no es poden barrejar entre ells. Com a exemples podem citar: productes derivats del petroli, industria alimentària i, en un sentit invers però equivalent, la recollida selectiva de deixalles. Es tracta d'un problema de distribució en el qual l'objectiu principal és minimitzar els costs.

En un entorn econòmic en que el consumidor exigeix continuadament millores en la relació preu/qualitat, un decrement del cost de distribució pot ser un avantatge competitiu important. En el cas dels productes que poden ser emmagatzemats amb certa facilitat però de transport costós, la tècnica emprada sol ser situar estratègicament diferents bases logístiques (d'aquí endavant, bases), cadascuna amb una flota de vehicles més o menys important. Aquest fet li dóna al problema la seva característica multibase.

Quan els productes són de transport incompatible, la solució òbvia d'utilitzar un vehicle per a cada producte normalment no és la més rendible perquè requereix d'un desplegament més gran que l'estrictament necessari. Aleshores, la pràctica habitual és la utilització de vehicles multicompartimentats, tot i que això genera una logística més complexa. Aquesta complexitat s'agreuja si pensem en que només en casos molt excepcionals és possible mantenir una flota homogènia degut a la constant evolució de la industria de l'automoció del transport i també a les necessitats de flexibilització dels repartiments, fent-se necessària la introducció del concepte de flota heterogènia.

Com a conseqüència d'aquests factors, habitualment trobarem empreses que tenen assignats diferents tipus de vehicles a les seves bases per tal de subministrar els seus clients. La complicació de la definició del problema impossibilita rendibilitzar els mitjans dels que es disposa amb planificacions de rutes fetes de forma manual o semi-automàtica.

La mateixa dinàmica del sector demana també una reducció dels costos d'emmagatzematge a banda dels de transport, generant-se, per part del gestor, un cert grau de desplaçament de disponibilitat dels articles des de les bases als magatzems dels proveïdors. Tot i que, en el present treball, no tenim en compte els costos d'emmagatzematge, sí que permetem que aquests magatzems siguin inclosos en les rutes dels vehicles, permetent-se d'aquesta manera una demanda total superior a la disponibilitat de la base.

1.2 Objectius del treball

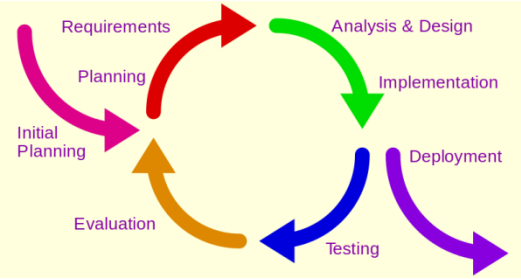
L'objectiu principal del treball es aconseguir un algorisme (i la seva implantació total en llenguatge Java), que trobi solucions properes a les òptimes tenint en consideració que es desitja minimitzar els costos de distribució, en operacions amb característiques definides que més endavant especificarem (veure punt 3). Quantitativament, s'espera que per a una operació real que havia estat planificada a mà l'algorisme proposi una solució amb un cost total, com a mínim, un 20% menor.

Com a objectius addicionals, podem mencionar:

- Suficientment flexibilitat com per que pugui ser utilitzat per una majoria de les empreses del sector.
- Disseny prou obert com per que es pugui ampliar la funcionalitat o la complexitat amb un esforç raonable.
- Documentació prou exhaustiva com per que la utilització i el manteniment siguin àgils.
- Temps d'execució raonable.

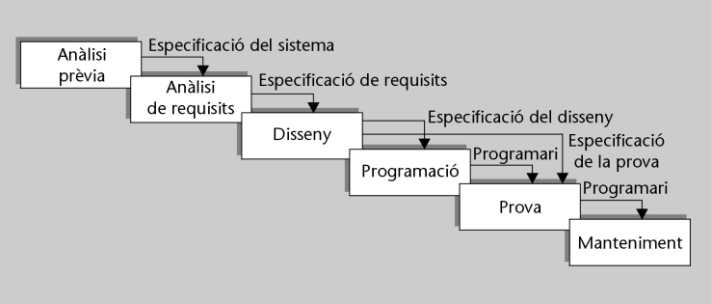
1.3 Enfocament del mètode seguit

Aplicarem un cicle de vida iteratiu i incremental (Cockburn, 2008).



Gràfic 1. El cicle de vida iteratiu i incremental

Descompondrem el projecte en parts i, a cadascuna d'aquestes, li aplicarem un cicle de vida clàssic.



Gràfic 2. El cicle de vida clàssic o en cascada

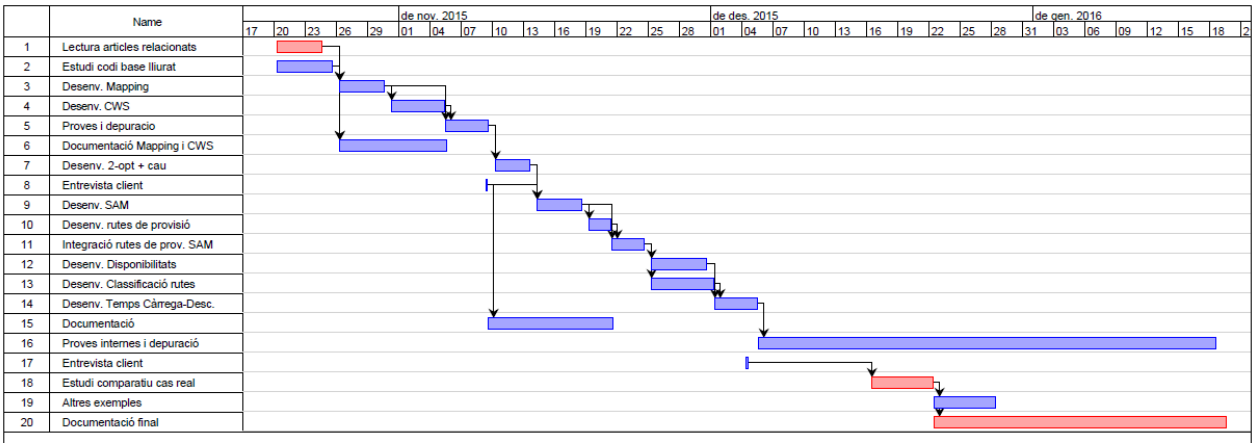
Durant el cicle de vida de cada part, tindrem la possibilitat de millorar o corregir parts que s'han desenvolupat anteriorment.

En aquest cas, les parts en que es divideix el projecte són fàcilment identificables. A continuació es llisten en l'ordre que ens ha semblat més adient:

1) Entrada de dades de l'algorisme i de la instància
2) Generació del mapatge
3) Procediment de generació de rutes
4) Millora de la solució
5) Presentació de la solució

1.4 Planificació del treball

Aquest treball ha estat planificat segons el següent diagrama de Gantt:



Gràfic 3. Diagrama de Gantt de la planificació del projecte

1.5 Productes obtinguts

D'aquest treball s'ha obtingut, com a objecte principal, un programa en Java capaç de resoldre el problema descrit en l'apartat 3, tot assolint els objectius mencionats a 1.2.

Addicionalment s'inclou un joc de proves que demostren el bon funcionament del programa i una descripció detallada, en la present memòria, de l'algorisme en el que està basat.

1.6 Descripció dels altres capítols de la memòria

En l'apartat 2, tot seguit, veurem un esbós històric del problema a on s'introduiran treballs anteriors d'altres autors en els quals ens hem basat per desenvolupar el projecte.

En l'apartat 3 es descriu l'enunciat del problema per, posteriorment en l'apartat 4, introduir les tècniques de programació i algorismes emprats per resoldre'l.

Més endavant, en l'apartat 5, veurem amb certa profunditat els desenvolupaments addicionals que ha calgut fer i les estructures de dades emprades, així com els enriquiments i adaptacions de les tècniques esmentades en l'apartat 4.

Finalment, en l'apartat 6 posarem a prova el producte obtingut amb casos teòrics i reals.

2. Esbós històric del problema

El Problema d'Encaminament de Vehicles (VRP) es planteja com la recerca d'un conjunt òptim de rutes per a una flota de vehicles de forma que puguin abastir un conjunt determinat de clients (Dantzig & Ramser, 1959). És una generalització del conegut Problema del Viatjant de Comerç (TSP), que tracta de trobar l'ordre de visites a una sèrie donada de clients per tal de minimitzar la longitud total de la ruta, i va estar formulat matemàticament per primera vegada a principis del segle XIX per W.R. Hamilton i T. Kirkman (Biggs, Lloyd, & Wilson, 1986).

Una ampliació natural del VRP és el Problema d'Encaminament de Vehicles Capacitats (CVRP) (Toth & Vigo, 2001), en el que es subministra des d'una base una quantitat determinada d'un

únic producte a cada client amb una flota homogènia de vehicles que tenen una capacitat limitada de transport.

El problema tractat en aquest treball és una conjunció dels problemes Multi-Depot Vehicle Routing Problem (MDVRP), Multi-Capacitated Vehicle Routing Problem (MCVRP) i Heterogeneous Fleet Capacitated Vehicle Routing Problem (HeCVRP) amb algunes altres ampliacions que descriurem més endavant.

Referent al MDVRP, disposem de contribucions de més de 40 anys. Per exemple: Tillman & Cain (1972), basat en CWS, és la primera variant del VRP amb més d'una base. L'algorisme combina rutes, com farem en el que estem presentant, maximitzant estalvis; però sense prèviament dividir-los entre les bases. Més recentment Juan, Pascual, Guimarans, & Barrios (2014) han encarat el problema fent ús de la aleatorització esbiaixada (com en aquest treball) i de la Cerca Local Iterada (ILS).

En el grup MCVRP, trobem literatura relacionada amb la distribució de productes derivats del petroli (Avella, Boccia, & Sforza, 2004) que formula el problema multiproducte, amb una sola base i una flota heterogènia de camions multicompartimentats, utilitzant la tècnica Branch-and-Price, més pròxima a la programació lineal. També pertany al mateix àmbit un altre article (Cornillier, Boctor, Laporte, & Renaud, 2008) en el que s'aborda el problema maximitzant la rendibilitat de les operacions logístiques fins a un horitzó de temps planificat. Per altra banda, també trobem estudis similars relacionats amb recol·lecció selectiva d'escombraries, en els que torna a ser imprescindible la separació dels productes en diferents compartiments (Henke, Grazia Speranza, & Wäscher, 2014) per aplicar-ho a la recollida de residus del vidre (que cal separar-los en diferents colors) per camions amb compartiments flexibles.

Pel que fa al HeCVRP destaca una primera aproximació (Taillard, 1999) utilitzant Generació Heurística de Columnes (HCG) i una proposta (Li, Golden, & Wasil, 2007) basada en una variant del algoritme "record-to-record travel" pel CVRP que te en compte la flota heterogènia (HRTR).

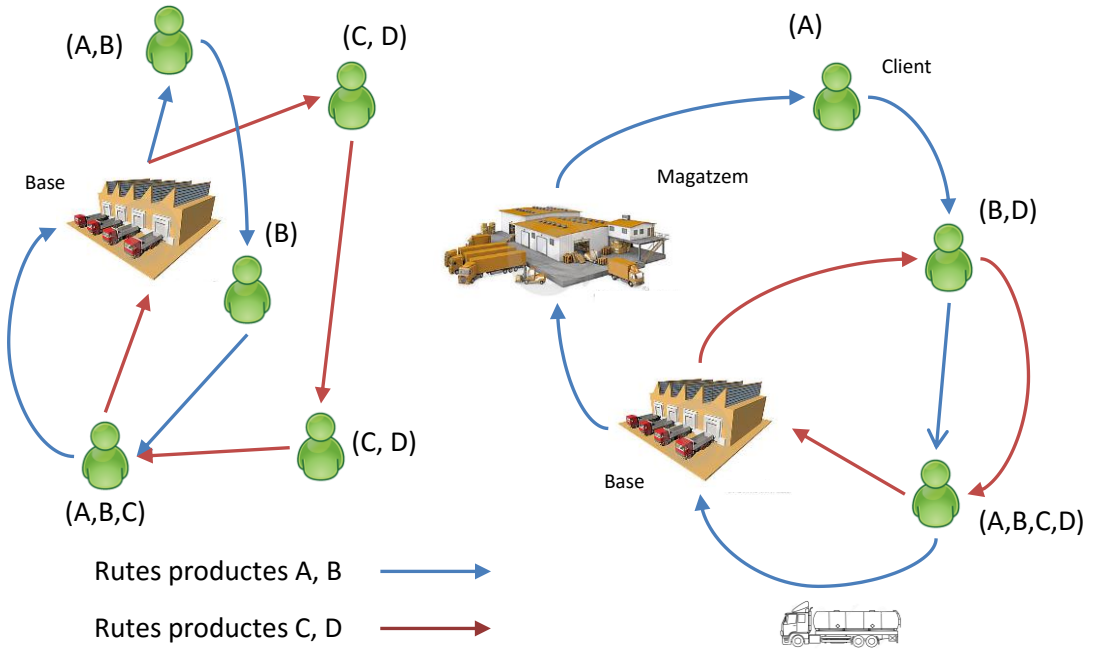
Cal també mencionar una recent solució del problema HeCVRP + MCVRP (Wang, Ji, & Chiu, 2014), batejada pels autors com amb l'acrònim HFFMCVRP (Heterogeneous and Fixed Fleet, Multi-capacitated VRP). En ella utilitzen una metaheurística anomenada Reactive Guided Taboo Search (RGTS).

3. Descripció

El problema que plantegem és el de minimitzar el cost de transport d'una empresa logística amb les següents característiques:

- L'empresa disposa d'un nombre indeterminat de clients que fan comandes de productes.
- El negoci manega un nombre parametritzable de productes.
- L'empresa disposa d'un nombre indeterminat de bases de distribució, amb unes existències inicials de productes per subministrar els clients.
- Es pot imposar un estoc mínim de seguretat de cada producte a cada base.
- Cada client pot ser subministrat per a una sola base.

- Es defineix una sèrie de tipus de vehicles, diferenciats entre si pel que fa al cost d'operació (tant de preparació com per km), al nombre de compartiments de càrrega i a les capacitats de cada compartiment.
- Cada base disposa d'una flota fixa parametritzable de vehicles de diferents tipus per subministrar els clients.
- Els vehicles, per tal de fer el repartiment dels productes als clients, defineixen rutes que comencen i acaben a la base que aquests tenen assignada.
- Cada vehicle pot fer més d'una ruta en una mateixa jornada.
- La jornada té una durada màxima definida.
- Cada client només accepta un subconjunt (no buit) de tipus de vehicle
- Un subconjunt de tipus de vehicle, en cas de mancança d'existències a la base, té la possibilitat de fer el primer tram de la ruta a un dels magatzems reguladors que tenen disponibilitat il·limitada de cada producte.
- El vehicle que faci el primer tram a un magatzem haurà de sortir buit de la base.
- Per a optimitzar la visita al magatzem, el vehicle sempre carregarà la totalitat de la seva capacitat.
- L'excedent del repartiment dels vehicles que passen pel magatzem es descarrega a la base, quedant disponible per a altres vehicles que iniciïn rutes a partir d'aquest moment.
- Es disposa de les distàncies i dels temps de trànsit entre cada parella de nodes (bases, clients i magatzems).
- Cal tenir en compte els temps de càrrega i descàrrega de productes (funció d'un cabal, comú a tot el problema) y el temps de preparació de la ruta (també comú per a tot el problema).
- Als trajectes dins d'una mateixa població se'ls aplica una penalització de temps.



Gràfic 4. Exemple gràfic

4. Tècniques de resolució emprades

Aquest problema és una extensió del CVRP, que és NP-difícil (Prins, 2004). Per nombres de client relativament baixos es pot resoldre utilitzant mètodes de programació lineal. Quan es considera que la dimensió pot ser de centenars de clients i/o el nombre de variables en joc relativament elevada, com és el cas, no és permissible abordar-lo des d'aquesta perspectiva per la quantitat de temps de càlcul que seria necessari.

4.1 Mètodes heurístics, mètodes aproximats i mètodes exactes

Els mètodes d'optimització es poden dividir en tres categories: heurístiques, algorismes d'aproximació i mètodes exactes.

Les heurístiques són mètodes que generalment obtenen solucions ràpidament i amb qualitat raonable però no proporcionen cap garantia de que així sigui i, molt menys, de que la solució sigui l'òptima.

Molts algoritmes heurístics són molt específiques i dependents del problema. D'altra banda, una metaheurística és un marc de referència independent del problema que aporta una sèrie de pautes o estratègies per dissenyar heurístiques subjacents que, a la seva vegada, poden resoldre problemes específics d'optimització (Glover, 1986).

Els algorismes d'aproximació són uns tipus especials d'heurístiques que aporten, junt amb la solució, una cota de l'error. Per exemple, un mètode pot garantir que la solució obtinguda és, com a molt, k vegades més costosa que la solució òptima.

Els mètodes exactes estan dissenyats de manera que es garanteix trobar la solució òptima en una quantitat finita de temps i d'espai. No obstant això, per problemes d'optimització NP-difícils, com seria el CVRP, aquesta "quantitat finita de temps" pot augmentar de forma exponencial respecte a les dimensions del problema. No obstant, per a certes classes de problemes, existeix la possibilitat de trobar algorismes que resolguin certes instàncies en un temps raonable.

4.2 El mètode Multiarrencada i la tècnica de l'aleatorització esbiaixada

Donada la quantitat de requeriments i regles de negoci a tractar en el problema, és evident que caldrà fer ús d'heurístiques per calcular solucions. Però hem vist que les heurístiques, per sí mateixes, només aporten una solució i, a més , sense cap garantia de qualitat.

El mètode Multiarrencada (György & Kocsis, 2011) permet millorar la qualitat de les solucions provinents d'heurístiques. El seu ús implica la introducció de components aleatoris en alguns dels passos de l'algorisme en els que es realitza una selecció. A partir d'aquí, es repeteix el procés una gran quantitat de vegades (normalment durant un temps predeterminat). Com que cada execució produirà una solució diferent, podrem comparar-les, pel que fa al cost de cadascuna d'elles, i seleccionar la millor.

L'algorisme Multiarrencada, en la seva forma més genèrica, seria:

```
1.  algorisme Multiarrencada
2.      entrada: tempsDexecució
3.      sortida: millorSolució
4.  millorSolució ← Ø
```

```
5.      Mentre no fi de tempsDexecució fer
6.          novaSolucio ← càlculHeurísticSolució()
7.          Si factible(novaSolució) Llavors
8.              Si millorSolucio = Ø o
9.                  cost(novaSolució) < cost(millorSolució) Llavors
10.                 millorSolució ← novaSolució
11.             Fi Si
12.         Fi Si
13.     Fi Mentre
14.     Retornar millorSolució
```

Pseudocodi 1. Multiarrencada

Multiarrencada necessita variabilitat en les solucions per tal de poder triar la millor. Aleshores caldrà afegir un o més graus de llibertat a l'hora en que l'algorisme hagi de prendre decisions. Com a norma general, la solució d'aquests tipus de problemes es situen a les fulles d'un arbre que l'algorisme ha de recórrer. L'heurística evita el recorregut total de l'arbre indicant en cada moment el camí a seguir (tries heurístiques), però aplicant-la estrictament s'arribaria sempre a una mateixa solució que normalment seria de baixa qualitat.

Per tal d'aportar variabilitat a les tries heurístiques, s'aplica una aleatorització esbiaixada (Juan A. , et al., 2009) (Juan, Pascual, Guimarans, & Barrios, 2014): en comptes de directament assignar l'opció heurísticament més avantatjosa en cada pas del procés, s'assigna "una de les més" avantatjoses. Les probabilitats de triar cadascuna de les opcions venen donades per una distribució de probabilitat geomètrica de raó β, essent β és el coeficient de biaix de la aleatorització. És a dir, per generar $i \in [0, n-1]$ enter a partir del nombre pseudoaleatori $rnd \in (0,1)$ real:

$$i = rndB(n, \beta) = Ent\left(\frac{log(rnd)}{log(1-\beta)}\right)(mod\ n)$$

Observem que:

$\beta \longrightarrow 1 \Rightarrow i \longrightarrow 0$ (sempre es tria la millor opció, sense variabilitat)

$\beta \longrightarrow 0 \Rightarrow \frac{\log(rnd)}{\log(1-\beta)} \longrightarrow \infty$ i, en prendre $(mod\ n)$, i es distribuirà uniformement en $[0, n-1]$. És a dir, no hi haurà influència de l'heurística.

a partir d'aquí, podem generar una llista d'índexs esbiaixats amb l'algorisme:

```
1.      Càlcul Array d'índexs esbiaixats
2.          entrada: paràmetre de biaix β, nombre d'elements n
3.          sortida: array d'índexs
4.      Siguin posArray, auxArray arrays amb n elements
5.      Sigui auxArray[i] = i per a i en [0, n-1]
6.      Per a cada posició de posArray i fer
7.          biasedPos ← rndB(n-i, β)
8.          posArray[i] ← auxArray(biasedPos)
9.          Per a posicions d'auxArray j∈[biasedPos, n-i-2] fer
```

```
10.                 auxArray[j] = auxArray[j+1]
11.             Fi Per
12.         Fi Per
13.         Retornar posArray
```

Pseudocodi 2. El càlcul del vector d'índexs esbiaixats

Exemple de seqüencia de 14 posicions amb $\beta = 0{,}5$:

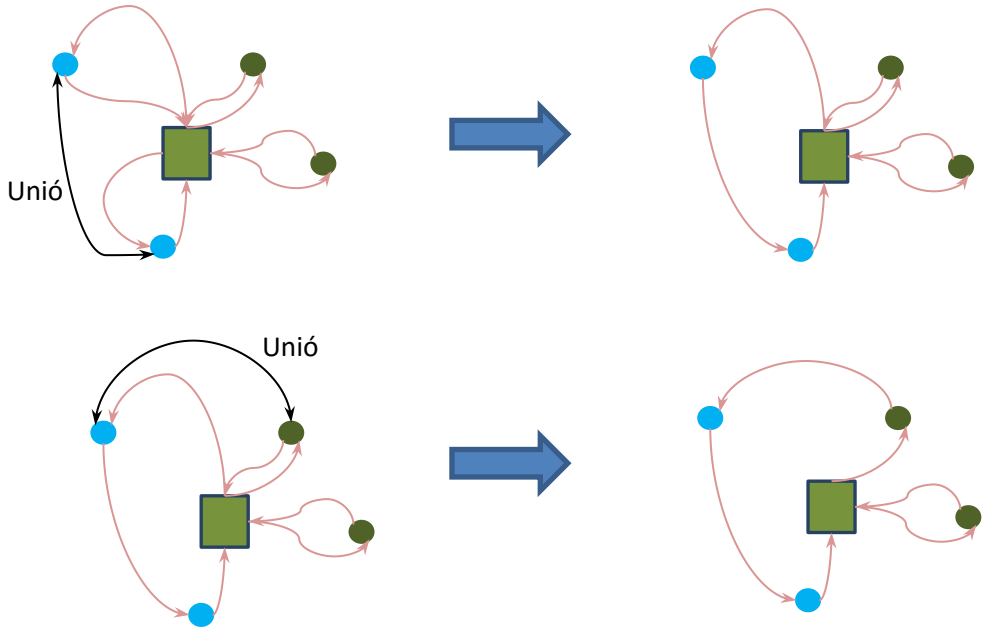
0 1 0 1 2 2 0 0 3 0 1 0 0 1

Com veurem més endavant, aquesta tècnica s'aplica en diferents punts del projecte.

4.3 L'heurística Clarke & Wright Savings aplicada al CVRP

L'heurística d'estalvis Clarke & Wright (CWS) (Clarke & Wright, 1964) s'aplica al problema CVRP amb una sola base, un sol producte i flota homogènia, en principi, il·limitada.

Inicialment es generen rutes individualitzades per a cada client (rutes *dummy*) per tot seguit procedir a la unió de parelles de rutes, sempre que es compleixin les condicions necessàries. La més obvia: que el vehicle sigui capaç de transportar el total de la demanda d'ambdues rutes, ja sigui per la capacitat, per la quantitat de productes diferents o per ambdues raons. No obstant, veurem que en el problema que plantegem caldrà imposar altres condicions.



Gràfic 5. Visualització gràfica del CWS

L'heurística CWS imposa que cal fer primer les unions amb les que s'aconsegueixi un estalvi més alt, sent l'estalvi de la unió dels node i i j:

$$estalvi_{ij} = c_{ij} - c_{0j} - c_{0i}$$

On c_{ij} és el cost del trajecte $i - j$ i 0 és el node corresponent a la base.

Per prioritzar les unions amb més estalvi, caldrà disposar d'una llista, ordenada segons estalvis, de totes les arestes que sigui possible formar dels nodes clients (i magatzems) entre sí. A continuació podrem iniciar el procés d'unió de rutes, eliminant de la llista l'aresta que és utilitzada (o que s'ha intentat utilitzar).

Tot i que l'heurística CWS diu que sempre cal triar el màxim estalvi, això només produiria una solució. Per aconseguir la variabilitat necessària ens cal aplicar l'aleatorització esbiaixada en aquest punt, de manera que la probabilitat de triar la primera aresta de la llista (posició 0) vindrà determinada pel paràmetre β.

Pseudocodi de l'algorisme CWS, en la versió d'aleatorització esbiaixada, per trobar una solució al problema CVRP de flota homogènia:

```
1.  Biased-randomized CWS
2.      Entrada: β, nodes
3.      Sortida: solució
4.  Obtenir la base
5.  solució ← generar solució dummy
6.  Generar llista d'arestes estalvis
7.  Ordenar estalvis descendent
8.  Mentre estalvis != Ø fer
9.      Selecciona aleatòriament esbiaixat( β ) índex
10.     e_ij ← estalvis_index
11.     Eliminar e_ij d'estalvis
12.     iNode ← node origen(e_ij)
13.     jNode ← node final(e_ij)
14.     R_i ← ruta que conté iNode
15.     R_j ← ruta que conté jNode
16.     iE ← aresta que uneix iNode amb base
17.     jE ← aresta que uneix jNode amb base
18.     Verificar si unió R_i, R_j usant e_ij és possible
19.     Si unió possible llavors
20.         Eliminar aresta iE de ruta iR
21.         Si nombre d'arestes de R_i, > 1 llavors iNode.ésInterior ← cert  Fi si
22.         Si origen de primera aresta de R_i no és la base llavors invertir R_i Fi si
23.         Eliminar aresta jE de la ruta R_j
24.         Si nombre d'arestes de R_j > 1 llavors jNode.ésInterior ← cert Fi si
25.         Si origen de primera aresta de R_j és la base llavors invertir R_j Fi si
26.         Afegir aresta e_ij a R_i
27.         Ruta(jNode) ← R_i
28.         Afegir arestes(R_j) a R_i
29.         Eliminar R_j de solució
30.         dummy(R_i) ← fals
31.     Fi Si
32. Fi Mentre
33. Retornar solució
```

Pseudocodi 3. Algorisme CWS amb selecció aleatòriament esbiaixada d'arestes d'estalvis

Mencionem que en Pseudocodi 3, línia 27 s'actualitza l'atribut del node que indica a quina ruta està assignat. Aquest valor s'utilitza a les línies 14 i 15. En la línia 30 s'estableix que la ruta deixa de ser *dummy*, atribut normalment útil per validar la unió de rutes.

4.4 Una heurística basada en estalvis aplicada a l'assignació de nodes a bases

Per a aplicar l'algorisme anteriorment detallat cal trobar abans una partició del conjunt de nodes per a assignar cadascun d'ells a una base. El procediment haurà de ser el màxim d'eficient possible, atès que influirà decisivament en el cost de la solució final.

El mètode emprat està basat en estalvis, igual que el CWS: per a cada base b es calcula un vector d'estalvis, cada component del qual es defineix com la diferència de costos entre el trajecte del node n a la més propera de la resta de les bases i el trajecte del node n a la base b:

$$e_n^b = \min_{i \neq b}(c_{ni}) - c_{nb}$$

En un segon pas cal definir una mesura de la capacitat de servei de les bases. Aquesta magnitud associada a la base haurà de decréixer cada vegada que un node li és assignat. Així doncs, atès que cada base disposa d'una llista dels nodes ordenada per preferència, l'ordre de tria vindrà donat pel valor d'aquesta magnitud en cada base i en cada torn.

Aquest procés divideix el mapa global de nodes (clients) en un conjunt de submapes (un per a cada base) que es resoldran individualment.

Com que estem parlant d'un mètode heurístic, una altra vegada ens trobem amb el fet de que, molt probablement, el mapatge no serà l'òptim. Conseqüentment caldrà afegir també en aquest punt una aleatorització esbiaixada que provoqui que la base no sempre triï el primer node en la llista de preferències.

```
1.  Generació Mapatge
2.      Entrada: nodes, bases
3.      Sortida: mapa (o null, si no és possible generar mapa)
4.  Per a cada base fer
5.      Generar llista de nodes ordenada segons estalvis
6.      Crear submapa afegint-hi la base
7.      Agefir submapa a mapa
8.      Afegir submapa a llistaSubmapes
9.      Generar vector de posicions aleatòriament esbiaixat
        (com s'ha explicat a Pseudocodi 2)
10. Fi Per
11. nNodesAssignats ← 0
12. Mentre nNodesAssignats < nNodes & llistaSubmapes ≠ Ø fer
13.     Ordenar llistaSubmapes (segons la magnitud adient)
14.     sm ← submapes(0)
15.     assignarMillorNode(sm)
        (afectat per aleatorització esbiaixada)
16.     Si era l'últim llavors eliminar sm de llistaSubmapes Fi si
17. Fi Mentre
18. Si nNodesAssignats < nNodes llavors
19.     Retornar null
20. altrament
```

```
21.         Retornar mapa
22.     Fi Si
```

Pseudocodi 4. Algorisme d'assignació de nodes basat en estalvis

4.5 El mètode d'aproximacions successives per a solucions amb flotes heterogènies

Fins el moment tenim eines per resoldre amb garanties el problema amb flota homogènia, però el plantejament que estem abordant especifica una flota heterogènia. Això significa que cada base disposa de diferents tipus de vehicle, tant pel que fa al nombre de compartiments com a la capacitat de cadascun d'ells.

El mètode d'aproximacions successives (SAM) (Juan Á. A., Faulin, Cáceres-Cruz, Martínez, & Barrios, 2014) és una heurística per reduir el problema de una flota heterogènia amb n tipus de vehicle a n instàncies de flota homogènia (una per cada tipus de vehicle).

En un principi es resol una instància CWS amb tots els clients utilitzant un dels tipus de vehicle amb quantitat il·limitada de vehicles. De les rutes resultants, se'n trien tantes com vehicles es disposi. Els clients no subministrats passaran a una segona iteració amb un altre tipus de vehicle i, així, successivament fins que s'acabin els clients o els vehicles. Si no s'acaben els clients, no hi ha solució al problema.

Cal fer notar que en el procés es prenen dues decisions, en principi arbitràries:

- L'ordre en el qual s'utilitzen els vehicles.
- L'ordre en que es trien les rutes en cada iteració quan el nombre de rutes generades es superior al nombre de vehicles de que es disposa.

Podem pensar que qualsevol d'aquestes decisions (o ambdues) formen part de l'heurística. És a dir, que prendre-la en un sentit millora normalment la qualitat de la solució. Per exemple, utilitzar sempre primer els vehicles amb més capacitat, o amb més compartiments, o bé triar sempre les rutes més llargues, o amb més arestes. Si així fos, es podria afegir una aleatorització esbiaixada per tal de beneficiar-nos d'aquesta heurística, sense deixar de provar algunes tries que no la seguissin.

```
1.  SAM
2.      Entrada: submapa
3.      Sortida: subsolució
4.  tipusVehicles ← tipusVehicles(submapa)
5.  Nodes ← nodes(submapa)
6.  Mentre tipusVehicles ≠ Ø i nodes ≠ Ø a fer
7.      tipusV ← extreure(tipusVehicles)
        (Pot ser aleatòriament esbiaixat)
8.      llistaRutes ← CWS(nodes, tipusV)
9.      n ← min(mida(llistaRutes), nVehicles(tipusV))
10.     Afegir n rutes de llistaRutes a subsolució
        (Pot ser aleatòriament esbiaixat)
11. Fi Mentre
12. Retornar subsolució
```

Pseudocodi 5. El mètode SAM

La subsolució retornada pel Pseudocodi 5 pot no ser una solució vàlida. Ja hem vist a Pseudocodi 1, línia 7, que existeix un mètode específicament dissenyat per validar les solucions generades.

4.6 Les tècniques de memòria cau

El mètode Multiarrencada, com hem vist a 4.2, fa que es parteixi de zero cada vegada que es troba una solució. Això te dos inconvenients:

- És molt probable la repetició dels mateixos càlculs que, en alguns casos, poden ser relativament complexos.
- La major part de les entitats que es calculen depenen d'aleatoritzacions, ja siguin esbiaixades o no, i és molt poc probable poder arribar al millor valor de cadascuna d'elles en una mateixa iteració.

És per això que s'apliquen tècniques de memòria cau (Juan A. , et al., 2011). En el problema que estem tractant tenim un cas del primer tipus: la millora de rutes que, com veurem a 4.7, pot ser un procediment relativament costós en temps de computació.

Efectivament, el tractament d'un conjunt determinat de nodes que formen una ruta, sense importar l'ordre, pot aparèixer amb certa freqüència en el procés Multiarrencada i, en cada cas, cal intentar millorar-ne el cost canviant la seqüència. Si cada vegada que es millora un conjunt de rutes s'emmagatzema el resultat indexant-lo amb un hash de la pròpia seqüència, només serà necessari fer el càlcul d'optimització en la primera ocurrència.

Per altra banda, en els casos en que Multiarrencada fa una partició dels nodes per a repartir-los entre les bases (Pseudocodi 4), també existeix una probabilitat alta de que la mateixa partició sigui generada gran quantitat de vegades, intervenint en cadascuna d'elles diferents pseudo-aleatoritzacions en el càlcul de la subsolució. Si es disposa en cada moment del millor d'elles per a un conjunt determinat de nodes, en cas d'arribar a una subsolució de més cost que l'emmagatzemada, aquella es pot substituir directament per aquesta.

Aquest procediment assegura que la solució final sempre serà la composició de les subsolucions amb menys cost trobades per a una mateixa partició de nodes. Per la mateixa raó, també s'aconsegueix arribar a una solució d'un cert nivell de qualitat amb menys temps de computació.

De forma genèrica, la lògica aplicada és:

```
1.  TrobaResultatMillorat
2.      Entrada: instància (formada per nodes)
3.      Sortida: instànciaMillorada
4.  nodes ← nodes(instància)
5.  instànciaEnCau ← CercarAcau(nodes)
6.  Si no trobada entitatEnCau llavors
7.      instànciaMillorada ← Millorar(instància)
        (veure 4.7)
8.      AfegirACau(nodes, instànciaMillorada)
9.  altrament
10.     Si cost(instància) < cost(instànciaEnCau) llavors
11.         treureDeCau(nodes)
```

```
12.                 instànciaMillorada ← Millorar(instància)
13.                 AfegirACau(nodes, instànciaMillorada)
14.             altrament
15.                 instànciaMillorada ← instànciaEnCau
16.             Fi si
17.         Fi Si
18.         Retornar instànciaMillorada
```

Pseudocodi 6. Millora de resultats amb memòria cau

4.7 Algorisme 2-opt de millora de rutes

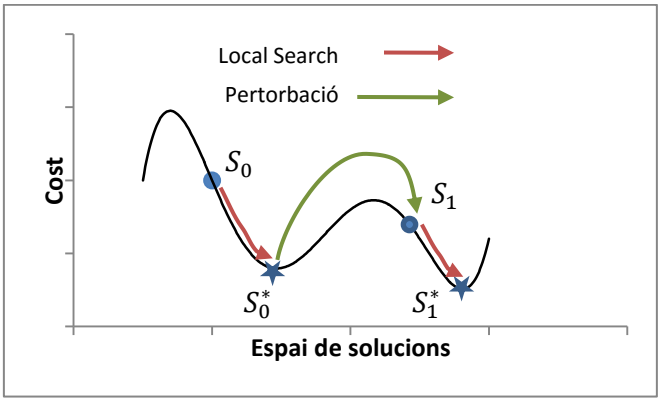
En dependre la qualitat de la solució final de la suma dels costos de les rutes que la composen, un procés de millora local es fa imprescindible. En els problemes derivats del TSP, com és el que estem tractant, s'utilitza per a aquest propòsit de forma generalitzada la metaheurística anomenada *Cerca Local* (LS) (Arya, et al., 2004), que es basa en trobar una solució qualsevol al problema i una relació de veïnatge entre solucions, és a dir, un mètode per trobar una solució S' a partir de la solució S, mantenint contínuament la solució amb cost mínim en S^*.

LS s'atura quan no és possible trobar cap solució veïna amb cost inferior.

```
1.  CercaLocal
2.      Entrada: solució S
3.      Sortida: solució Millorada S*
4.  S* ← S
5.  Mentre ∃S' ∈ veïnat(S*); cost(S') < cost(S*) fer
6.      S* ← S'
7.  Fi Mentre
8.  Retornar S*
```

Pseudocodi 7. El procés de LS

Quan existeixen diferents solucions veïnes es tria la que proporciona una millora més important. Cal fer notar que LS no tan sòls no troba el mínim global (que seria un problema NP-difícil) sinó que s'aturarà en el mínim local més proper a S, que pot ser de baixa qualitat. ILS (Lourenço, Martin, & Stützle, 2010) és una extensió de LS que permet efectuar *pertorbacions* dins del espai de solucions quan el procés arriba a un mínim local.

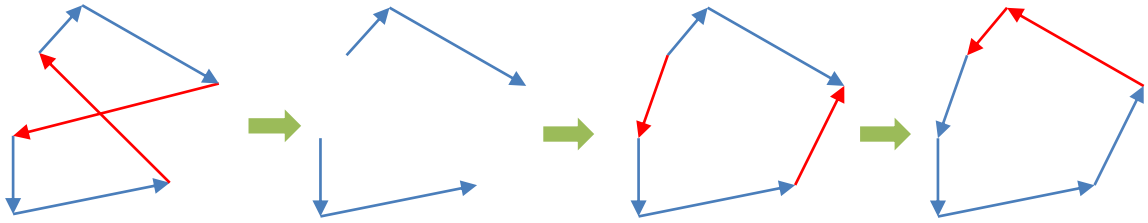


Gràfic 6. Pertorbacions en LS

La pertorbació provoca un "salt" fins a una solució no veïna S_1 que podrà, alhora, derivar per LS a un mínim $S_1^* < S_0^*$

L'algorisme *2-opt* (Croes, 1958) (Johnson & McGeoch, 1997) és una aplicació d'LS en la que la relació de veïnatge està basada en eliminar dues arestes de la ruta i tornar a unir els dos segments resultants d'una forma diferent[1]. La nova ruta serà veïna de la primera i podrà tenir més o menys cost.

Atès que les rutes són grafs orientats, després de la transformació caldrà fer els canvis necessaris pel que respecta a l'orientació de les arestes.



Gràfic 7. Procés 2-opt de millora de rutes

S'ha considerat innecessari aplicar pertorbacions en el cas que tractem perquè l'increment de temps de computació no es veu compensat en els resultats. Això és perquè l'algorisme genera gran quantitat de rutes en les que intervenen els mateixos nodes i normalment es generaran solucions en l'àrea d'atracció de diferents mínims locals. Posteriorment el cau de rutes podrà propagar la solució de cost mínim a altres iteracions del Multiarrencada. Així doncs, podríem dir que el problema es resol amb un ILS on la pertorbació és desconstruir i reconstruir cada vegada la solució sencera i el procés LS és l'aplicació de millora 2-opt localment.

```
1.  MillorarRuta
2.      Entrada: ruta
3.      Sortida: ruta millorada
4.  fiProcés ← fals
5.  Mentre no fiProcés fer
6.      millorMillora ← 0
7.      fiProcés ← cert
8.      Per a cada {aresta1, aresta2; no consecutives} fer
9.          aresta1alt ← aresta(origen(aresta1), origen(aresta2))
10.         aresta2alt ← aresta(fi(aresta1), fi(aresta2))
11.         cost ← cost(aresta1)+cost(aresta2)
12.         costAlt ← cost(aresta1Alt)+cost(aresta2Alt)
13.         Si costAlt < cost llavors
14.             millora ← cost - costAlt
15.             Si millora > millorMillora llavors
16.                 fiProcés ← fals
17.                 millorMillora ← millora
18.                 old1 ← aresta1; old2 ← aresta2
19.                 new1 ← aresta1alt; new2 ← aresta2alt
20.             Fi si
21.         Fi si
```

[1] Atès que el resultat ha de ser un graf simple diferent de l'original, la forma de tornar-los a unir és única.

```
22.             Fi Per
23.             Si no fiProcés llavors
24.                 canviar aresta old1 per aresta new1
25.                 canviar aresta old2 per aresta new2
26.                 invertir arestes que uneixen new1 amb new2
27.             Fi si
28.         Fi Mentre
29.         Retornar r
```

Pseudocodi 8. Millora de rutes amb 2-opt

5. Descripció detallada de l'algorisme

Basats en els conceptes introduïts en la secció 4, ara ens disposem a detallar la implementació del problema duta a terme en el present projecte.

5.1 El conjunt de dades d'entrada

El problema necessita dos conjunts de dades d'entrada: les que intervenen directament en el funcionament de l'algorisme, és a dir, els seus paràmetres, i les que especifiquen la instància que cal resoldre.

5.1.1 Paràmetres de l'algorisme

El programa és capaç de resoldre vàries instàncies en una execució (una per cada línia del fitxer *tests2run.txt*). Definim paràmetres de l'algorisme diferents per a cadascuna de les instàncies.

L'algorisme s'ha intentat dissenyar amb un conjunt el més petit possible de paràmetres. Aquests són:

- Nom de la instància, expressat per un nom de fitxer que conté els paràmetres d'aquesta, que serà definit a 5.1.2
- Paràmetre β_{MAP} relatiu a la formació de submapes (veure 4.4)
- Paràmetre β_{CWS} relatiu al procés CWS (veure 4.3)
- Llavor per a la generació de nombres aleatoris. Totes les crides a mètodes en els que intervé l'aleatorització són fetes amb aquesta llavor. Això permet que dues execucions amb els mateixos paràmetres proporcionin el mateix resultat, cosa que és útil per depurar el programa.
- Temps d'execució: temps, en segons, durant el que el procés Multiarrencada segueix iterant (veure 4.2)
- Temps màxim de servei (TMS) en minuts: (veure 5.1.2 a.7) Si és zero o negatiu, s'utilitza el valor de la instància. Permet que amb un sol fitxer d'instància se'n puguin executar moltes d'elles amb diferents valors per aquest paràmetre.
- Nombre de repeticions de l'execució de la instància. A cadascuna d'elles s'incrementa la llavor de la seqüència pseudoaleatòria en una unitat.

Exemple de contingut del fitxer *tests2run.txt* per a l'execució de 8 instàncies, amb 30 repeticions cadascuna d'elles, i diferint entre elles només en el TMS:

```
# name|beta map|beta CWS|seed|runtime|opTime|nTimes
test_real_01 0.2 0.5 221 40 670. 30
test_real_01 0.2 0.5 221 40 650. 30
```

```
test_real_01 0.2 0.5 221 40 630. 30
test_real_01 0.2 0.5 221 40 608.52 30
test_real_01 0.2 0.5 221 40 590. 30
test_real_01 0.2 0.5 221 40 570. 30
test_real_01 0.2 0.5 221 40 550. 30
test_real_01 0.2 0.5 221 40 530. 30
```

5.1.2 Instància

Una instància del problema es defineix amb la següent estructura de paràmetres:

a. Paràmetres generals del problema
    1. Nombre de clients
    2. Nombre de bases
    3. Nombre de magatzems
    4. Nombre de productes
    5. Nombre de tipus de vehicle
    6. Mode d'obtenir la distància i el temps de trànsit entre dos nodes:
        - 0: es calcula la distància euclidiana. També es calcula el temps en funció de la velocitat indicada a aquest mateix grup de paràmetres
        - 1: s'obté la distància real de la matriu de distàncies (f) i el temps real de la matriu de temps (g)
    7. TMS
    8. Velocitat mitjana (en km/min, només utilitzat si Mode=0)
    9. Cabal de càrrega i de descàrrega (en l/min)
    10. Temps de preparació de càrrega o descàrrega (en min)
    11. Temps de penalització (intra-ciutat) en transports dins d'una mateixa localitat (en min)
b. Definició dels tipus de vehicle
    1. Índex correlatiu començant per 0
    2. Cost fix
    3. Cost variable (per km recorregut)
    4. Accés a magatzem
        - 0: no pot carregar a magatzem
        - 1: pot carregar a magatzem
    5. (Grup) Capacitat dels compartiments (en l)
c. Definició dels nodes client
    1. Índex correlatiu de node començant per 0
    2. Nom de la localitat, per calcular la penalització intra-ciutat
    3. Coordenada x (en unitats de pantalla)
    4. Coordenada y (en unitats de pantalla)
    5. (Grup) Demandes per producte (en l)
    6. (Grup) Índexs dels tipus de vehicle que hi poden accedir
d. Definició dels nodes base
    1. Índex correlatiu de node, seguint la seqüència del dels clients
    2. Nom de la localitat, per calcular la penalització intra-ciutat
    3. Coordenada x (en unitats de pantalla)
    4. Coordenada y (en unitats de pantalla)

5. (Grup) Existències inicials de productes (en l)
6. (Grup) Estocs mínims de productes (en l)
7. (Grup) Composició de la flota <nombre> <tipus vehicle> ...

e. Definició dels nodes magatzem
   1. Índex correlatiu de node, seguint la seqüència del de les bases
   2. Nom de la localitat, per calcular la penalització intra-ciutat
   3. Coordenada x (en unitats de pantalla)
   4. Coordenada y (en unitats de pantalla)

f. Matriu de distàncies (si Mode=1)

g. Matriu de temps (si Mode=1)

Com a exemple d'instància completa del problema, veure 6.2.2

5.2 L'estructura de dades bàsica

Aquests dos conjunts de dades especificats són llegits del fitxer i emmagatzemats en les classes d'instància única Test (per els paràmetres de l'algorisme) i Input (per les dades de la instància)

La classe Algorithm conté bàsicament el procés principal de càlcul, i és qui es relaciona directament amb les classes de suport de les dades.

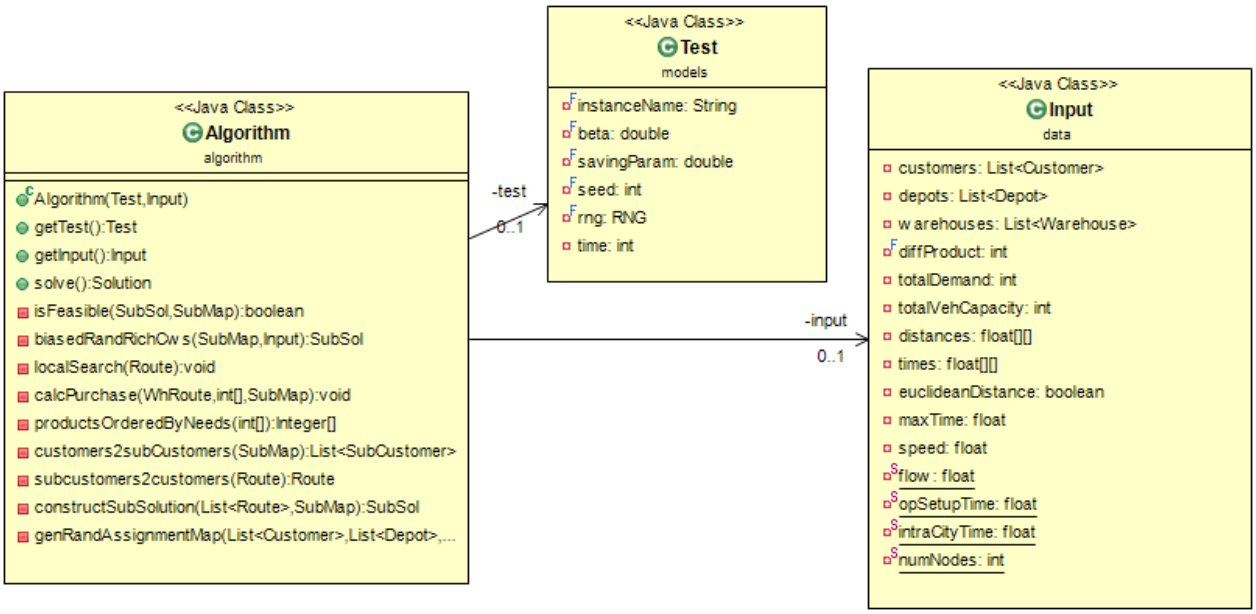


Diagrama 1. Les dades inicials del problema

A partir d'aquest procés principal, es comença a treballar amb nodes, arestes i rutes i es modela una classe per a cadascun d'ells.

Cada ruta incorpora una llista amb les arestes de que es composa. La primera aresta té com a origen la base i la darrera aresta la té com a final. Si la ruta passa per un magatzem (anomenarem a aquests tipus de ruta "ruta de provisió") aquest serà sempre el final de la primera aresta.

Cada aresta té dues relacions amb el node, indicant l'origen i el final, tanmateix, cada node té dues relacions amb l'aresta indicant les arestes que l'uneixen amb la base.

Tant el node com l'aresta contenen un atribut de pertinença a la ruta i les arestes tenen associada l'aresta inversa.

Tot això es fa visible en el diagrama UML:

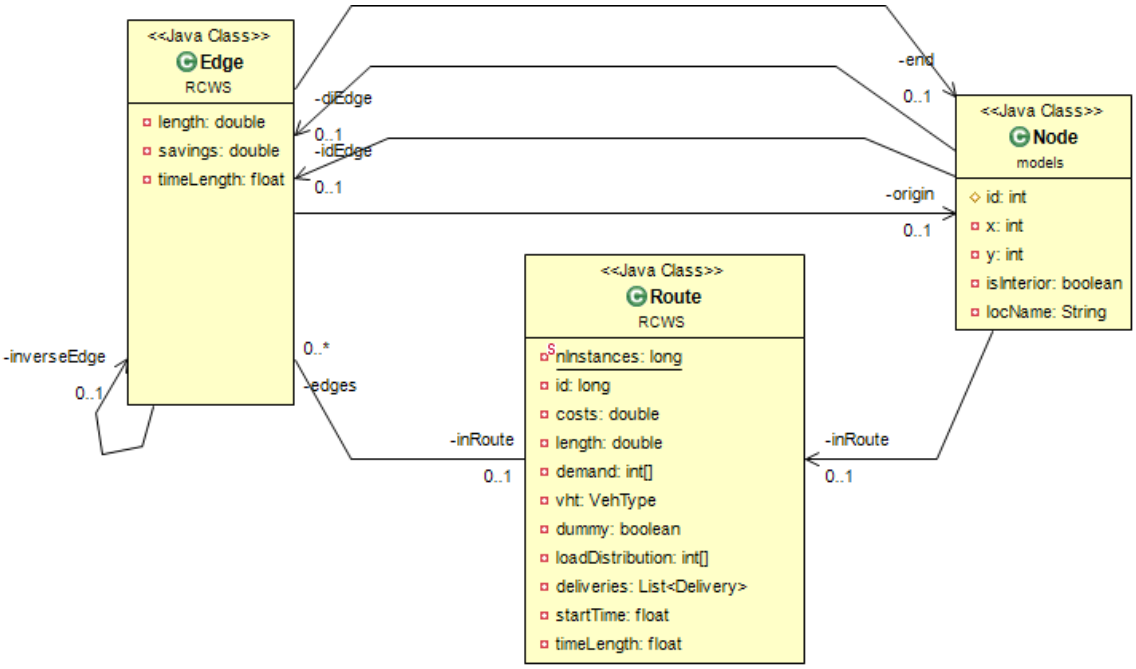


Diagrama 2. Estructura Ruta-Aresta-Node

Pel que fa als nodes, n'hi ha de diferents tipus: clients, bases, magatzems i subclients. Cada subclient (que és la projecció d'un client sobre un producte) te associat el client corresponent i cada client disposa d'una llista amb tots els seus subclients.

Tot plegat ho veiem gràficament en el següent diagrama UML:

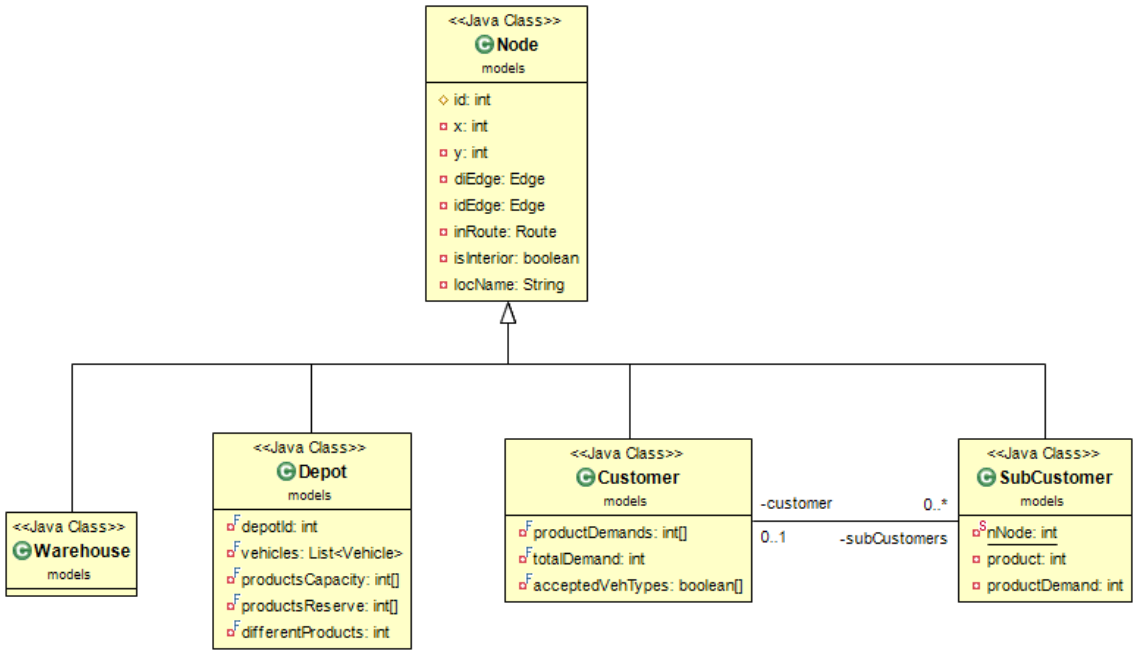


Diagrama 3. La classe Node i les seves especialitzacions

Per un altra banda, existeixen els grups de rutes, que incorporen totes les rutes operades per un mateix vehicle en un repartiment.

Un tipus especial de ruta, com hem vist abans, és la ruta de provisió. En el diagrama UML que s'incorpora a continuació podem veure que aquestes rutes disposen d'un atribut addicional que indica la compra dels diferents productes que fan al magatzem.

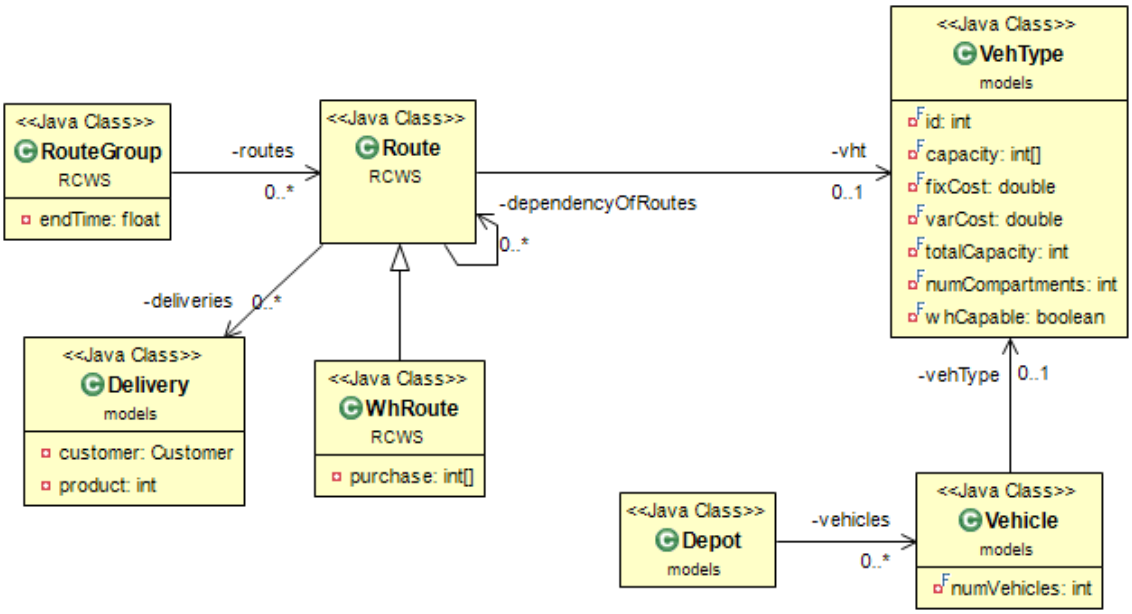


Diagrama 4. Ruta i Vehicle

També queda palès en el diagrama que la classe Vehicle és associativa entre la base i el tipus de vehicle i incorpora el nombre de vehicles de cada tipus disponibles a cada base.

Finalment tenim les estructures que suporten la solució del problema. Una solució es composa de diferents subsolucions (una per cada base). Cadascuna d'aquestes és suportada per un submapa, que també n'hi ha un per cada base i conté la informació relativa a l'assignació de clients a bases i planificació de rutes que veurem més endavant, i tots els submapes junts formen el mapa.

El submapa també incorpora informació sobre les entregues que cal fer per cobrir la demanda dels clients i una estructura que indica les diferents disponibilitats de productes en funció del temps per l'arribada de les rutes de provisió.

Vegem-ho gràficament per a una millor comprensió:

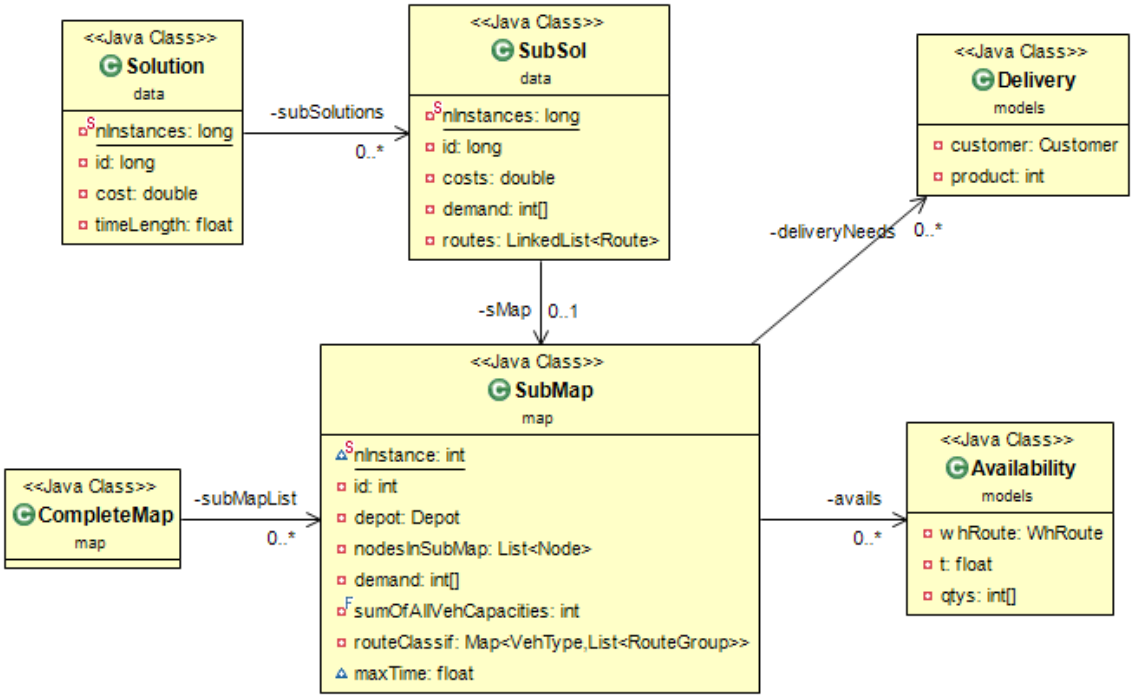


Diagrama 5. SubSolucions i SubMapes

5.3 L'algorisme a un primer nivell

En un primer nivell tenim, bàsicament, el bucle Multiarrencada. Prèviament es calculen els components que són invariants: estalvis pel mapeig, llista de clients, llista de bases i creació de subclients.

El procés Multiarrencada s'executa tantes vegades com sigui possible en un interval de temps prefixat (paràmetre de l'algorisme) i en cada iteració es genera un mapa i es resol cadascun dels submapes de que es composa, verificant la viabilitat de la subsolució. En cas de ser viable, es millora cercant-la en al cau i s'afegeix a la solució global. Una partició de clients determinada es repeteix durant el procés Multiarrencada. El procediment garanteix que cada solució de cadascuna de les diferents particions de clients està formada per les millors subsolucions que s'han trobat fins al moment de cada subconjunt de la partició (submapa).

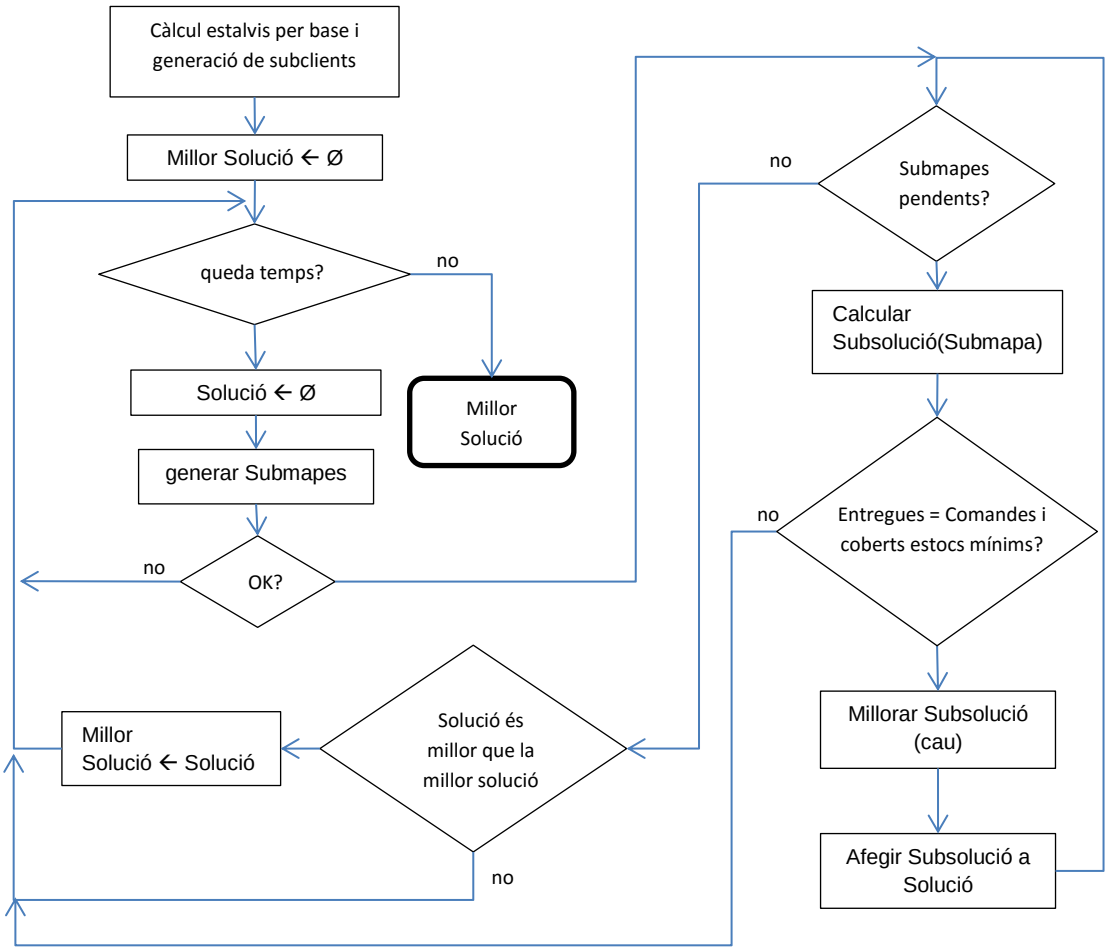
El test de viabilitat de cada subsolució està basat en dues condicions:

- El fet de que cada producte es subministri en una sola entrega a cada client i que cada client només pugui ser subministrat per una base fa que, per a cada submapa, el nombre d'entregues hagi de ser el mateix que el nombre de comandes (comptant una entrega o comanda per cada producte).
- L'estoc al final del servei ha de ser superior o igual a l'estoc mínim exigit per a cada producte.

Si totes les subsolucions són viables, es composa una solució i es compara amb la millor trobada fins al moment, actualitzant-la si la millora.

Al final del procés tenim la millor solució trobada.

Visualitzat com a diagrama de flux:



Flow-chart 1. Bucle principal Multiarrencada

Notem que existeixen dues condicions que provoquen l'avortament prematur de la iteració: l'impossibilitat de poder fer un mapeig i l'impossibilitat de resoldre un submapa.

5.4 La generació de submapes

Entenem per submapa un conjunt de nodes format per un o més clients i una base (els magatzems no s'assignen en aquest estadi). La unió de tots els submapes formen el mapa, que és el resultat del procés partició del conjunt total de clients per a assignar cadascun d'ells a una de les bases.

Aquest procés segueix l'heurística indicada en el punt 4.4, basada en que cada base escull el node que provoca un estalvi més alt, comparant la longitud del trajecte fins el node amb la del trajecte des de la més propera de la resta de les bases fins al mateix node.

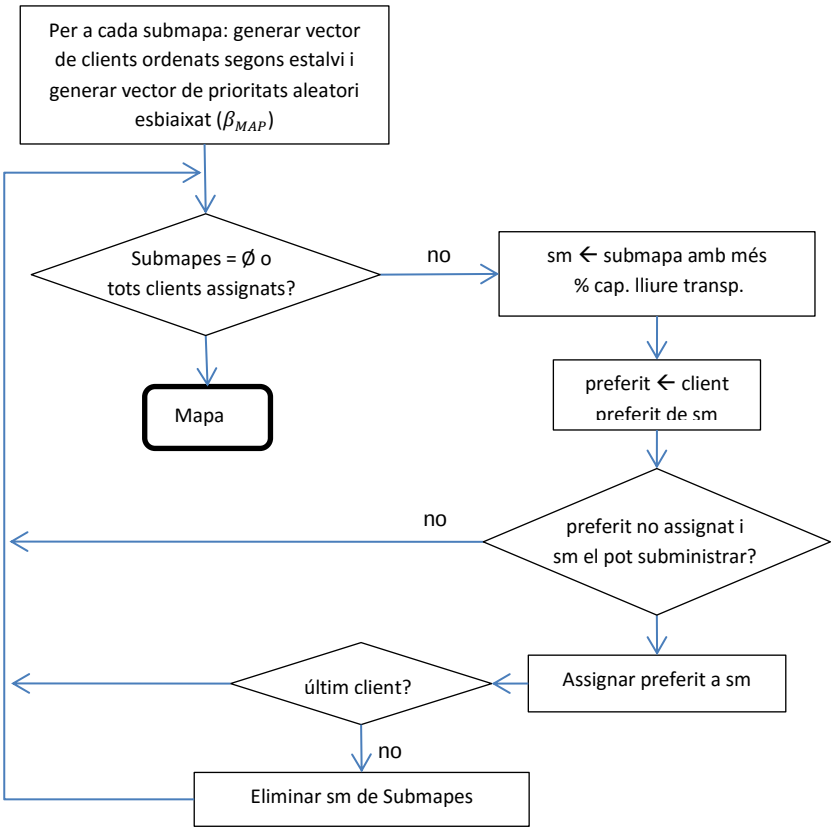
Fem notar que estem comparant distàncies i no costos, atès que els costos depenen del tipus de vehicle utilitzat i en el present moment encara no el coneixem.

L'heurística requereix definir una magnitud que indiqui quina és la base que tria node en cada iteració. En la implementació actual s'ha optat per que aquesta sigui el percentatge de capacitat de transport lliure, considerant que els vehicles fan un sol viatge. Aquest criteri

afavoreix el repartiment en funció de la mida de la flota. L'hem considerat millor que el de premiar les existències d'estoc més elevades degut a que (quasi) sempre existeix la possibilitat de visitar el magatzem quan hi ha carència d'algun producte. En no ser possible assignar serveis a vehicles en aquest punt del procés, es considera la capacitat de transport com la suma total de les capacitats dels compartiments de tots els vehicles. De la mateixa manera, el decrement de capacitat de transport quan s'assigna un client és la suma de les demandes de producte d'aquest client.

Una altra opció descartada ha estat considerar la capacitat lliure de transport en valor absolut. El motiu és que quan la demanda total és inferior a la capacitat de transport d'alguna de les bases, aquesta tria la totalitat dels clients.

Atès que un client només accepta un subconjunt dels tipus de vehicle, a més de comprovar que encara no està assignat a cap altra base, també cal verificar que la base receptora disposa, al menys, d'un tipus de vehicle que el pot subministrar. D'aquesta manera s'evita l'avortament prematur de la iteració per impossibilitat de resoldre el submapa que s'hagués generat.



Flow-chart 2. Generació de submapes

Bàsicament existeix una única situació en que s'avorta el mapatge: quan no és possible assignar tots els nodes. Això passarà en el cas que un client no accepti cap tipus de vehicle dels disponibles a cap base.

5.5 La solució del submapa

La solució del submapa és la part més complexa de l'algorisme i la que requereix de més enriquiments i modificacions dels mètodes especificats de forma genèrica en l'apartat 4. Veurem les dificultats que presenten factors característics del problema com poden ser la limitació del servei en el temps, la possibilitat d'inserir magatzems a les rutes i el multitrajecte, entre d'altres.

5.5.1 La reducció al problema de flota homogènia

Per cada mapa que es configura[2] s'intenta generar una solució al problema. El procés està basat en el SAM, vist a 4.5, però adaptat a les següents peculiaritats:

- Existeix una limitació de temps per a fer el repartiment que caldrà tenir en compte a l'hora de validar la ruta.
- Els vehicles poden efectuar més d'una ruta, o sigui, que no s'han d'extreure definitivament de la llista quan s'hagin utilitzat.
- Cal inserir rutes de provisió quan sigui necessari, el que implica que cal controlar en tot moment la utilització que es fa de l'estoc de la base.
- Les bases poden requerir estocs mínims.

Vegem l'algorisme en el Pseudocodi 9, a les línies del qual es fan les referencies d'aquest subapartat.

En cada iteració, després de solucionar la ruta amb flota homogènia amb un tipus de vehicle determinat (línia 15), aquest es retira només en el cas que no s'hagi afegit cap nova ruta a la solució global. No obstant, quan s'afegeix una ruta de provisió, tots els vehicles són habilitats una altra vegada (línia 25).

Els nodes client, com en l'algorisme SAM genèric, també es retiren quan són servits (línia 21).

Per altra banda, l'algorisme actualitza en tot moment la demanda total (per a cada producte) del submapa i coneix l'estoc mínim requerit al final de la sessió (línia 22).

Totes aquestes dades ens permeten dissenyar un bucle principal que acaba quan es produeix qualsevol de les següents condicions:

- No queda cap node client per servir i la base ha assolit l'estoc mínim estipulat.
- No queda cap vehicle seleccionable per a continuar iterant.
- No existeix magatzem, quan aquest és necessari.
- No existeixen tipus de vehicle aptes per a magatzem, quan aquest és necessari.

La conjunció d'aquestes restriccions és la condició de sortida del bucle (línia 9). A cada pas d'aquest bucle, una vegada resolta la instància amb flota homogènia, es processen , en ordre aleatori, cadascuna de les rutes generades.

En cas de ser ruta de provisió, es calcula la càrrega de productes que adquireix al magatzem (línia 17).

[2] Evidentment, un mateix mapa es molt probable que es repeteixi moltes vegades al llarg de l'execució Multiarrencada. Aquest fet fa que es puguin generar moltes subsolucions de cada submapa.

En aquest moment coneixem tots els transvasaments de producte que caldrà fer per a les operacions de càrrega i de descàrrega i podem calcular el temps necessari per a aquestes.

A continuació s'intenta classificar (línia 19)segons les següents opcions: reutilitzant un vehicle que ja ha gestionat una ruta o utilitzant un vehicle que encara no s'ha fet servir. Naturalment, aquests vehicles són del tipus que s'està tractant.

Veurem més endavant que en la classificació hi ha cert grau d'aleatorietat i que la lògica emprada depèn de si la ruta és de provisió o no. La classificació de les rutes és vigent durant tota la resolució del submapa.

Quan la classificació és exitosa s'afegeix la ruta a les que formen la subsolució, s'eliminen els nodes ja servits i s'actualitza la demanda pendent del submapa (línies 21-23)

En el cas d'haver afegit una ruta de provisió (línia 24), es rehabiliten tots els vehicles, atès que les condicions de disponibilitat de producte han canviat positivament i tipus de vehicles descartats anteriorment ara poden tornar a ser útils. També cal afegir la nova disponibilitat de productes associada al moment en que arriba a la base, concepte que també veurem més endavant.

Finalment, en el cas en que no s'hagi pogut classificar cap de les rutes generades pel procés CWS de flota homogènia, s'inhabilita per a posteriors iteracions el tipus de vehicle tractat (línia 31).

Tot seguit il·lustrem el que hem exposat amb pseudocodi, més adient que el flow-chart en aquest cas.

```
1.  Resoldre submapa
2.      Entrada: submapa
3.      Sortida: subsolució
4.  rutes ← Ø
5.  barrejar vehicles
6.  demanda pendent ← demanda del submapa
7.  nodes client ← clients(submapa)
8.  vehicles seleccionables ← vehicles(base)
9.  Mentre no fi fer (veure condicions de finalització en el text, més amunt)
10.     vehicle ← triar aleatoriament de vehicles seleccionables
11.     nodes ← nodes client + base
12.     Avaluar necessitat deruta de provisió
13.     Si necessària llavors nodes ← nodes + cerca millor magatzem Fi Si
14.     Generar estalvis(nodes)
15.     rutesFH ← resoldre amb flota homogènia (submapa, vehicle)
16.     Per a cada ruta de rutesFH fer
17.         Si és ruta de provisió llavors calcular compra fi si
18.         Afegir temps de carrega i descàrrega
19.         ok = classificar(ruta)
20.         Si ok llavors
21.             Eliminar clients(ruta) de nodes client
22.             Demanda pendent ← demanda pendent – demanda(ruta)
23.             Afegir ruta a rutes
24.             Si ruta és de provisio llavors
```

```
25.                         vehicles seleccionables ← vehicles(base)
26.                         Afegir disponibilitat (temps d'arribada, compra-demanda)
27.                     Fi si
28.                 Fi si
29.             Fi per
30.             Si no s'ha afegit cap ruta llavors
31.                 Eliminar vehicle de vehicles seleccionables
32.             Fi si
33.         Fi mentre
34.         Construir subsolució (rutes, submapa)
```

Pseudocodi 9. Resolució del submapa

5.5.2 La generació de subclients

Observem que el mètode gestiona eficientment la característica multiproducte del problema, atès que, en la resolució de cada instància, CWS és capaç de rebutjar les unions de rutes en que la resultant no pot ser suportada pel tipus de vehicle emprat.

No obstant això, cal modificar-lo per tal que aprofiti la flexibilització de l'enunciat del problema, que permet varies visites a un mateix client efectuades per diferents vehicles, sempre que parteixin de la mateixa base.

Per resoldre aquest punt s'ha definit el concepte de subclient, que és el node amb la ubicació d'un client i la demanda d'un sol producte. Passar de subministrar clients a subclients implica una reducció del problema que fa que cada node només demani un producte i que la formació de rutes s'adapti millor a la flota disponible en cada base.

Així doncs, en el Pseudocodi 9, les referències a nodes client les entenem com a subclients. No obstant, abans de donar la solució final, els subclients es tornen a convertir a clients mercès a que, com veiem al Diagrama 3, el subclients conserven una associació amb els clients als qui pertanyen.

En aquesta reconversió es té en compte que els subclients d'un mateix client poden no estar agrupats seqüencialment en la ruta atès que, encara que la ruta arriba a aquest punt ja optimitzada (veure 5.5.6), clients amb la mateixa ubicació geogràfica podrien tenir els subclients barrejats.

Addicionalment, en aquest procés es calculen les entregues de material que es realitzen a cada client, que seran posteriorment útils per a verificar la factibilitat de la solució i per enriquir la informació aportada en la solució final.

5.5.3 Les rutes de provisió

La ruta de provisió és una especialització de la ruta que permet totes les operacions d'aquestes, és a dir, subministrar productes a clients, fusionar-se amb altres rutes, etc., però permet una operació addicional que és la càrrega de productes al magatzem i la posterior descàrrega dels mateixos a clients i/o a la base. El conjunt de quantitats carregades en el magatzem l'anomenem *compra*.

La compra de productes al magatzem, per imperatiu de l'enunciat del problema, és farà sempre amb el vehicle buit. Això obliga a que el magatzem sigui sempre el primer node de la ruta.

Donat que cada ruta pot abastir-se d'un magatzem com a màxim, cal establir un criteri per a seleccionar aquest magatzem d'entre tots els disponibles quan sigui necessari. Aquest és: triar aleatòriament el magatzem, però amb un biaix ($\beta = 0.5$) favorable als que tenen una menor suma de distàncies fins a la resta dels nodes.

La decisió de quines quantitats de producte són comprades en una visita al magatzem és un dels aspectes que milloren la qualitat de la solució final. Cal trobar una heurística el màxim d'efectiva per a aquesta operació. Com a premissa inicial, també obligada per l'enunciat, el vehicle utilitzarà tota la seva capacitat de càrrega.

El càlcul de la compra es pot demorar fins que la ruta està completada. Durant la fase de generació de rutes l'algorisme CWS assigna una distribució de càrrega a totes les rutes, incloent les de provisió. És a dir, que es coneix quins productes van a cada contenidor del vehicle.

Això simplifica una primera part de l'algorisme de càrrega: emplenar totalment els contenidors utilitzats pel repartiment assignat al vehicle.

No obstant, pot haver-hi un subconjunt de contenidors que vagin buits degut a que, en un principi, no són necessaris pel repartiment que s'està executant. En un cas extrem, es possible que tots ells estiguin buits. Un cas habitual és el de vehicles que no són aptes per repartir a clients i sòls es poden utilitzar per aprovisionar la base.

En la primera fase, a més d'assignar la part lliure dels compartiments a la càrrega, també hem generat un vector de necessitats de cada producte, restant a la demanda pendent la disponibilitat actual de la base i actualitzant-lo segons anàvem emplenant compartiments. Ara només cal iterar pels compartiments buits i emplenar cadascun d'ells amb el producte que en aquest moment té més necessitats segons aquest vector.

Per aconseguir més variabilitat, i atès que estem utilitzant una heurística, l'assignació del següent compartiment a emplenar es fa de forma aleatòria. Cada vegada que s'omple un compartiment s'actualitza el vector de necessitats i es torna a cercar el producte amb més necessitats pel proper compartiment.

```
1.  Calcular compra
2.      Entrades: demanda pendent
3.      Sortides: compra
4.  necessitats ← demanda pendent – estoc final de la base
5.  compra ← 0
6.  NcompartimentsBuits ← 0
    Fase 1: compartiments utilitzats
7.  Per a cada compartiment fer
8.      Producte ← distr_càrrega(compartiment)
9.      Si compartiment no buit llavors
10.         vol ← volum(compartiment)
11.         compra(producte) ← compra(producte) + vol
12.         necessitats(producte) ← necessitats(producte) - vol
13.     altrament
14.         NcompartimentsBuits ← NcompartimentsBuits + 1
15.     Fi si
16. Fi per
```

```
                Fase 2: compartiements buits
17.             Mentre NcompartimentsBuits > 0 fer
18.                 compartiment ← compartiment buit aleatori
19.                 vol ← capacitat(compartiment)
20.                 producte ← producte amb més necessitats
21.                 compra(producte) ← compra(producte) + vol
22.                 necessitats(producte) ← necessitats(producte) - vol
23.                 distr_càrrega(compartiment) ← producte
24.                 NcompartimentsBuits ← NcompartimentsBuits - 1
25.             Fi mentre
26.             Retornar compra
```

Pseudocodi 10. Càlcul de la compra de les rutes de provisió

5.5.4 El multitrajecte

La possibilitat de que cada vehicle pugui executar més d'una ruta en un mateix repartiment comporta la necessitat d'implementar una estructura que suporti la seva activitat. Aquesta serà el grup de rutes: una llista que allotja rutes de forma seqüencial i en ordre temporal (veure Diagrama 4).

5.5.5 La classificació de les rutes

Per tenir una visió complerta de l'estat del problema en qualsevol moment és necessari conèixer les rutes assignades a cada vehicle, el seu inici, el seu fi, els vehicles de cada tipus que s'estan utilitzant i els que queden lliures, entre d'altres informacions.

Hem vist a Pseudocodi 9 (línia 19) que el procediment de solució del submapa confia la tasca de classificar la ruta i d'avaluar la factibilitat d'incorporar-la a la subsolució a un mètode que hem anomenat *classificar()*.

Aquest mètode està basat en una estructura que ubica cada ruta en un vehicle i mantenint la seqüència temporal de totes aquelles que el comparteixen. Bàsicament es tracta d'una taula que permet accedir, per tipus de vehicle, a una llista de grups de rutes amb tants elements com vehicles del tipus estiguin utilitzats en el moment.

Gràficament:

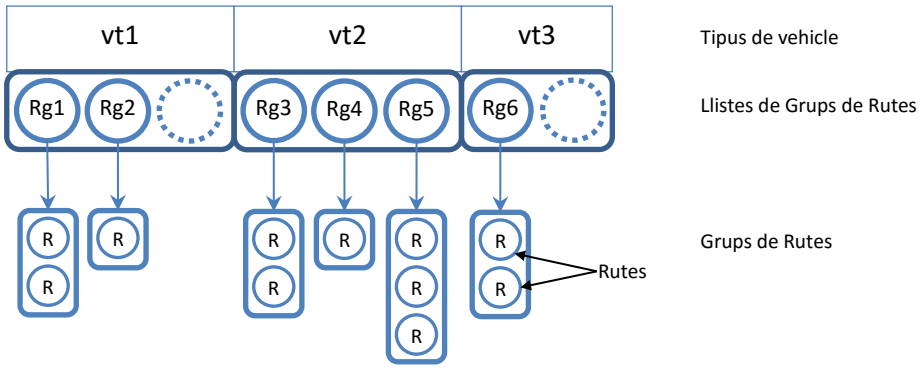


Diagrama 6. Taula de classificació de les rutes

En el Diagrama 6 podem veure una taula de classificació de rutes per a un problema amb 3 tipus de vehicle i amb 3, 3 i 2 vehicles de cadascun dels tipus. En el moment present s'estan utilitzant 2 vehicles del tipus 1, 3 vehicles del tipus 2 i 1 vehicle del tipus 3 (cada vehicle

utilitzat es visualitza com un grup de rutes). Per exemple, el vehicle número 3 del tipus de vehicle 2 efectua una seqüència de 3 rutes.

Aquesta taula de classificació ens permet realitzar operacions bàsiques per a la resolució del problema com són:

- Afegir una ruta a la solució, si compleix les condicions (mètode *putRoute()* de la classe *SubMap*, que torna un valor booleà fals si no és possible).
- Calcular l'hora mínima d'inici d'una ruta (mètode *calcStartTime()* de la classe *SubMap*.

Pel que fa a l'addició d'una ruta, es realitza segons el Pseudocodi 11

```
1.  Afegir ruta
2.      Entrada: ruta, vehicle
3.      Sortida: ok
4.  tipusVehicle ← tipus(vehicle)
5.  grupsRutes ← classificador(tipusVehicle)
6.  restenVehicles ← mida(grupsRutes) < nVehicles(vehicle)
7.  ok ← fals
8.  Si ruta és de provisió llavors
9.      Si restenVehicles i durada(ruta) < tempsMàximServei llavors
10.         gr ← nou grupRutes
11.         Afegir ruta a gr
12.         Afegir gr a grupsRutes
13.         ok ← cert
14.     altrament
15.         gr ← grupRutes amb mínima durada
16.         Si durada(gr) + durada(ruta) < tempsMàximServei llavors
17.             Temps inici de ruta ← temps final de rg
18.             Afegir ruta a gr
19.             ok ← cert
20.         Fi si
21.     Fi si
22. altrament
23.     a ← Cercar disponibilitat per a demanda(ruta)
24.     Si trobada a llavors
25.         Si temps(a) + durada(ruta) < tempsMàximServei llavors
26.             rnd ← pseudoaleatori (0…9)
27.             Si restenVehicles o rnd < 5 llavors
28.                 ok ← afegir en grupRutes existent aleatori
29.             Fi si
30.             Si no ok i restenVehicles llavors
31.                 ok ← cert
32.                 Afegir en nou grupRutes
33.             Fi si
34.         Fi si
35.     Fi si
36. Fi si
37. Retornar ok
```

Pseudocodi 11. Afegiment d'una ruta al classificador

En el cas de les rutes de provisió es prioritza que s'iniciïn el més d'hora possible per tal de generar disponibilitats més aviat i, així, possibilitar la creació d'altres rutes.

Per decidir si una ruta no de provisió és afegida a un grup iniciat o a un de nou quan ambdues opcions són possibles, es fa aleatòriament, amb $p = 0.5$, per així aconseguir un altre factor de variabilitat pel Multiarrencada, atès que no hi ha cap raó evident per decidir si una opció és millor que l'altra. Per la mateixa raó, quan es decideix afegir-la a un grup existent, aquest també es tria de forma aleatòria i equiprobable.

5.5.6 L'aplicació del mètode CWS

Hem vist a 4.3 el mètode heurístic CWS per calcular una solució al problema CVRP més genèric. En el cas que estem estudiant tenim un conjunt de peculiaritats que es converteixen en restriccions a l'hora de validar l'execució del pas que podríem considerar més atòmic del procediment: la unió de dues rutes a partir de l'aresta d'estalvis.

En el Pseudocodi 3 (línia 19) es verifica que la unió sigui possible o, millor podríem dir, adequada per a la finalitat del problema.

En el cas més genèric, les condicions que cal verificar són:

a. Els dos vèrtexs[3] no pertanyen a la mateixa ruta.
b. Els dos vèrtexs són exteriors en les seves rutes (directament connectats amb la base).
c. La demanda agregada d'ambdues rutes es pot transportar amb el tipus de vehicle que s'està tractant.

En el problema enriquit cal tenir en compte la possible presència de magatzems a la ruta, que no poden deixar d'estar connectats amb la base. Això porta a exigir:

d. Si un dels dos vèrtexs és un magatzem, aquest ha de pertànyer a una ruta *dummy*[4].

Finalment, la condició més complexa que ens obliga a més desenvolupament de l'heurística estàndard:

e. La ruta resultant ha de caber, en temps de servei, en algun dels vehicles del tipus de vehicle tractat.

No tindria cap sentit que CWS generés rutes de les quals es té la certesa de que no podran ser utilitzades posteriorment per SAM.

Per validar aquesta condició cal avaluar primer el temps total de la ruta. Això implica que:

- Cal calcular-la com si el resultat de l'avaluació fos positiu, però treballant sobre una còpia de cada ruta, perquè finalment es pugui revertir la unió si el resultat no és favorable.
- Cal millorar-la amb el procés LS. Altrament l'avaluació podria donar un fals negatiu. Lo habitual és aplicar el procés LS només a les rutes definitives del submapa.
- Cal calcular els temps de transferència de productes.
- Finalment, cal verificar que es pot existeix un instant en que la suma d'ambdues demandes és disponible, i que aquest instant sumat al temps total de la ruta és inferior

[3] Ens referim als vèrtexs de l'aresta obtinguda del vector d'estalvis.
[4] L'algorisme SAM, en la seva crida al CWS, només pot incloure una ruta de provisió.

al TMS. Com a última opció, també s'accepta la unió si en la subsolució que s'està tractant ja existeix un ruta de provisió (que podria abastir els productes que habilitessin una disponibilitat adient) i aquesta acaba prou d'hora com per a encabir la ruta resultant. De fet, encara no es coneix la compra de la ruta de provisió, però actuem d'una forma optimista per tal de no descartar cap solució vàlida, tot i el risc de que es tracti d'un fals positiu, que no es detecti fins una fase posterior de l'algorisme.

Si la unió és finalment validada, la funció de validació (Pseudocodi 12) modifica la ruta del primer paràmetre conseqüentment i retorna un booleà cert. Si no és així, deixa les rutes intactes i retorna fals.

En l'algorisme principal CWS, quan es produeix la unió, només resta actualitzar convenientment els costos, longituds i temps de la primera ruta i esborrar la segona.

```
1.  Validar unió
2.      Entrada: ruta1, ruta2, aresta
3.      Sortida: validació_OK, ruta1
        Condició 1
4.  Si ruta1 = ruta 2 llavors retornar fals fi si
5.  node1 ← origen(aresta)
6.  node2 ← final(aresta)
        Condició 2
7.  Si node1 o node2 són interiors llavors retornar fals fi si
        Condició 3
8.  Si node1 és magatzem i ruta1 no és dummy llavors retornar fals fi si
9.  dc ← distribucióCàrrega(ruta1, ruta2)
        Condició 4
10. Si dc és nul·la llavors retornar fals fi si
        Condició 5
11. r1 ← còpia(ruta1)
12. r2 ← còpia(ruta2)
13. r1 ← unir r1 i r2 (com a Pseudocodi 3, línies 20-30)
14. millorar(r1)
15. temps ← temps(r1) + temps de transferencia de producte de ruta1 i ruta2
16. horaInici ← hora de disponibilitat de la suma de les demandes
17. Si horaInici + temps > TMS llavors
18.     Si r1 és ruta de provisió o no hi ha ruta de provisió llavors
19.         Retornar fals
20.     altrament
21.         horaInici ← hora de disponibilitat de la ruta de provisió
22.         Si horaInici + temps(ruta de provisió) > TMS llavors
23.             Retornar fals
24.         Fi si
25. Fi si
26. distribució de càrrega(ruta1) ← dc
27. Recalcular costos, tems i longitud de ruta1
28. Treure costos(aresta base-node2) a ruta2
29. node2.ruta ← ruta1
30. Per a cada node final de les arestes de r2 fer
31.     node ruta ← ruta1
32. Fi per
```

```
33.     Afegir demanda(ruta2) a demanda(ruta1)
34.     ruta1 dummy ← fals
35.     ruta1 arestes ← r1 arestes
36.     Recalcular nodes interiors de ruta1
37.     Retornar cert
```

Pseudocodi 12. Mètode per validar la unió de dues rutes

5.5.7 El càlcul del temps total de la ruta

El temps total d'una ruta incorpora tres components:

a. El temps que necessita el vehicle per recórrer el trajecte.

El problema rep aquesta dada, ja sigui per la matriu de temps o pel càlcul a partir de la distància i la velocitat, que es considera homogènia en el problema.

b. El temps de transvasament de producte, ja sigui de càrrega o de descàrrega.

És una funció lineal del volum transvasat

$$t = aV + b\ ,$$

on V és volum transvasat, a és cabal de càrrega o descàrrega (200 l/s, en una aplicació real) i b és el temps de gestió administrativa (300 s).

Es calcula a nivell de ruta quan és seleccionada de la solució de la flota homogènia.

c. El temps de transport quan la totalitat del trajecte (inici i final de l'aresta) es produeix dins d'una mateixa població.

Es tracta d'un temps de penalització per trànsit dins del casc urbà. Cada node té assignat un nom a nivell d'entrada de dades. La penalització s'aplica quan el nom és idèntic, però els dos nodes no són subclients del mateix client.

Aquest component es calcula a nivell d'aresta quan aquesta és creada.

5.5.8 La distribució de càrrega dels vehicles

L'heterogeneïtat i multicompartimentació de la flota de transport són factors diferencials del problema que tractem pel que fa al CVRP. En el procés de fusió de dues rutes del CWS, tot i que la flota s'ha reduït a homogènia, cal calcular si l'estructura i capacitat del compartiments del vehicle que s'està utilitzant permeten allotjar la demanda d'ambdues rutes.

No existeix cap limitació pel que fa als productes que es poden transportar a cada compartiment i un mateix producte es pot allotjar en més d'un compartiment. En aquest subproblema és important no rebutjar cap fusió de rutes que no sigui realment impossible, pel que s'ha optat per utilitzar un algorisme que calcula permutacions dels compartiments del vehicle fins que en troba una que permet carregar seqüencialment la suma de les demandes de les dues rutes.

Vist en pseudocodi:

```
1.      calculaDistribucióCàrrega
2.          Entrada: tipus de vehicle, ruta1, ruta2
3.          Sortida: productesEnComp
4.      llistaComp ←Generar llista de compartiments del vehicle
```

```
5.      productesEnComp ← Generar vector de productes indexat per compartiment
6.      Iterar
7.          permutació ← Trobar següent permutació de llistaComp
8.          prod ← 0
9.          comp ← 0
10.         iterar
11.             demanda ← demanda(prod, ruta1) + demanda(prod, ruta2)
12.             Mentre demanda > 0 i comp < nombre comp del tipus de vehicle fer
13.                 compartiment ← permutació[comp]
14.                 demanda ← demanda – capacitat(compartiment)
15.                 productesEnComp[índex(compartiment)] ← prod
16.                 comp ← comp + 1
17.             Fi mentre
18.             prod ← prod + 1
19.         Fins que demanda > 0 o prod >= nombre de productes
20.     Fins que fi de permutacions o demanda <= 0
21.     Si demanda > 0 llavors productesEnComp ← nul fi si
22.     Retornar productesEnComp
```

Pseudocodi 13. Càlcul de la distribució de la càrrega en els compartiments dels vehicles.

Encara que l'algorisme és de cost $O(n!)$, cosa en principi poc recomanable en el mètode Multiarrencada. No obstant, el nombre màxim de compartiments en un vehicle sol ser de 2 o 3, podent arribar com a màxim a 6, el qual suposaria 720 iteracions de molt baix cost de computació. Aquesta quantitat sembla admissible si es considera que ajuda a trobar una solució quan aquesta existeix.

5.5.9 La disponibilitat de producte a la base

Així com en el problema simple CVRP es disposa d'un estoc inicial a cada base que va disminuint a mesura que van sortint camions, en el nostre cas aquest estoc pot augmentar quan arriba una ruta de provisió.

Addicionalment, perquè pugui iniciar-se una ruta cal, per una banda, que el vehicle estigui disponible i, per altra, que hi hagi suficient estoc a la base. En aquestes condicions, el decrement d'estoc produït per l'inici de la ruta pot produir-se en qualsevol moment.

Serà necessari, aleshores, conèixer la disponibilitat de producte en qualsevol instant de temps per poder avaluar la possibilitat de que es pugui dur a terme.

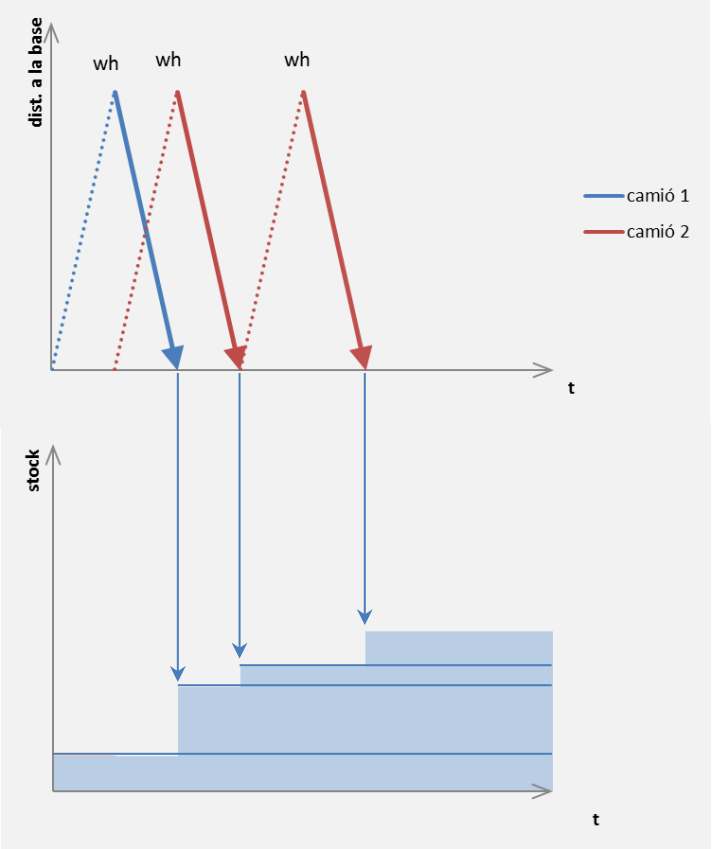
Així doncs, en aquest apartat veurem més a fons el que ja s'havia introduït a 5.5.1 referent a la disponibilitat de productes respecte al temps.

L'estructura principal que conté les dades és una llista, cadascun dels elements de la qual és el que anomenem una *disponibilitat*, modelada amb els següents atributs:

- Una ruta de provisió, que és la raó per la que existeix. La disponibilitat inicial de la base és la única que no incorpora ruta de provisió.
- Un valor temporal t, que és el seu inici de validesa. El fi de validesa és l'inici de la següent disponibilitat.
- Un vector de quantitats de producte, que són les disponibles a partir del moment t.

Podem representar les rutes, de forma simplificada, en funció del temps i de la distància a la base (en aquests moments no ens interessa els nodes pels que passen) i, simultàniament i en el mateix eix temporal, podem representar les disponibilitats dels productes en un diagrama de barres apilades (per simplificar el gràfic utilitzarem un sol producte, tot i que els procediments que veurem són generalitzables a n productes, simplement repetint-los per a cadascun d'ells).

Segons això, representarem, per exemple, 3 rutes de provisió de les quals 2 es duen a terme pel mateix vehicle:



Gràfic 8. Afegiment de disponibilitats en arribar rutes de provisió

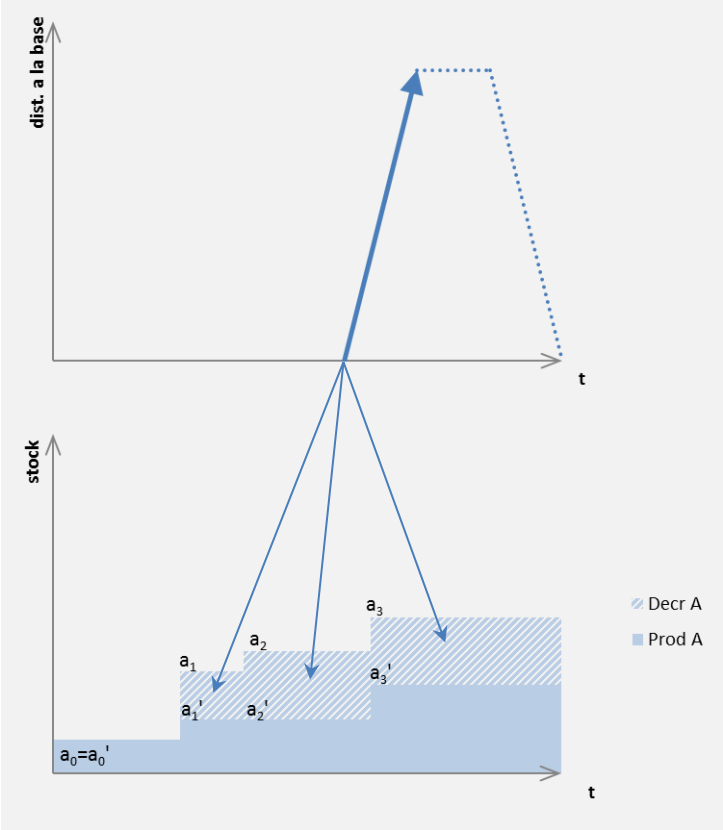
En el Pseudocodi 14 es pot apreciar com cada ruta de provisió genera una nova disponibilitat amb inici en el tems d'arribada i en la que les quantitats disponibles són la suma de les existents en aquell moment i de les aportades per la ruta. Aquest increment de la disponibilitat es propaga a les disponibilitats futures, però no a les passades. Fixem-nos també en l'existència d'un estoc inicial a la base en forma d'una disponibilitat vàlida per a tot l'interval temporal.

```
1.  AfegirDisponibilitat
2.      Entrada: ruta r (de provisió)
3.      Sortida:
4.  quantitats ← compra(ruta) – demanda(ruta)
5.  Si quantitats > 0 per a algun producte llavors
6.      horaArribada ← horaInici(ruta) + durada(ruta)
7.      i ← Trobar posició per insertar la nova disponibilitat
```

```
8.          Si i=0 llavors
9.              novesQtys ← vector nul
10.         altrament
11.             novesQtys ← qtys(dispo(i-1))
12.         Fi si
13.         Afegir disponibilitat en posició i,  ruta , horaArribada, novesQtys
14.         Per a disponibilitats(j); j>=i fer
15.             qtys ← qtys + novesQtys
16.         Fi per
17.     Fi si
18.     retorn
```

Pseudocodi 14. Afegiment d'una nova disponibilitat en arribar una ruta de provisió

Representem ara l'inici d'una ruta, que representa un decrement de l'estoc de la base, i veiem com això repercuteix sobre les disponibilitats existents.



Gràfic 9. Decrement de disponibilitats en partir una ruta

Quan parteix una ruta en un instant t_0, es produeix:

- Un decrement equivalent al vector demanda $\vec{d}$ d'aquesta ruta en les quantitats de l'última disponibilitat a_m amb $t_m <= t_0$ (en el Gràfic 9, a_2). O sigui: $\vec{q}'_m = \vec{q}_m - \vec{d}$
- A les disponibilitats posteriors a_n; $n > m$ (a_3 en el gràfic) el decrement també serà $\vec{d}$ per a totes elles: $\vec{q}'_n = \vec{q}_n - \vec{d}$
- A les anteriors a_n; $n < m$ (a_0 i a_1 en el gràfic), les quantitats definitives seran $\vec{q}'_n = \overrightarrow{\min}(\vec{q}_m, \vec{q}_n)$. Veiem que això afecta a a_1 però no a a_0.

El decrement de disponibilitats ve donat pel Pseudocodi 15:

```
1.  Treure Disponibilitat
2.      Entrada: ruta, primeraDispo
3.      Sortida:
4.  qtys ← demanda(ruta)
5.  novesQtys ← estoc actual de la base - qtys
6.  Per a cada producte fer
7.      novaQty ← novesQtys[producte]
8.      anteriors ← fals
9.      Per a cada disponibilitat en ordre decreixent fer
10.         Si anterior llavors
11.             qty ← min(qty, novesQtys[producte])
12.         altrament
13.             qty ← novesQtys[producte]
14.         Fi si
15.         Si disponibilitat = primeraDispo llavors anteriors ← cert fi si
16.     Fi per
17. Fi Per
18. Eliminar disponibilitats nules
19. retorn
```

Pseudocodi 15. Decrement de disponibilitats en partir una ruta

Podem veure en el pseudocodi que ja es parteix del coneixement previ de la disponibilitat a_m. Això és degut a que, abans de treure disponibilitat s'ha hagut de comprovar si es possible afegir la ruta i això comporta el fet de cercar-la.

5.6 L'optimització de les rutes

El mètode 2-opt utilitzat per optimitzar les rutes permet ser aplicat individualment a cada ruta sense tenir en compte en quina posició es troben dins del seu grup. Aleshores, la característica multitrajecte no requereix cap modificació en lo exposat fins al moment referit a aquest aspecte.

Tanmateix, existeix el requeriment de que la base hagi de ser sempre el primer i l'últim node. De la mateixa manera, pot haver-hi un magatzem inserit entre la base i el primer node client sense que aquell pugui restar situat a cap altre lloc. Tot això obliga a modificar l'algorisme genèric exposat en Pseudocodi 8.

L'algorisme 2-opt es basa en intercanviar els extrems finals de dues arestes. Cal prevenir aquest intercanvi quan qualsevol de les arestes sigui la que uneix la base amb el magatzem, tenint també en compte que durant el procés de millora aquestes arestes poden estar en qualsevol posició i també invertides.

En acabar la millora de la seqüència de nodes, pel fet recentment esmentat, també cal reorganitzar la ruta de manera que el primer i últim node torni a ser la base i, si hi ha magatzem, aquest sigui el segon node.

Aquest procés es realitza amb l'ajut d'una segona llista d'arestes a la que es van traspassant les arestes de la ruta mentre l'origen no sigui una base. Quan aquesta es troba, estarà situada en la primera posició, atès que les anteriors s'han eliminat per passar-les a la llista auxiliar.

Com a següent pas, s'afegeixen al final de la ruta les arestes de la llista auxiliar en el ordre en que es troben.

Finalment, s'inverteix tota la ruta si l'última aresta té origen a la base i fi al magatzem.

6. Proves

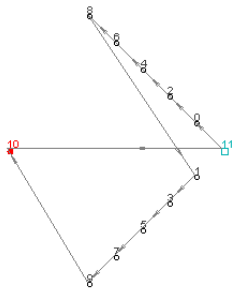
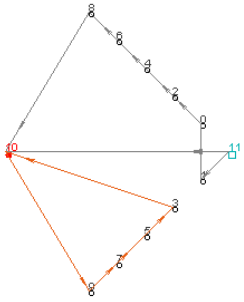
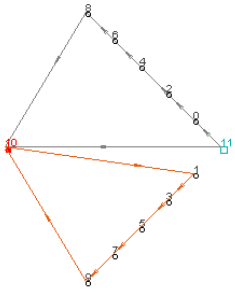
6.1 Cas teòric simplificat

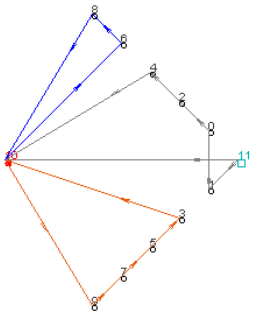
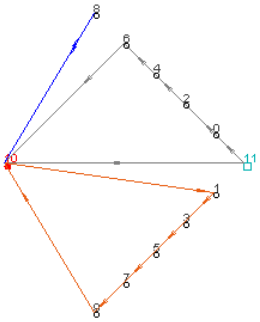
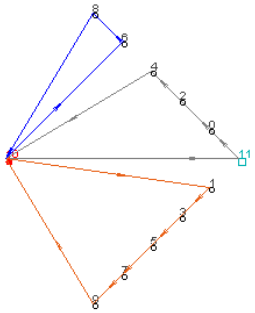
Primerament presentem aquest cas teòric de 10 clients, una base i un magatzem amb el propòsit de comprovar la correctesa de l'algorisme pel que respecta al magatzem i a la gestió del TMS. Tanmateix es mostra gràficament la influència d'aquest paràmetre en el cost de la solució del problema.

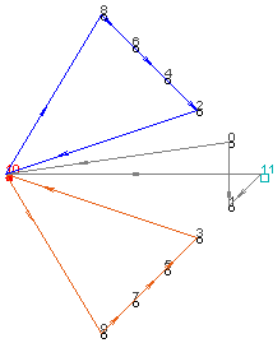
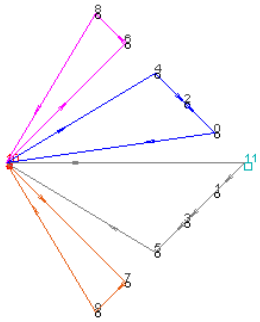
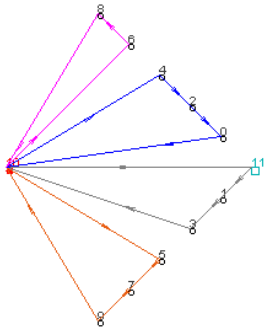
Per simplificar, establim un cabal de càrrega/descàrrega infinit i un temps de preparació nul. La base (de color vermell en els gràfics) disposa d'un nombre il·limitat de vehicles d'un sol tipus, cadascun d'ells capaç de subministrar tots els clients (grisos). Les existències inicials suficients només per a 9 dels 10 clients, la qual cosa obliga la visita al magatzem (turquesa). Tots els clients accepten el tipus de vehicle de la base. El diferent color de les rutes simbolitza que són servides per diferents vehicles.

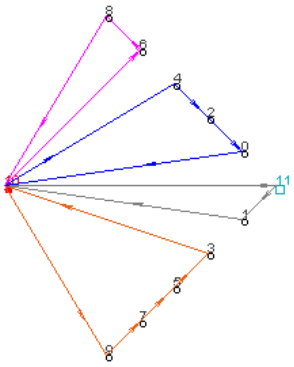
Comencem les proves amb un TMS il·limitat: la intenció és anar reduint el TMS sempre per sota del temps de servei de l'última solució calculada.

Taula 1. Solucions del cas simplificat

TMS	∞	140,70 min	95,00 min
Solució			
	Cost: 168,85; t= 140,72 min	Cost: 196,50; t=95,43 min	Cost: 197,30; t= 87,10 min

TMS	87,00 min	83,60 min	80,40 min
Solució			
	Cost: 246,87; t= 83,65 min	Cost: 247,67; t=80,48 min	Cost: 247,67; t= 77,33 min

TMS	77,30 min	77,00 min	75,30 min
Solució			
	Cost: 256,41; t= 77,03 min	Cost: 298,04; t=75,31 min	Cost: 301,38; t= 71,48 min

TMS	71,47 min	68,60 min
Solució		Sense solució
	Cost: 307,58; t= 68,70 min	

S'aprecia com el decrement del TMS comporta un increment del nombre de rutes què fa que la solució tingui més cost.

6.2 Cas real

A continuació es planteja un cas real que ha estat proposat per una empresa del sector del transport d'hidrocarburs. Les dades es basen en les operacions dutes a terme en una jornada en la que es van servir comandes d'un total de 31 clients des de 3 bases.

Especifiquem els paràmetres generals del problema i les característiques dels vehicles i de les bases:

Taula 2. Paràmetres generals

Nombre de clients	31. Cap d'ells accepta el vehicle més gran i 3 d'ells tampoc accepten el mitjà
Nombre de bases	3, una d'elles amb estoc mínim d'un dels productes
Nombre de magatzems	1
Nombre de productes	4 (A, B, C, C+)
Nombre de tipus de vehicles	3, que anomenarem *gran*, *mitjà* i *petit*
Cabal de càrrega	200 l/min
Temps d'establiment de trajecte	5 min
Temps de penalització intra-ciutat	10 min

Taula 3. Tipus de vehicles

Tipus	Capacitat Compartiments (l)
Gran (5 compartiments)	10000, 8000, 8000, 4000, 2000
Mitjà (2 compartiments)	6000, 5000
Petit (2 compartiments)	2000, 2000

Taula 4. Característiques de les bases

Base	Vehicles	Estocs inicials	Estocs de seguretat
1	1 petit, 2 mitjans, 1 gran	12000, 17000, 18000, 14000	15000, 0, 0, 0,
2	1 mitjà, 1 gran	0, 8000, 1000, 0	0, 0, 0, 0
3	2 mitjans	25000, 7000, 22000, 0	0, 0, 0, 0

6.2.1 Solució implementada per l'empresa del sector

L'empresa proporciona els districtes postals i les localitats de cadascun dels serveis efectuats, considerant com a servei l'entrega d'un sol producte (subclients). També indica les rutes que s'han dut a terme incloent-hi els temps aproximats de pas, la base des de la que s'ha iniciat cadascuna d'elles i el vehicle utilitzat. No disposem del nom del client ni de l'adreça completa, informacions que ens permetrien fer una separació exacta dels clients. L'empresa també ens proporciona les incompatibilitats client-tipus de vehicle.

El criteri per realitzar aquesta separació ha sigut empíric, considerant, en principi, un client per a cada codi postal diferent, excepte en els casos següents, en els que es considera que hi ha diversos clients en un mateix codi postal:

- Serveis a un mateix codi postal des de diferents bases.
- Incompatibilitats client-vehicle diferents a un mateix codi postal.
- Entregues d'un mateix producte a un mateix codi postal, exceptuant aquelles que són consecutives i amb diferència de temps de menys de 5 minuts.

Amb tot això, s'arriba a la següent taula de serveis, en la que s'inclou l'assignació a les bases:

Taula 5. Serveis executats per la solució manual

Base	Node	ciutat	CP	Prd A	Prd B	Prd C	Prd C+	Total general
d32	**c00**	ambroa	15313			790		790

	c01	la coruña	15007		1.617			1.617
	c02	arteixo	15690		722			722
	c03	betanzos	15300	1.000				1.000
	c04	abegondo	15318	633	229	732	1.000	2.594
	c05	coristanco	15147	639				639
	c06	negreira	15830	2.080				2.080
	c07	betanzos	15300	275				275
	c08	coiros	15316	5.167				5.167
	c09	oza-cesuras	15390		625			625
	c10	abegondo	15318		1.676		1.000	2.676
	c11	abegondo	15318	1.000				1.000
	c12	abegondo	15318	522				522
	c13	cerceda	15185			500		500
	c14	irixoa	15314			400		400
	c29	coiros	15316				500	500
	c30	oza-cesuras	15390			892		892
d33	**c15**	melide	15800	1.550	1.200	735		3.485
	c16	melide	15809			722		722
	c17	melide	15809			1.000		1.000
	c18	lugo cent	27001			900		900
	c19	milladoiro	15895			800		800
	c20	oza-cesuras	15390		1.030			1.030
d34	**c21**	lorbe	15177			1.500		1.500
	c22	lubre	15166			760		760
	c23	sada	15160			1.000		1.000
	c24	bergondo	15165			933		933
	c25	mera	15178			437		437
	c26	mera	15178			792		792
	c27	lorbe	15177			349		349
	c28	oleiros	15173			1.800		1.800
	c31	la coruña	15004			500		500
Total general				**12.866**	**7.099**	**15.542**	**2.500**	**38.007**

La distribució de rutes que es va dur a terme, dissenyada amb mètodes manuals, va ser:

Taula 6. Distribució de rutes

base	tipus	grup	ruta	Node	Dist	t
d32	**mitjà**	1	1	c01	30,15	40,535
				c02	17,38	31,41
				c03	31,42	41,3
				c04	12,94	27,615
				c05	59,92	54,165
				c06	36,12	54,82
				d32	76,88	98,335
			2	c04	1	16,145
				c07	12,94	25,825
				c08	8,79	44,365
				c09	15,21	32,475
				c10	23,57	41,45
				c11	1	20
				c12	1	17,61
				d32	1	62,47

		Suma 1			329,32	608,52
	gran	2	3	w35	30,73	190,4
				d32	30,73	190,4
		Suma 2			61,46	380,8
	petit	3	4	c04	1	18,66
				c13	26,75	38,22
				c29	48,5	47,87
				d32	21,52	37,84
			5	c04	1	20
				c10	1	20
				d32	1	25
			6	c30	23,57	37,53
				c00	26,96	39,52
				c14	7,36	16,17
				d32	22,38	47,38
		Suma 3			181,04	348,19
d33	**mitjà**	4	7	c15	1	22,75
				c16	5,63	19,16
				c17	1	20
				c18	51,09	65,18
				c19	144,14	97,3
				d33	0	89,83
			8	c15	1	24,675
				c20	35,89	44,33
				d33	35,89	54,005
		Suma 4			275,64	437,23
	gran	5	9	w35	82	223,07
				d33	82	223,07
		Suma 5			164	446,14
d34	**mitjà**	6	10	c21	14,02	30,95
				c22	15,21	28,45
				c23	4,62	19,68
				c24	6,29	21,635
				c31	23,43	29,08
				c25	13,13	29,715
				c26	1	18,96
				c27	5,12	17,245
				c28	7,97	27,2
				d34	4,03	53,885
		Suma 6			94,82	276,8
Total general					**1106,28**	

La taula especifica:

- La base de cada submapa, que disposa d'un nombre d'unitats de tipus de vehicle.
- El tipus de vehicle, que subministra tants grups de rutes com unitats disposi en la base.
- Els grups de rutes, que són conjunts de rutes servides seqüencialment per un mateix vehicle físic.
- Les rutes per servir els productes als clients.
- Els nodes numerats seqüencialment, amb diferent inicial segons el tipus: clients (c), bases (d) i magatzems (w).

- La distancia recorreguda (en km) en cada tram, obtinguda mitjançant l'API *directions* de Google Maps[5] en una aplicació separada. Per evitar la creació de rutes amb cost zero, s'assigna una distància de 1 km quan el codi postal de la sortida és el mateix que el de la arribada.
- El temps total de cada tram (en minuts), també calculat a partir de la mateixa API. Incorporen els temps de: trànsit, transferència de producte, preparació i penalització intra-ciutat (en aquest cas, intra-CP).

Atès que l'empresa no ha proporcionat els costos fixes i variables per a cada tipus de vehicle, hem d'assumir que els primers són inexistents i els segons són d'una unitat monetària (um) per km.

Amb tot, la solució implementada per l'empresa te un cost de 1.016 um. El temps de servei ha de considerar-se com el del grup de rutes amb més durada: 608,52 min

6.2.2 Solució calculada per l'algorisme

A partir de les dades proporcionades pel client, generem el fitxer d'entrada de dades tal com s'ha definit en el punt 5.1:

Taula 7. Paràmetres d'entrada del programa

32 3 1 4 3 1 608.52 1.2 200. 5. 10.	Paràmetres de la instància
0 0. 1.0 0 2000 2000 1 0. 1.0 1 5000 6000 2 0. 1.0 1 10000 8000 8000 4000 2000	Tipus de Vehicle
00 15313 58 33 0 0 790 0 0 1 01 15007 34 19 0 1617 0 0 0 1 02 15690 30 35 0 722 0 0 0 1 03 15300 49 34 1000 0 0 0 0 1 04 15318 43 48 633 229 732 1000 0 1 05 15147 11 58 639 0 0 0 0 1 06 15830 11 108 2080 0 0 0 0 1 07 15300 49 34 275 0 0 0 0 1 08 15316 53 42 5167 0 0 0 0 1 09 15390 52 57 0 625 0 0 0 1 10 15318 43 48 0 1676 0 1000 0 1 11 15318 43 48 1000 0 0 0 0 1 12 15318 43 48 522 0 0 0 0 1 13 15185 29 58 0 0 500 0 0 1 14 15314 53 34 0 0 400 0 0 15 15800 62 110 1550 1200 735 0 0 1 16 15809 63 101 0 0 722 0 0 1 17 15809 63 101 0 0 1000 0 0 1 18 27001 95 88 0 0 900 0 0 1 19 15895 20 121 0 0 800 0 0 1 20 15390 52 57 0 1030 0 0 0 1 21 15177 41 13 0 0 1500 0 0 1 22 15166 44 25 0 0 760 0 0 1 23 15160 45 20 0 0 1000 0 0 1 24 15165 47 28 0 0 933 0 0 1 25 15178 41 18 0 0 437 0 0 1 26 15178 41 18 0 0 792 0 0 1 27 15177 41 13 0 0 349 0 0 1 28 15173 41 23 0 0 1800 0 0 1 29 15316 53 42 0 0 0 500 0 30 15390 52 57 0 0 892 0 0	Clients

[5] La crida a GoogleMaps es realitza mitjançant un mètode http tipus GET des d'un full de càlcul Excel (mòdul VBA) que proporciona directament les matrius de distàncies i de temps de recorregut entre cada dos districtes postals. Exemple de GET per calcular el trajecte del CP 15165 al 15178: https://maps.googleapis.com/maps/api/directions/xml?origin=15165,Spain&destination=15178,Spain

31 15004 34 17 0 0 500 0 0 1	
32 15318 43 48 12000 17000 18000 14000 15000 0 0 0 1 0 2 1 1 2 33 15800 62 110 00000 08000 01000 00000 0 0 0 0 1 1 1 2 34 15176 40 26 25000 07000 22000 00000 0 0 0 0 2 1	Bases
35 15140 31 24	Magatzem

Per brevetat, i perquè no tenen més interès, no incloem les matrius de distancies i de temps.

Calculem una solució amb el mateix temps de servei que la de l'empresa. El resultat s'inclou en l'annex 0.

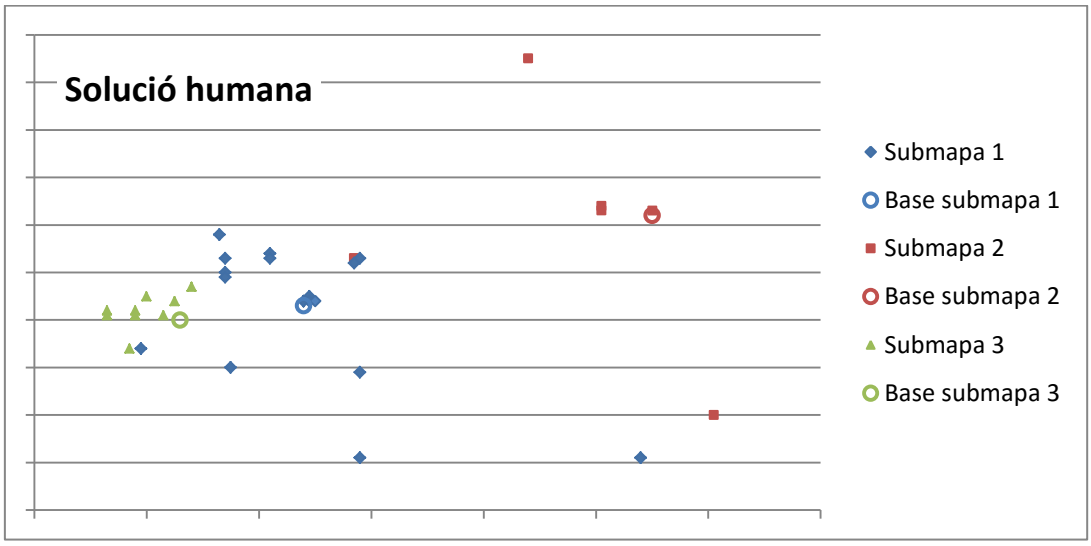
Aquesta consta de 3 subsolucions.

La primera d'elles, que utilitza els 3 tipus de vehicle, comença enviant un vehicle gran al magatzem, per cobrir la demanda i l'estoc mínim que te assignat i, a continuació, assigna 3 grups de rutes: 2 per als vehicles mitjans, amb una seqüència de 3 rutes i una ruta respectivament, i un altre grup per al vehicle petit amb una sola ruta. Verifiquem que la disponibilitat final del producte 1 és de 15764, superior a l'estoc mínim requerit.

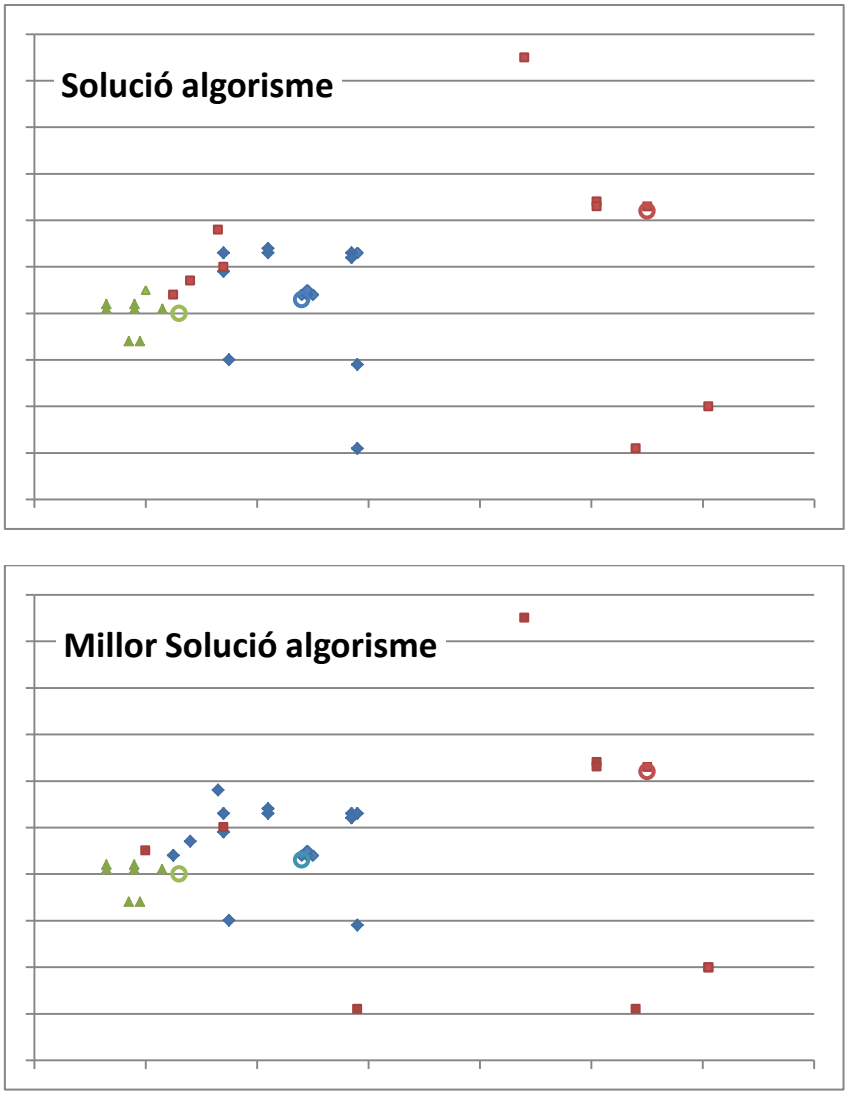
La segona subsolució, que és la més llarga en temps i, per tant, la que proporciona el temps de servei, opera amb només un vehicle mitjà, tot i que la base en disposa de 2. Aquest fa un viatge al magatzem amb una llarga tornada a la base en la que serveix 10 clients. Posteriorment recarrega per servir un clients més.

La tercera subsolució realitza 2 grups de rutes operats pels vehicles mitjans, cadascun amb una sola ruta.

Fixem-nos, al Gràfic 10[6], en els mapatges generats per la solució humana, l'algorisme i una tercera solució corresponent a una de les millors, que s'ha calculat en un batch de repeticions amb diferents llavors. El temps de servei màxim aplicat es correspon al que abans hem calculat per a la solució humana de 608,52 min. Observem que les bones solucions tenen alguns dels clients lluny de la base que els abasteix (en l'exemple ho podem veure en el conjunt de color vermell) bàsicament perquè cal una càrrega al magatzem.



[6] En aquest gràfic, els punts que corresponen a una mateix codi postal s'han separat artificialment. Les bases tenen fons blanc.



Gràfic 10. Diferents mapatges per a la instància de prova

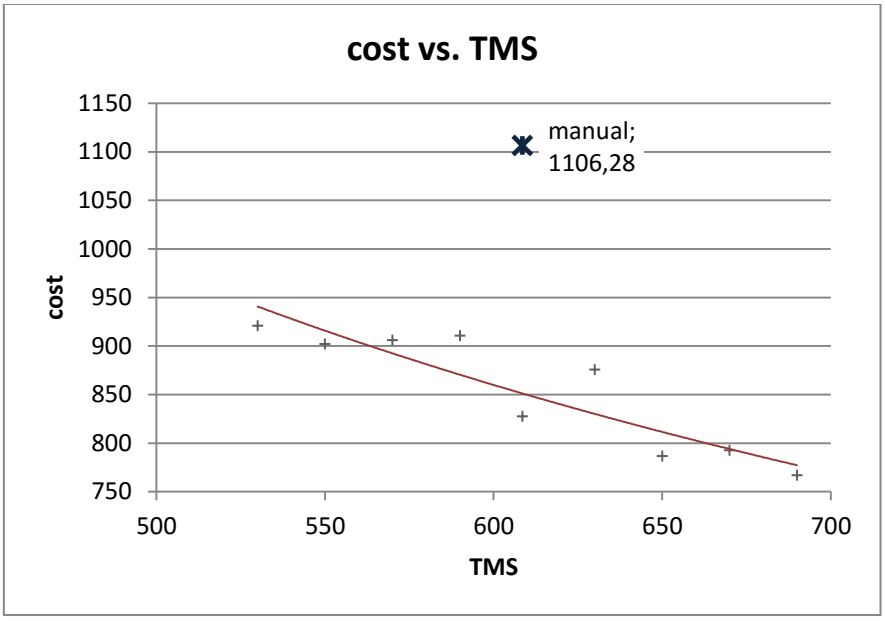
Són mapatges molt similars, amb els clients agrupats en torn a les bases, com era d'esperar. No obstant, es pot apreciar que alguns dels punts blaus de la solució humana es tornen vermells en les algorísmiques. Aquest és el seguit de clients subministrats pel vehicle que torna del magatzem, potser una de les raons més importants per a la diferència de costos a favor de la segona, degut a que el planificador humà, probablement, no pot valorar adequadament tots els factors que intervenen en la composició d'una bona solució. Entre ells, aprofitar el pas pel magatzem, element clau en aquesta instància del problema.

Calculem diferents costos variant la limitació de temps (grup de rutes amb més durada). El temps de procés de cada execució s'ha fixat en 40 seg.

Després d'un nombre bastant elevat d'experiments, s'ha arribat a la conclusió que el biaix de la generació pseudoaleatòria requerit per a arribar a solucions prou bones ha de ser baix. En particular, el corresponent al procés de mapatge que ha proporcionat millors resultats ha estat $\beta_{MAP} = 0{,}2$. Pel que fa al biaix del procés CWS, s'ha adoptat un valor de $\beta_{CWS} = 0{,}5$.

Taula 8. Solucions obtingudes comparades amb la manual

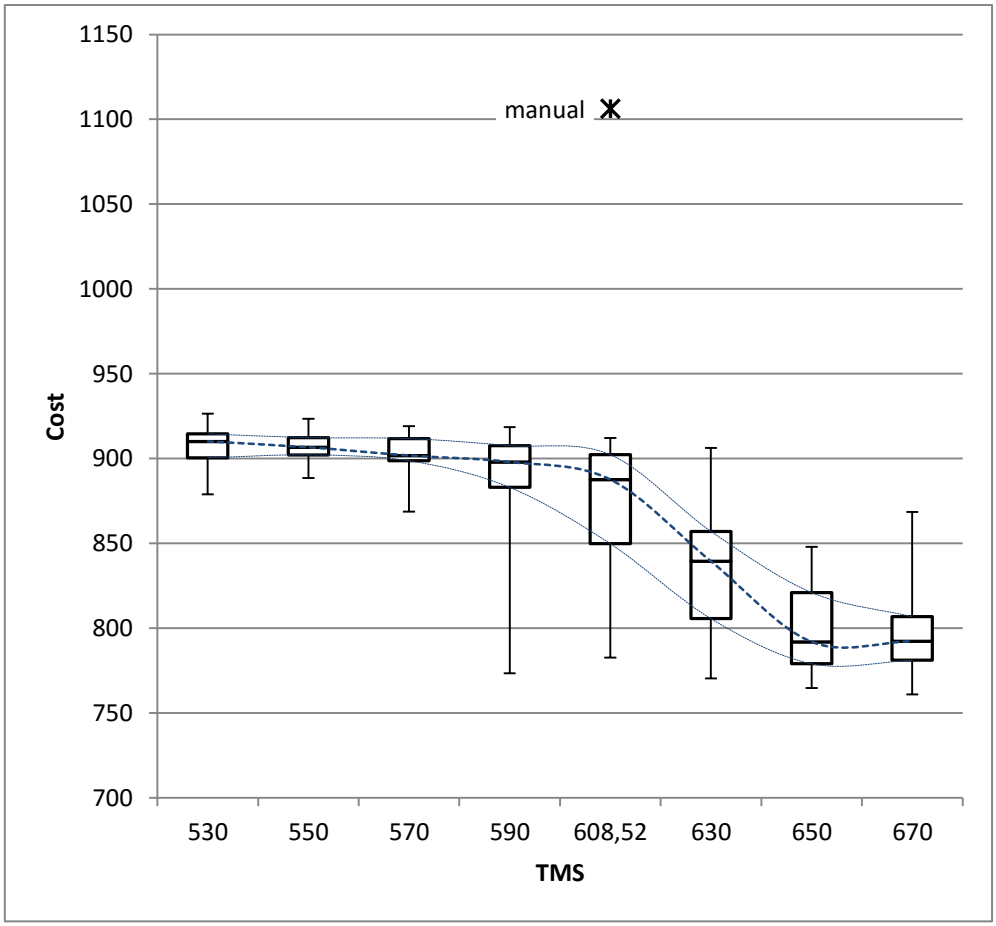
TSM	algorisme	manual
530	920,54	
550	902,06	
570	905,81	
590	910,38	
608,52	827,38	1106,28
630	875,43	
650	786,39	
670	792,35	
690	766,62	



Gràfic 11. Núvol de punts de la solució manual vs. La solució algorísmica

Amb un cost total de 827,38, l'algorisme proposa una solució un 25,21% millor que la humana amb el mateix temps de servei de 608 minuts. També s'aprecia que l'algorisme pot trobar una solució amb un temps de servei de 530, un 13% inferior a la humana, essent la solució un 17% més econòmica.

No obstant, es pot veure una correlació baixa entre el temps de servei i el cost, observació que ens porta a repetir la prova amb 30 execucions per a cada temps de servei. Resultats que es representen en el Gràfic 12



Gràfic 12. Box-plot de la solució algorísmica

En el box-plot es pot apreciar l'alta dispersió en la qualitat de les solucions, sobre tot en temps de servei entremitjos, que podríem dir que són els requerits en casos reals.

6.3 Instància de dimensió més alta

A continuació mostrem la resolució d'una instància en la que el nombre de clients és més alt, per tal de comprovar la potència de l'algorisme en casos més extrems que l'estudiat fins ara, tot i que no es correspon amb cap cas real. Excepte els forçats per restringir les solucions, tots els valors s'han calculat aleatòriament.

A continuació es detallen els paràmetres d'aquest cas.

Taula 9. Paràmetres del problema

Nombre de clients	120	Cap d'ells accepta el vehicle més gran. 29 d'ells es situen en un clúster separat.
Nombre de subclients	432	
Nombre de bases	4	Dues d'elles amb estoc inicial clarament insuficient
Nombre de magatzems	2	
Nombre de productes	4	A, B, C, C+
Nombre de tipus de vehicles	3	Que anomenarem *gran*, *mitjà* i *petit*
Cabal de càrrega	200 l/min	
Temps d'establiment de trajecte	5 min	
Temps de penalització intra-ciutat		Tots els nodes estan en diferents ciutats

TMS	600 min

Taula 10. Característiques dels tipus de vehicles

Tipus	Capacitat Compartiments (l)
Gran (5 compartiments)	10000, 8000, 8000, 4000, 2000
Mitjà (2 compartiments)	6000, 5000
Petit (2 compartiments)	2000, 2000

Taula 11. Característiques de les bases

Base	Vehicles	Estocs inicials	Estocs de seguretat
Base 1	7 petits, 7 mitjans, 1 gran	7000, 7000, 7000, 7000,	Cap estoc de seguretat
Base 2	7 petits, 7 mitjans, 1 gran	63504, 65862, 62178, 74586	
Base 3	7 petits, 5 mitjans, 1 gran	69210, 82320, 73266, 61920	
Base 4	7 petits, 7 mitjans	6000, 6000, 6000, 6000	

Els paràmetres de l'algorisme són, en aquest cas:

$$\beta_{MAP} = 0{,}5; t_{exec} = 60\ s$$

β_{CWS} l'ajustarem segons el cas per obtenir el mínim cost.

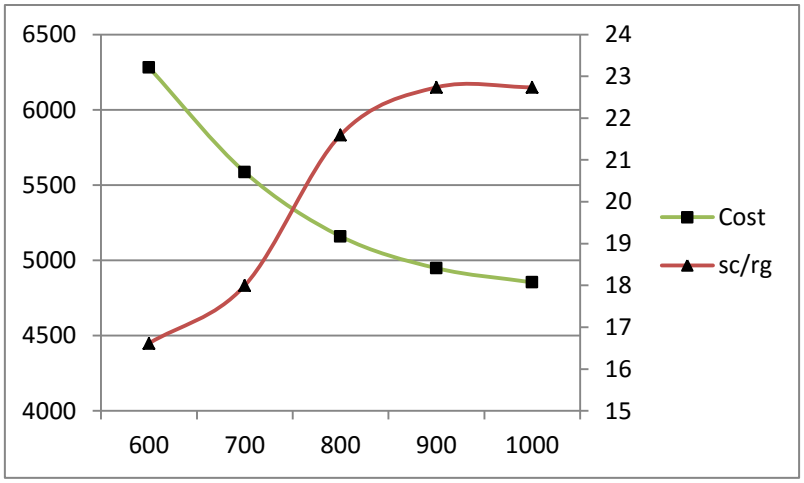
En la taula següent es resumeixen els resultats variant el TMS:

Taula 12. Resultats variant el TMS

TMS	r	rp	rg	cost	t	sol	r/rg	sc/r	sc/rg	β_{CWS}
600	30	7	26	6280,38	594,44	17	1,42	11,68	16,62	0,5
700	26	7	24	5585,25	676,31	565	1,37	13,09	18,00	0,5
800	26	6	20	5159,10	793,96	869	1,60	13,50	21,60	0,7
900	27	3	19	4947,82	878,11	921	1,58	14,40	22,74	0,7
1000	29	4	19	4853,54	999,76	842	1,74	13,09	22,74	0,85

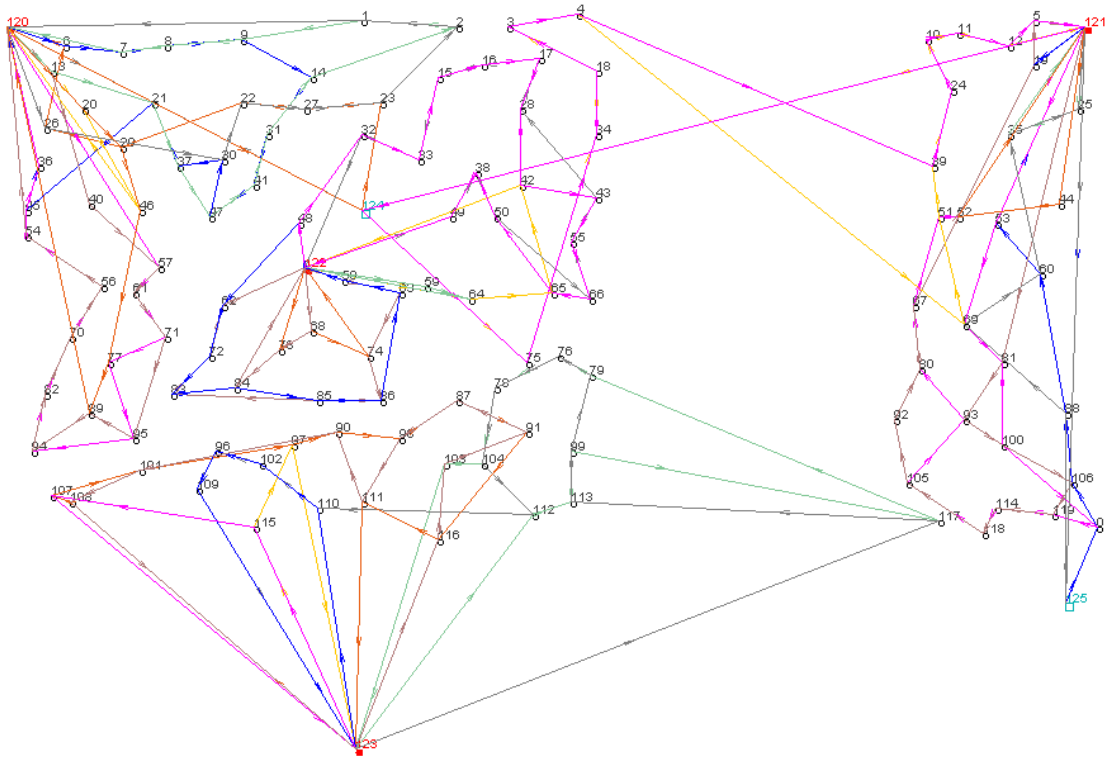
On r: nombre de rutes (no de provisió), rp: nombre de rutes de provisió, rg: nombre de grups de rutes, t: temps de servei, sol: nombre de solucions trobades, sc: nombre de subclients.

Aquest exemple (Gràfic 13) presenta una millor correlació entre TMS i cost que el cas anterior. També podem observar que el nombre de subclients (serveis d'un producte a un client) augmenta en funció del TMS. Això era d'esperar, atès que la disponibilitat de més temps de transport fa que les rutes puguin ser més llargues, estalviant costos. Ambdues gràfiques tenen una cota, entorn als 1000 min de TMS, a partir de la qual el resultat no millora.



Gràfic 13. Variació del cost i del nombre de subclients per grup de rutes

En el Gràfic 14 s'exposa la solució calculada per a TMS de 800 min. Els punts de color vermell són bases, els grisos són clients i els turquesa són magatzems.



Gràfic 14. Sortida gràfica per la solució de 800 min

En aquest gràfic s'observa, referent a les rutes de provisió, que el submapa que correspon a la base 120 en realitza 2, ambdues al magatzem 124 que és el que té més a prop. La primera és amb el vehicle gran i no reparteix cap entrega (cap client l'accepta) i la segona lliura als clients 23, 27, 22, ...

Per altra banda, la base 121 genera 4 rutes de provisió, dues al magatzem 125, que està més a prop, però les altres dues fan cap al magatzem 124 (en el gràfic no es distingeixen perquè

estan superposades). L'elecció del magatzem més llunyà li permet aprofitar els viatges tot servint els nodes 75, 34, 18, ... i aconseguint, d'aquesta manera, un estalvi en costs.

7. Conclusions i proposta de treballs futurs

En aquest projecte s'ha presentat un algorisme per resoldre un problema d'encaminament de vehicles, enriquit amb característiques com la limitació de temps total de servei, la possibilitat d'efectuar més d'una ruta per vehicle, l'addició d'aprovisionaments externs i l'ús de diferents tipus de vehicle, entre d'altres.

La complexitat de l'enunciat del problema no ha estat fruit de la imaginació d'una ment que dissenya un exercici teòric, sinó que es basa en les necessitats reals d'empreses del sector del transport de productes derivats del petroli. Hem explicat diferents formes d'afrontar aquests tipus de problemes i també hem vist clarament que el recorregut complet de tot l'espai de solucions, que seria la tàctica emprada pels mètodes exactes, no seria possible en un temps raonable ni tan sols per a un nombre mitjanament alt de nodes.

En comptes de recórrer un arbre, que és la forma habitual de l'espai de solucions, hem preferit utilitzar heurístiques per avaluar amb una certa probabilitat d'èxit quina branca cal seguir en totes i cadascunes de les bifurcacions arribant d'una manera directa (amb cost polinòmic) a la solució. Per tal de compensar la probabilitat d'arribar a una solució no prou bona, hem aplicat el mètode Multiarrencada per repetir el procés totes les vegades que sigui possible en un interval de temps preestablert amb petites variacions pseudoaleatòries, més o menys esbiaixades, en cada iteració.

A partir de dades proporcionades per una empresa del sector, hem calculat el cost i el temps d'un servei real. Comparant aquest resultat amb el proporcionat per l'algorisme hem vist com aquest calculava una solució que estalviaria el 25% del cost.

Ignorem el temps que ha dedicat la persona responsable per a arribar a la planificació que ha produït aquest resultat, però de ben segur que és molt superior als 40 s emprats per l'algorisme.

El resultat es considera prou bo però, analitzant altres execucions de la mateixa instància amb diferent variant TMS, s'han observat certes aberracions, com el fet de que existeixi algun resultat menys costós amb un TMS més baix. És a dir, el núvol de punts no és estrictament decreixent.

Vist això, hem decidit realitzar 30 execucions per a cada TMS i representar els resultats en un diagrama *scatter-box*, apreciant el següent:

- Es troben bones solucions per a un ample rang de temps de servei.
- Per sota d'un cert llindar de temps (entorn als 570 min), l'algorisme troba solucions similars, fins que arriba a un altre llindar a partir del qual no en troba cap. La solució més ràpida ha estat de 512,56 min amb un cost de 883,38 uc.
- La variabilitat del cost és més alta en temps intermedis, que possiblement seran els més usuals en el món real. Això vol dir que pot ser interessant per a l'usuari executar el càlcul més d'una vegada amb diferents llavors pseudoaleatòries.
- Les pitjors solucions ofertes per l'algorisme, per a qualsevol temps de servei que tingui solució són, com a mínim, un 17% millors que la manual.

- Per a temps de servei intermedis existeixen solucions molt bones (per exemple, 762 uc, un 31% millor que la manual) en el mateix temps de servei, però que es troben amb molt baixa probabilitat. Alguna d'aquestes solucions s'ha pogut aïllar (veure el tercer mapatge del Gràfic 10) i s'ha observat que alguna de les bases (en particular, la marcada amb un quadrat vermell) té una sèrie de clients lluny de la base, de fet, com totes les que són mitjanament bones. Com que l'heurística aplicada en el mapatge està exclusivament basada en criteris geogràfics, els mapatges (més) òptims són infreqüents i un resultat molt bo és molt poc probable.

En termes generals podem afirmar que la confecció de les rutes de forma manual ha de ser una tasca feixuga pel planificador i de resultats difícilment satisfactoris. El producte obtingut en aquest treball és una eina que pot proporcionar estalvis importants, tant en cost com en temps d'operació, a empreses del sector de la distribució de productes només transportables per separat. Però pensem que la seva utilitat també s'estén al camp de la simulació, com a assistent per a presa de decisions de caire econòmic-logístic com poden ser la d'adquirir més vehicles, augmentar els estocs mínims, obrir més bases, reubicar-les o plantejar obertures de mercat amb increment mínim d'infraestructura.

Com a línies de treball de futur podem esmentar:

- Millora de l'heurística del mapatge: es podria assajar amb una heurística per punts en la que el criteri geogràfic fos una part, però que també tingués un cert pes específic l'encaix del mix de productes amb l'estoc inicial de la base i amb les característiques de la flota. També seria interessant trobar una manera de fer intervenir la proximitat als magatzems en el procés de mapatge, tot i que existeix l'inconvenient de no saber si cal magatzem fins que no s'acaba el procés.
- Consideració de les despeses d'emmagatzemament a les bases: la compra al magatzem s'està calculant de forma que el vehicle s'ompli en la seva totalitat. Aquesta estratègia pot comportar altes despeses d'emmagatzematge a la base que no estan penalitzant perquè no formen part del càlcul del cost. Per altra banda, una compra molt ajustada al servei d'un dia provocaria una disponibilitat inicial molt baixa (o nul·la) pel dia següent, la qual cosa comportaria un sobrecost en el transport. Caldria avaluar si aquest aspecte justificaria una implementació multiperiode.
- Els estocs mínims moltes vegades es deuen a clàusules de disponibilitat assegurada en el contracte amb certs clients. Aleshores, cal que aquests clients estiguin permanentment assignats a la base que suporta aquesta disponibilitat. Això s'implementa fàcilment en la inicialització dels submapes.
- Possibilitar la demanda estocàstica dels clients, de manera que aquests puguin variar les comandes "in situ" quan arribi el camió de repartiment. La magnitud d'aquestes variacions no es coneix a priori, però es coneix la distribució de la seva probabilitat. En aquest escenari caldrà reservar, a cada vehicle que cobreixi una ruta, una àrea d'estoc de seguretat per a la demanda estocàstica.

Agraïments

Vull agrair als professors de la UOC Jésica de Armas i Ángel A. Juan pels seus consells i els ànims que m'han adreçat durant tot el desenvolupament del projecte i a aquest últim també per cedir-me les classes Java bàsiques. També a Enoc Martínez, del Dep. D'Informàtica, Multimèdia i Telecomunicació de la UOC, per la implementació de un primer esborrany del programa i a Roberto García Meizoso, de l'empresa Gasóleos San Marcos SL, per la informació tècnica i operativa referida al repartiment d'hidrocarburs.

Però el més especial dels agraïments ha de ser per a la meva família, que ha hagut d'aguantar el meu mal humor quan les coses no anaven prou bé i m'ha animat a continuar la feina en els moments més crítics. Gràcies, Marta, Gerard i Albert per la vostra paciència.

Glossari

CVRP:
Capacitated Vehicle Routing Problem (Problema d'Encaminament de Vehicles Capacitats)

CWS:
Clarke & Wright Savings heuristic (heurística d'estalvis Clarke & Wright)

HCG:
Heuristic Column Generation (Generació Heurística de Columnes)

HeCVRP:
Heterogeneous Fleet Capacitated Vehicle Routing Problem (Problema d'Encaminament de Vehicles Capacitats amb Flota Heterogènia)

HFFMCVRP:
Heterogeneous and Fixed Fleet, Multi-capacitated VRP (VRP Multicapacitat amb Flota Fixa Heterogènia)

HRTR:
Heterogeneous Record To Record (Camí Registre a Registre amb Flota Heterogènia)

ILS:
Iterated Local Search (Cerca Local Iterada)

LS:
Local Search (Cerca Local)

MCVRP:
Multi-Capacitated Vehicle Routing Problem (Problema d'Encaminament de Vehicles Multicapacitats)

MDVRP:
Multi-Depot Vehicle Routing Problem (Problema d'Encaminament de Vehicles Multibase)

RGTS:
Reactive Guided Tabu Search (Cerca Tabú Guiada Reactivament)

SAM:
Successive Approximations Method (Mètode d'Aproximacions Successives)

TMS:
Temps Màxim de Servei

TSP:
Travelling Salesman Problem (Problema del Viatjant de Comerç)

VRP:
Vehicle Routing Problem (Problema d'Encaminament de Vehicles)

Bibliografia

Arya, V., Garg, N., Khandekar, R., Mayerson, A., Munagala, K., & Pandit, V. (2004). Local search heuristics for k-median and facility location problems. *SIAM Journal on Computing, 33*(3), 544-562.

Avella, P., Boccia, M., & Sforza, A. (2004). Solving a fuel delivery problem by heuristic and exact approaches. *European Journal of Operational Research, 152*(1), 170-179.

Biggs, N., Lloyd, E., & Wilson, R. (1986). *Graph Theory 1736-1936.* Oxford: Clarendon Press.

Clarke, G., & Wright, J. (1964). Scheduling of vehicles from a central depot to a number of delivering points. *Operations Research*(12), 568–581.

Cockburn, A. (Maig / 2008). Using Both Incremental and Iterative Development. *STSC CrossTalk (USAF Software Technology Support Center), 21*(5), 27-30.

Cornillier, F., Boctor, F., Laporte, G., & Renaud, J. (2008). A heuristic for the multi-period petrol station replenishment problem. *European Journal of Operational Research, 191*(2), 295–305.

Croes, G. (1958). A method for solving traveling salesman problems. *Operations Research, 6*, 791-812.

Dantzig, G., & Ramser, J. (1959). The Truck Dispatching Problem. *Management Science, 6*(1), 80-91.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers and Operation Research, 5*(13), 533–549.

György, A., & Kocsis, L. (07 / 2011). Efficient Multi-Start Strategies for Local Search Algorithms. *Journal of Artificial Intelligence Research, 41*, 407-444. Consultat el 7 / 12 / 2015, a https://www.jair.org/media/3313/live-3313-5749-jair.pdf

Henke, T., Grazia Speranza, M., & Wäscher, G. (2014). The Multi-Compartment Vehicle Routing Problem with Flexible Compartment Sizes. *FEMM Working Papers*(140006).

Johnson, D., & McGeoch, L. (1997). The Traveling Salesman Problem: A Case Study in Local Optimization. A E. Aarts, & J. Lenstra, *Local Search in Combinatorial Optimization* (p. 215-310).

Juan, Á. A., Faulin, J., Cáceres-Cruz, J., Martínez, E., & Barrios, B. B. (2014). A successive approximations method for the heterogeneous vehicle routing problem: analysing different fleet configurations. *European J. Industrial Engineering, 8*(6), 762-788.

Juan, A. A., Pascual, I., Guimarans, D., & Barrios, B. (2014). Combining biased randomization with iterated local search for solving the multidepot vehicle routing problem. *International Transactions in Operational Research, 22*(4), 647-667.

Juan, A., Faulin, F., Jorba, J., Riera, D., Masip, D., & Barrios, B. (2011). On the Use of Monte Carlo Simulation, Cache and Splitting Techniques to Improve the Clarke and Wright Savings Heuristics. *Journal of the Operational Research Society*(6), 1085-1097.

Juan, A., Faulin, J., Ruíz, R., Barrios, B., Gilbert, M., & Vilajosana, X. (2009). Using oriented random search to provide a set of alternative solutions to the capacitated vehicle routing problem. *Operations research and Cyber-Infrastructure.*, 331–346.

Li, F., Golden, B., & Wasil, E. (2007). A record-to-record travel algorithm for solving the heterogeneous fleet vehicle routing problem. *Computers and Operations Research, 34*(9), 2734–2742.

Lourenço, H., Martin, O., & Stützle, T. (2010). Iterated Local Search: Framework and Applications. A M. Gendreau, & J. Potvin, *International Series in Operations Research & Management Science. Handbook of Metaheuristics* (2nd Edition ed., Vol. 146, p. 363-397). Springer.

Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research, 31*(12), 1985–2002.

Taillard, E. (1999). A heuristic column generation method for the heterogeneous fleet VRP. *RAIRO Operations Research, 33*(1), 1-14.

Tillman, F., & Cain, T. (1972). An upperbound algorithm for the single and multiple terminal delivery problem. *Management Science, 18*, 664-682.

Toth, P., & Vigo, D. (2001). Branch-and-bound algorithms for the capacitated VRP. A P. Toth, & D. Vigo, *The Vehicle Routing Problem* (p. 29-52). Philadelphia: SIAM.

Wang, Q., Ji, Q., & Chiu, C. (2014). Optimal Routing for Heterogeneous Fixed Fleets of Multicompartment Vehicles. *Mathematical Problems in Engineering, 2014*, Article ID 847630, 11 pages.

Annexos

Annex 1. Solució cas real

```
New best found. Cost: 827.38, time length: 603.04 among 427 solutions in 32.16 s
Found 531 solutions in 40.387016 s


Solution [id=27005,
  subSolutions=[
    SubSol [id=160491, costs=366.04, timeLength=501.12,
      sMap=SubMap [id=54192, depot=D32,
        nodes=[
          C12,C9,C11,C4,C10,C8,C20,C7,C13,C30,C29,C2,C14,C5
        ],
        demand=[8236, 4282, 2524, 2500],
        deliveryNeeds=[
          [1000*(0)->C11],[522*(0)->C12],[1000*(3)->C10],[625*(1)->C9],[1676*(1)->C10],
          [633*(0)->C4],[229*(1)->C4],[732*(2)->C4],[1000*(3)->C4],[500*(2)->C13],
          [500*(3)->C29],[722*(1)->C2],[5167*(0)->C8],[275*(0)->C7],[400*(2)->C14],
          [1030*(1)->C20],[892*(2)->C30],[639*(0)->C5]
        ],
        routeClassif={
          VehType [id=2, capacity=[10000, 8000, 8000, 4000, 2000], whCapable=true]=[
            RouteGroup [routes=[
              WhRoute [id=3491122,
                edges=[
                  [D32->W35(30.7)],[W35->D32(30.7)]
                ],
                costs=61.46, length=61.46, demand=[0, 0, 0, 0],
                loadDistribution=[1, 3, 0, 0, 3],
                deliveries=[],
                startTime=0.00, timeLength=381.30, arrivalTime=381.30
                purchase=[12000, 10000, 0, 10000]
              ]]
            ]
          ],
          VehType [id=1, capacity=[5000, 6000], whCapable=true]=[
            RouteGroup [routes=[
              Route [id=3491070,
                edges=[
                  [D32->C11(1.0)],[C11->C4(1.0)],[C4->D32(1.0)]
                ],
                costs=3.00, length=3.00, demand=[1633, 229, 0, 0],
                loadDistribution=[0, 1],
                deliveries=[
                  [1000*(0)->C11],[229*(1)->C4],[633*(0)->C4]
                ],
                startTime=0.00, timeLength=68.62, arrivalTime=68.62
              ],
              Route [id=3491068,
                edges=[
                  [D32->C10(1.0)],[C10->C12(1.0)],[C12->C9(23.6)],[C9->C20(1.0)],
                  [C20->C8(15.2)],[C8->C7(8.8)],[C7->C2(31.4)],[C2->D32(27.0)]
                ],
                costs=109.02, length=109.02, demand=[5964, 4053, 0, 0],
                loadDistribution=[1, 0],
                deliveries=[
                  [1676*(1)->C10],[522*(0)->C12],[625*(1)->C9],[1030*(1)->C20],
                  [5167*(0)->C8],[275*(0)->C7],[722*(1)->C2]
                ],
                startTime=68.62, timeLength=294.84, arrivalTime=363.46
              ],
              Route [id=3491080,
                edges=[
                  [D32->C13(26.8)],[C13->C5(29.3)],[C5->D32(59.9)]
                ],
                costs=116.01, length=116.01, demand=[639, 0, 500, 0],
                loadDistribution=[0, 2],
                deliveries=[
                  [500*(2)->C13],[639*(0)->C5]
                ],
                startTime=363.46, timeLength=137.66, arrivalTime=501.12
              ]]
            ],
```

```
        RouteGroup [routes=[
          Route [id=3491073,
            edges=[
              [D32->C4(1.0)],[C4->C10(1.0)],[C10->D32(1.0)]
            ],
            costs=3.00, length=3.00, demand=[0, 0, 732, 2000],
            loadDistribution=[2, 3],
            deliveries=[
              [1000*(3)->C4],[732*(2)->C4],[1000*(3)->C10]
            ],
            startTime=0.00, timeLength=77.32, arrivalTime=77.32
          ]]
        ]
      ],
      VehType [id=0, capacity=[2000, 2000], whCapable=false]=[
        RouteGroup [routes=[
          Route [id=3491114,
            edges=[
              [D32->C14(22.4)],[C14->C29(12.4)],[C29->C30(15.2)],[C30->D32(23.6)]
            ],
            costs=73.55, length=73.55, demand=[0, 0, 1292, 500],
            loadDistribution=[2, 3],
            deliveries=[
              [400*(2)->C14],[500*(3)->C29],[892*(2)->C30]
            ],
            startTime=0.00, timeLength=145.33, arrivalTime=145.33
          ]]
        ]
      ]
    },
    avails=[
      Availability [
        whRoute.id= (initial inventory), t=0.00, qtys=[3764, 12718, 15476, 11500]
      ],
      Availability [
        whRoute.id=3491122, t=381.30, qtys=[15764, 22718, 15476, 21500]
      ]
    ]
  ]
],
SubSol [id=239973, costs=401.76, timeLength=603.04,
  sMap=SubMap [id=81013, depot=D33,
    nodes=[
      C15,C17,C19,C6,C16,C18,C22,C3,C0,C24
    ],
    demand=[4630, 1200, 6640, 0],
    deliveryNeeds=[
      [735*(2)->C15],[1200*(1)->C15],[790*(2)->C0],[1550*(0)->C15],[933*(2)->C24],
      [722*(2)->C16],[760*(2)->C22],[2080*(0)->C6],[1000*(2)->C17],[800*(2)->C19],
      [900*(2)->C18],[1000*(0)->C3]
    ],
    routeClassif={,
      VehType [id=1, capacity=[5000, 6000], whCapable=true]=[
        RouteGroup [routes=[
          WhRoute [id=5207703,
            edges=[
              [D33->W35(82.0)],[W35->C6(74.3)],[C6->C19(17.7)],[C19->C22(75.3)],
              [C22->C24(5.1)],[C24->C3(4.6)],[C3->C0(13.1)],[C0->C18(69.0)],
              [C18->C16(51.1)],[C16->C17(1.0)],[C17->C15(5.6)],[C15->D33(1.0)]
            ],
            costs=399.76, length=399.76, demand=[4630, 0, 5905, 0],
            loadDistribution=[0, 2],
            deliveries=[
              [2080*(0)->C6],[800*(2)->C19],[760*(2)->C22],[933*(2)->C24],
              [1000*(0)->C3],[790*(2)->C0],[900*(2)->C18],[722*(2)->C16],
              [1000*(2)->C17],[1550*(0)->C15]
            ],
            startTime=0.00, timeLength=548.69, arrivalTime=548.69
            purchase=[5000, 0, 6000, 0]
          ],
          Route [id=5207705,
            edges=[
              [D33->C15(1.0)],[C15->D33(1.0)]
            ],
            costs=2.00, length=2.00, demand=[0, 1200, 735, 0],
            loadDistribution=[1, 2],
            deliveries=[
```

```
                [1200*(1)->C15],[735*(2)->C15]
              ],
              startTime=548.69, timeLength=54.35, arrivalTime=603.04
            ]]
          ]
        ]
      },
      avails=[
        Availability [
          whRoute.id= (initial inventory), t=0.00, qtys=[0, 6800, 265, 0]
        ],
        Availability [
          whRoute.id=5207703, t=548.69, qtys=[370, 6800, 360, 0]
        ]
      ]
    ]
  ],
  SubSol [id=66062, costs=59.58, timeLength=169.94,
    sMap=SubMap [id=22322, depot=D34,
      nodes=[
        C26,C1,C31,C25,C21,C23,C27,C28
      ],
      demand=[0, 1617, 6378, 0],
      deliveryNeeds=[
        [1800*(2)->C28],[1500*(2)->C21],[437*(2)->C25],[500*(2)->C31],[792*(2)->C26],
        [1617*(1)->C1],[1000*(2)->C23],[349*(2)->C27]
      ],
      routeClassif={
        VehType [id=1, capacity=[5000, 6000], whCapable=true]=[
          RouteGroup [routes=[
            Route [id=1445438,
              edges=[
                [D34->C23(8.4)],[C23->C27(10.5)],[C27->C21(1.0)],[C21->C25(5.1)],
                [C25->C26(1.0)],[C26->C28(3.4)],[C28->D34(4.0)]
              ],
              costs=33.45, length=33.45, demand=[0, 0, 5878, 0],
              loadDistribution=[2, 2],
              deliveries=[
                [1000*(2)->C23],[349*(2)->C27],[1500*(2)->C21],[437*(2)->C25],
                [792*(2)->C26],[1800*(2)->C28]
              ],
              startTime=0.00, timeLength=169.94, arrivalTime=169.94
            ]]
          ],
          RouteGroup [routes=[
            Route [id=1445439,
              edges=[
                [D34->C1(11.6)],[C1->C31(2.3)],[C31->D34(12.3)]
              ],
              costs=26.13, length=26.13, demand=[0, 1617, 500, 0],
              loadDistribution=[1, 2],
              deliveries=[
                [1617*(1)->C1],[500*(2)->C31]
              ],
              startTime=0.00, timeLength=78.53, arrivalTime=78.53
            ]]
          ]
        ]
      },
      avails=[
        Availability [
          whRoute.id= (initial inventory), t=0.00, qtys=[25000, 5383, 15622, 0]
        ]
      ]
    ]
  ]],
  cost=827.38,
  timeLength=603.04
]
```