Pràctica 3 - Planificació

Pol Mañé, Carles Orriols, Pau Vilaró, Neus Mayol

10 de gener de 2023

Índex

1 Introducció

Les computadores són màquines impressionants, és admirable tota la quantitat de coses que poden arribar a fer, i les que encara queden per idear! Podem calcular, aprendre, comunicar-nos, guardar-hi coses, treballar-hi. Encara així, les habilitats d'un ordinador només tenen sentit quan les fusionem amb l'enginy i el coneixement humà. De fet, els mateixos ordinadors són producte del coneixement humà. I és que els humans interpretem tot allò que ens mostren les màquines i en traiem les conclusions pertinents.

Aquesta pràctica n'és un exemple clar, perquè nosaltres definirem una sèrie d'objectes en un domini, els quals una màquina no pot relacionar, en canvi nosaltres sí que ho podem fer i utilitzarem la rapidesa de càlcul d'un ordinador per a poder obtenir resultats ràpids. D'aquesta manera podrem agafar els resultats i extreure'n les nostres pròpies conclusions.
En aquest cas treballarem la planificació, una vessant de la intel·ligència artificial que podem fer servir per crear eines que organitzin un seguit de procediments. Per fer-ho partirem d'un estat inicial que evolucionarà a partir dels processos que nosaltres mateixos programarem i arribarà a un estat final, el nostre objectiu.

2 Plantejament del problema

Se'ns planteja una tasca d'organitzar les comunicacions entre les diferents bases del planeta Mart. Les comunicacions es fan a través de vehicles *rover*, que utilitzarem per moure els recursos d'una base a una altra. Sempre existeix un camí entre dues bases, però aquest no té per què ser directe: Potser haurem de passar per una o més bases abans d'arribar a la que ens interessa. A partir d'aquí el problema pressenta varies extensions:

- Primera extensió: Els *rovers* podran transportar també fins a dues persones. Quan els *rovers* transportin persones no podran transportar cap altre tipus de materials.
- Segona extensió: S'afegeix a la primera extensió el fet de que els vehicles tenen un diposit de combustible que es decrementa en una unitat cada vegada que fem un desplaçament entre una base i una altra. També buscarem minimitzar el consum de benzina.
- Tercera extensió: S'afegeix a la segona extensió la capacitat de poder assignar una prioritat a cada tipus de petició. A part, també assignarem pesos a cadascun dels aspectes del problema per tal de poder veure com podem prioritzar la minimització de l'ús de combustible i les prioritats d'aquesta extensió.

2.1 Primer punt extra

Per tal de provar la nostra solució necessitem pensar mapes, distribucions de recursos. Per això intentarem fer un programa *Python* que ens permeti automatitzar el procés.

2.2 Segon punt extra

El segon punt extra és una extensió del primer on el generador crearà jocs de proves de dificultat incremental.

Els dos punts extra s'han fet a la vegada i es troben al fitxer *punt-extra-1.py*.

3 Metodologia

Per començar es va llegir atentament tot el problema i en vam identificar-ne les seves característiques. Vam començar definint els elements que incorporava aquest, els objectes que necessitàvem representar al nostre programa. Després vam identificar com es relacionava cada element del problema, i quines possibles situacions en podrien donar en aquests. Ja per acabar, vam pensar en quins casos podem passar d'una situació a una altra: què és necessari per que es doni la situació i quins efectes comportarà.

En acabat vam utilitzar els jocs de proves per comprovar que el que havíem programat funcionés adequadament. Aquest procés el vam repetir quatre vegades: una pel cas base i per cadascuna de les extensions del problema.

4 Descripció del domini

4.1 Variables

Aquests són els elements que vam creure adients per representar el nostre problema, ja que són clau per representar un estat i els seus canvis. Finalment també vam haver d'afegir més elements per poder resoldre algunes de les extensions.

`rover - object` Representa un vehicle que transporta recursos entre bases.

`base - object` Un assentament des d'on podem carregar i descarregar recursos, estacionar-hi *rovers* i fer peticions. Una base és també el node del graf que representa el terreny.

`recursos - object` Representa un element que es troba emmagatzemat a una base, o bé el podem transportar mitjançant un *rover*.

`asentamiento - base` Aquesta subclasse de la classe base serveix per indicar que la base en qüestió és un assentament o, en altres paraules, que és una base on hi conviuen persones. Es troba a partir de la primera extensió.

`almacen - base` Aquesta subclasse de la classe base inclou tot aquelles bases que siguin magatzems i que per tant només guardin materials. Es troba a partir de la primera extensió.

`persona - recursos` Correspon a un esser humà que es trobarà a una base o bé serà transportat. Es defineix com una subclasse de l'objecte recurs i es fa servir a partir de la primera extensió.

`suministro - recursos` Correspon a un material que es trobarà a una base o bé serà transportat. Es defineix com una subclasse de l'objecte recurs i es fa servir a partir de la primera extensió.

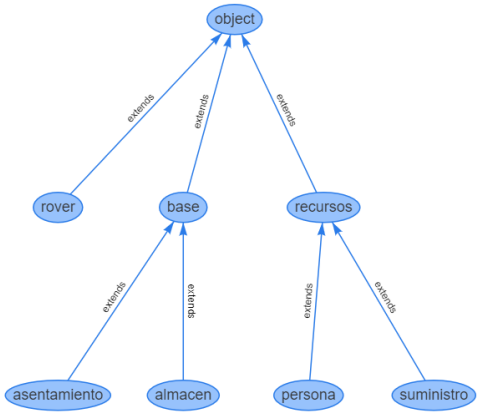

Figura 1: Jerarquia d'objectes del domini

4.2 Predicats

Utilitzarem els predicats per definir l'estat dels elements del problema. Aquests ens ajudaran a relacionar les diferents variables que hem explicat a l'apartat anterior.

(estacionado ?r - rover ?b - base) Relaciona un *rover* amb una base. Serà cert si el vehicle en qüestió es troba estacionat a la base introduïda.

(enBase ?rec - recursos ?b - base) Relaciona un recurs i una base del programa. Serà cert quan la base disposi del recurs en qüestió.

(enRover ?r - rover ?rec - recursos) Es pot donar el cas que el recurs no es trobi a cap base, però sí que el puguem trobar dins un *rover*. Aquest predicat serà cert en aquests casos. Relaciona un recurs amb un *rover*.

(peticion ?rec - recursos ?as - asentamiento) Quan un assentament necessita un recurs farà una petició. Aquest predicat relacionarà l'assentament amb el recurs que necessita.

(adyacente ?b1 - base ?b2 - base) Sabem que existeix un camí entre dues bases, però que no sembre serà un camí directe. Aquest predicat relaciona dues bases i serà cert quan el camí entre elles es directe i per tant, no passa per cap altra base.

(servido ?rec - recurso) Cert quan s'ha abastit la necessitat d'un recurs.

4.3 Funcions

En farem servir a partir de la primera extensió i aquestes s'aniran acumulant en les futures extensions o inclús modificant per certs casos específics on es necessiti per un canvi d'estratègia.

[Extensió 1]

(cantidad-personas ?r - rover) Indica la quantitat de persones que té el *rover* en qüestió. Aquest nombre mai podrà ser superior a dos.

(cantidad-suministros ?r - rover) Indica el nombre de subministraments que té el *rover*, mai podrà haver-hi més d'un subministrament.

[Extensió 2]

(combustible-gastado ?r - rover) Guarda el combustible gastat per el *rover* en qüestió.

(capacidad-combustible) Quantitat de combustible gastat per el *rover* en qüestió.

(suma-combustible-gastado) Quantitat de combustible gastat per tots els *rovers* del sistema.

[Extensió 3]

(suma-combustible-no-gastado) Es correspon al combustible no gastat respecte la capacitat total de tots els *rovers* del sistema. S'intercanvia la funció *suma-combustible-gastado* de l'extensió anterior per aquesta, ja que ens farà falta realitzar aquest canvi de cara a l'estratègia de la mètrica a utilitzar.

(prioridad-peticion ?r - recursos ?b - base) Prioridat d'una petició per portar un recurs a una base.

`(suma-prioridades)` Sumatori total de totes les prioritats de les peticions que s'han portat a terme.

4.4 Accions

`cargar_persona` Aquesta acció serveix per poder carregar una persona a un *rover*. Els paràmetres d'entrada seran els identificadors d'un *rover*, una persona i un assentament. Per poder dur a terme l'acció primer haurem de comprovar que la persona i el *rover* es trobin estacionats a l'assentament i que la persona estigui disponible com a recurs. Si es compleixen aquestes condicions es procedirà a carregar la persona al *rover* i a indicar que aquella persona ja no està disponible.

[Extensió 1] S'afegeix la possibilitat de carregar persones, i per tant l'acció tindrà en compte que al carregar la persona no hi hagi cap subministrament carregat i que la quantitat de persones al *rover* mai sigui superior a dos. En cas que compleixi les condicions s'incrementarà en una el nombre de persones al *rover*.

[Extensió 2 i 3] No es modifica l'acció respecte l'extensió anterior.

`descargar_persona` Aquesta acció s'encarrega de descarregar una persona d'un *rover* a un assentament. Els paràmetres d'entrada seran els identificadors d'un *rover*, una persona i un assentament. Per poder dur a terme l'acció primer haurem de comprovar que la persona es trobi al *rover* en qüestió, que el *rover* estigui estacionat a l'assentament i que l'assentament hagi fet una petició de la persona carregada al *rover*. Si es compleixen aquestes condicions es procedirà a descarregar la persona a l'assentament, a indicar que la petició s'ha complert i la persona ja no està disponible per servir.

[Extensió 1] S'afegeix la possibilitat de descarregar persones, per tant ara l'acció decrementarà el nombre total de persones al *rover*.

[Extensió 2] No es modifica l'acció respecte l'extensió anterior.

[Extensió 3] Es suma la prioritat realitzada de la petició feta del personal entregat.

`cargar_suministro` Funciona igual que `cargar_persona` però ara només funcionarà en subministraments i quan les bases siguin de tipus magatzem.

[Extensió 1] Es comprova que no hi hagi cap persona ni subministrament al *rover*. En cas de ser així, es carrega el subministrament.

[Extensió 2 i 3] No es modifica l'acció respecte l'extesió anterior.

`descargar_suministro` Funciona igual que `descargar_persona` però ara només funcionarà en subministraments i quan les bases siguin de tipus magatzem.

[Extensió 1] Decrementa la quantitat de subministres que hi ha al *rover*.

[Extensió 2] No hi ha canvis respecte l'extensió anterior.

[Extensió 3] Es suma la prioritat realitzada de la petició feta del subministrament.

`mover_rover` Els paràmetres d'entrada seran dos bases (una d'origen i una de destí) i un *rover*. En cas que les bases siguin adjacents i el *rover* es trobi estacionat a la base d'origen, es procedirà a desplaçar-lo entre ambdues bases.

[Extensió 1] No es modifica l'acció.

[Extensió 2] Afegirem una variable que ens digui si es pot realitzar el desplaçament quan el nivell de combustible ens ho permeti. En cas que el puguem executar incrementarà el combustible total gastat del programa.

[Extensió 3] L'únic canvi que es fa respecte l'extensió-2 és que en comptes d'incrementar el combustible gastat, el que es fa és decrementar el que no s'ha gastat. Això es fa per poder realitzar la mètrica de forma correcta al maximitzar aquest valor.

se_queda Com a paràmetres d'entrada tindrem un *recurs* i un *assentament*. Si existeix una petició en aquell assentament i el recurs pertany a aquest propi assentament, llavors es pren l'acció de que el recurs no es desplaci i es quedi per servir aquesta petició. Actualitzarem l'estat d'aquest recurs com a servit i esborrarem la petició d'aquest recurs en aquesta base.

La raó per la qual utilitzar aquesta acció es deu a que passi el cas en que generem un fitxer problema que doni aquesta casuística, que un assentament fa una petició d'un recurs que ja està en aquella base. De fet, aquesta acció només es realitza sobre recursos de subtipus *persona*, ja que son els únics que inicialment poden estar en un assentament.
Ja per acabar, gràcies a aquesta operació evitem que un *rover* que ha arribat a l'*assentament*, perdi temps carregant i descarregant el recurs per marcar-lo com a *servido*, cosa que passaria sense aquest operador.

[Extensions] Aquesta acció és manté en totes les extensions i no pateix cap canvi.

5 Modelatge del problema

En aquest apartat comentarem com hem dissenyat els fitxers de problema.

5.1 Objectes

En aquest apartat és on declarem els objectes que formaran part del problema en concret. Aquí també hem d'especificar el *tipus* de cada objecte si és que hem fet servir *:typing* en els requeriments del domini. Declarem quants i quins objectes tindrà el nostre problema, en el nostre cas: quantes *bases* (assentaments i magatzems) tenim, de quants i de quin tipus de *recursos* disposem i, en aquest cas, també el nombre de *rovers* que utilitzarem.

Com hem dit, aquí decidim la quantitat d'objectes que intervindran en el nostre problema; com el seu propi nom indica, cada un d'aquests serà un "objecteï per tant estarà identificat i podem tenir informació sobre ell. El planificador buscarà una solució que compleixi el problema plantejat, i es fixarà en l'estat i possibles canvis que pot realitzar sobre cada objecte, i mitjançant les accions permeses intentara arribar a un estat final.

La necessitat de diferenciar objectes és clara. Cada objecte té una finalitat i el seu conjunt d'accions: un *rover* pot transportar coses, les *bases* són localitzacions i estableixen camins i els *recursos* doncs són els que generen la necessitat d'aquesta mobilització. Com ja sabem, l'ús dels "tipus"es important per evitar que el planificador intenti executar les accions sobre cada objecte "genèric"de la base de dades... estem ajudant a la eficiència del programa.

5.2 Estat inicial

L'estat inicial conté diversos elements. És l'apartat on establim la informació de la base de dades del planificador, és on inicialitzem els predicats i on declarem, òbviament, l'estat inicial del problema. És a dir, des d'aquest apartat decidim des d'on parteix el nostre problema, en quin estat volem que comenci.

L'estructura de l'estat inicial dels nostres fitxers de problema es la següent:

- Comencem amb un apartat d'inicialitzacions de les funcions del nostre problema (inicialització de fluents). Inicialitzem, en el casos necessaris, la quantitat de combustible de que disposaran els *rovers* i la suma total d'aquest; també inicialitzem els estats dels *rovers*, quantes persones o subministraments hi ha des d'un inici en cada *rover* (inicialment estan buits) o de quanta gasolina disposen (inicialment plens).

- Seguim amb els predicats per situar en quina localització, *base*, estan els *rovers*. Amb el predicat, *estacionado*, és com ho adjudiquem.

- A continuació, realitzem les connexions del nostre mapa. Aquí és on declarem els predicats que permetran al nostre sistema identificar camins entre les *bases*. Mitjançant el predicat *adyacente* identifiquem quines bases estan adjacents entre elles, referint-nos a adjacents com a bases que tenen un camí que les connecta. Nosaltres hem considerat, que les connexions entre localitzacions són bidireccionals, és per això que trobem codificat les adjacències des d'una *base u* a una altra *v* de la següent manera: *adyacente(u, v) adyacente(v, u)*

- Per acabar, considerem el bloc de peticions (ja sigui el de peticions de personal, com el de subministraments). En aquest apartat, és on inicialitzem els predicats que relacionen un *recurs* amb la *base* on se'l necessita (gràcies al predicat *peticion*. A més a més, també indiquem en quina base es troba el recurs (predicat *enBase*) i per les extensions que faci falta, indiquem quina es la prioritat de cada petició amb el fluent *prioridad-peticion*.

Tenint en compte que ens interessa avaluar les diferents decisions que pren el planificador, hem optat per deixar un mapa estàtic (sempre el mateix) en els jocs de prova. Així podíem comparar

les diferents solucions obtingudes pel planificador d'una manera més senzilla. Tot i així, en el generador de problemes aleatoris, per posar a prova el planificador, vam decidir canviar el mapa per un amb més bases, però l'estructura que segueix el fitxer és la mateixa.

5.3 Estat final

Com s'ha mencionat anteriorment la vida a mart és complicada i no sempre serà possible atendre a totes les peticions existents ja que falten recursos. És per això que l'estat final ha de ser aquell que permeti arribar a realitzar l'entrega de tots aquells recursos que estan disponibles per tal de poder atendre les màximes necessitats possibles.

En el cas base i primera extensió només era necessari arribar a resoldre la màxima quantitat de peticions segons els recursos disponibles, però a mesura que anava avançant el treball vam necessitar optimitzar l'estat final perquè consideri altres aspectes de cara a arribar a una solució. Aquestes optimitzacions s'utilitzaven per arribar a estats finals amb el mínim combustible gastat dels *rovers*, atendre aquelles peticions de més prioritat o la combinació de les dues.

6 Desenvolupament dels models

Per al correcte desenvolupament d'aquesta pràctica hem optat pel tipus de disseny basat en prototipatge ràpid i disseny incremental, proposat en les anteriors pràctiques i en aquesta.

Hem començat resolent l'essència del problema, un nivell bàsic. En aquesta pràctica, al funcionar amb extensions, és normal que aquest nivell bàsic sigui l'arrel de tot, és on sorgeixen la majoria de predicats i accions que ens permetran resoldre el problema en si. Després amb les extensions anem modelant el problema i suposant situacions més realistes, o més elaborades.

Per dur a terme el disseny incremental, hem començat pel nivell bàsic i amb els jocs de prova hem comprovat el seu correcte funcionament. Si trobàvem un cas que no funcionava o comportaments inadequats en la planificació, revisàvem el codi i corregíem l'error. Per poder passar a la següent extensió s'ha d'intentar arrossegar el menor nombre d'errors possibles.

Ja que les extensions són acumulatives, una extensió superior requereix de l'extensió inferior, és per això que a l'hora de començar una nova extensió no partim del codi de nivell bàsic. Prenem de punt de partida el codi que necessitem i comencem a realitzar canvis per tal de complir les noves funcionalitats; això pot incloure afegir noves funcions (*:fluents*) o accions o modificar coses ja existents. En la majoria d'extensions hem hagut d'afegir noves funcions per tenir un control sobre quantitats que ens posaven un limit en la resolució del problema, i també, hem hagut de modificar les accions per adequar el correcte funcionament d'aquestes funcions.

És mitjançant aquest mètode com hem arribat a la darrera extensió del problema, a la última versió (Extensió 3) d'aquest programa de planificació. Pel que hem comentat, aquesta darrera versió inclou les anteriors. Tot i així, per tal d'assegurar un correcte funcionament del disseny incremental hem realitzat jocs de prova en cada extensió, ens disposem a comentar-los en la següent secció.

7 Jocs de prova

En aquest apartat hem fet un seguit de jocs de prova per veure com es comporta el nostre planificador. Per cada extensió hem fet un parell de problemes en PDDL i a l'última extensió hem fet servir de manuals, però també hem fet un petit estudi de com es comporta en augmentar diferents paràmetres usant el generador de problemes fet en *Python*.

7.1 Jocs de prova versió bàsica

En aquesta versió bàsica hem fet dos jocs de prova. Són casos molt semblants, però amb l'única diferència que *prob_basic.pddl* té menys recursos que *prob_basic_2.pddl*, conservant el nombre de peticions. A continuació podem veure les dues planificacions que fa, en primer lloc la de *prob_basic.pddl*:

```
step    0: MOVER_ROVER R01 AL05 AS03
        1: CARGAR_PERSONA R01 AS03 P01
        2: CARGAR_PERSONA R01 AS03 P02
        3: MOVER_ROVER R01 AS03 AL04
        4: CARGAR_SUMINISTRO R01 AL04 S01
        5: MOVER_ROVER R01 AL04 AS04
        6: CARGAR_PERSONA R01 AS04 P03
        7: DESCARGAR_PERSONA R01 AS04 P01
        8: MOVER_ROVER R01 AS04 AL05
        9: MOVER_ROVER R01 AL05 AS03
       10: DESCARGAR_PERSONA R01 AS03 P03
       11: MOVER_ROVER R02 AS02 AL03
       12: CARGAR_SUMINISTRO R02 AL03 S02
       13: MOVER_ROVER R02 AL03 AS02
       14: DESCARGAR_SUMINISTRO R02 AS02 S02
       15: MOVER_ROVER R01 AS03 AL05
       16: MOVER_ROVER R01 AL05 AS01
       17: DESCARGAR_PERSONA R01 AS01 P02
       18: DESCARGAR_SUMINISTRO R01 AS01 S01
       19: MOVER_ROVER R01 AS01 AL01
       20: CARGAR_SUMINISTRO R01 AL01 S03
       21: MOVER_ROVER R01 AL01 AS04
       22: MOVER_ROVER R01 AS04 AL05
       23: MOVER_ROVER R01 AL05 AS03
       24: DESCARGAR_SUMINISTRO R01 AS03 S03
```

Seguidament, veiem la planificació de prob_basic_2.pddl:

```
step    0: CARGAR_PERSONA R01 AS01 P03
        1: MOVER_ROVER R01 AS01 AL05
        2: MOVER_ROVER R02 AS04 AL05
        3: MOVER_ROVER R03 AS02 AL05
        4: MOVER_ROVER R01 AL05 AS03
        5: DESCARGAR_PERSONA R01 AS03 P03
        6: MOVER_ROVER R02 AL05 AS02
        7: CARGAR_PERSONA R02 AS02 P02
        8: MOVER_ROVER R02 AS02 AL05
        9: MOVER_ROVER R03 AL05 AS02
       10: MOVER_ROVER R01 AS03 AL03
       11: MOVER_ROVER R02 AL05 AS04
       12: CARGAR_PERSONA R02 AS04 P01
       13: MOVER_ROVER R02 AS04 AL05
```

```
14: MOVER_ROVER R01 AL03 AS03
15: CARGAR_PERSONA R01 AS03 P04
16: MOVER_ROVER R01 AS03 AL04
17: MOVER_ROVER R02 AL05 AS04
18: MOVER_ROVER R01 AL04 AS04
19: DESCARGAR_PERSONA R01 AS04 P04
20: MOVER_ROVER R01 AS04 AL04
21: MOVER_ROVER R02 AS04 AL01
22: CARGAR_SUMINISTRO R02 AL01 S01
23: CARGAR_SUMINISTRO R02 AL01 S02
24: CARGAR_SUMINISTRO R02 AL01 S03
25: CARGAR_SUMINISTRO R02 AL01 S04
26: MOVER_ROVER R02 AL01 AS04
27: DESCARGAR_SUMINISTRO R02 AS04 S04
28: MOVER_ROVER R02 AS04 AL05
29: MOVER_ROVER R02 AL05 AS03
30: DESCARGAR_SUMINISTRO R02 AS03 S03
31: MOVER_ROVER R02 AS03 AL05
32: MOVER_ROVER R02 AL05 AS02
33: DESCARGAR_SUMINISTRO R02 AS02 S02
34: MOVER_ROVER R02 AS02 AL05
35: MOVER_ROVER R02 AL05 AS01
36: DESCARGAR_PERSONA R02 AS01 P01
37: DESCARGAR_PERSONA R02 AS01 P02
38: DESCARGAR_SUMINISTRO R02 AS01 S01
```

Es pot apreciar que augmenta considerablement el nombre de moviments entre bases, de 13 a 23, que fan els *rovers*. Això ve donat perquè tenir més recursos obliga als *rovers* a passar per més bases i té més limitacions en on deixar els recursos, ja que les peticions són úniques. A diferència de l'altre cas que hi ha recursos que són demanats des de més d'una base i, per tant, pot escollir quina de les peticions satisfà.

7.2 Jocs de prova extensió 1

En aquesta extensió tenim límit de capacitat per cada *rover* operatiu. Per veure com evoluciona respecte al cas anterior hem fet servir els mateixos dos casos de prova. En aquest cas veurem les planificacions mencionades, generades pels problemes escrits en els fitxers *prob_ext1.pddl* i *prob_ext1_2.pddl*, respectivament:

```
step    0: MOVER_ROVER R01 AS01 AL02
        1: MOVER_ROVER R01 AL02 AS02
        2: MOVER_ROVER R01 AS02 AL03
        3: CARGAR_SUMINISTRO R01 AL03 S02
        4: MOVER_ROVER R01 AL03 AS02
        5: DESCARGAR_SUMINISTRO R01 AS02 S02
        6: MOVER_ROVER R02 AS04 AL04
        7: MOVER_ROVER R01 AS02 AL05
        8: CARGAR_SUMINISTRO R02 AL04 S01
        9: MOVER_ROVER R02 AL04 AS04
       10: MOVER_ROVER R02 AS04 AL01
       11: MOVER_ROVER R01 AL05 AS01
       12: MOVER_ROVER R02 AL01 AS01
       13: DESCARGAR_SUMINISTRO R02 AS01 S01
       14: MOVER_ROVER R01 AS01 AL02
```

```
15: MOVER_ROVER R02 AS01 AL05
16: MOVER_ROVER R02 AL05 AS04
17: MOVER_ROVER R01 AL02 AS02
18: CARGAR_PERSONA R02 AS04 P03
19: MOVER_ROVER R02 AS04 AL05
20: MOVER_ROVER R01 AS02 AL05
21: MOVER_ROVER R01 AL05 AS03
22: CARGAR_PERSONA R01 AS03 P01
23: MOVER_ROVER R02 AL05 AS03
24: DESCARGAR_PERSONA R02 AS03 P03
25: CARGAR_PERSONA R02 AS03 P02
26: MOVER_ROVER R02 AS03 AL05
27: MOVER_ROVER R01 AS03 AL04
28: MOVER_ROVER R02 AL05 AS01
29: DESCARGAR_PERSONA R02 AS01 P02
30: MOVER_ROVER R01 AL04 AS04
31: DESCARGAR_PERSONA R01 AS04 P01
32: MOVER_ROVER R01 AS04 AL01
33: CARGAR_SUMINISTRO R01 AL01 S03
34: MOVER_ROVER R01 AL01 AS04
35: MOVER_ROVER R01 AS04 AL05
36: MOVER_ROVER R01 AL05 AS03
37: DESCARGAR_SUMINISTRO R01 AS03 S03
```

i el segon cas:

```
step    0: MOVER_ROVER R01 AS01 AL02
        1: MOVER_ROVER R02 AS04 AL05
        2: MOVER_ROVER R01 AL02 AS02
        3: MOVER_ROVER R01 AS02 AL03
        4: MOVER_ROVER R02 AL05 AS04
        5: CARGAR_PERSONA R02 AS04 P01
        6: MOVER_ROVER R02 AS04 AL05
        7: MOVER_ROVER R02 AL05 AS01
        8: DESCARGAR_PERSONA R02 AS01 P01
        9: CARGAR_PERSONA R02 AS01 P03
       10: MOVER_ROVER R02 AS01 AL05
       11: MOVER_ROVER R01 AL03 AS03
       12: CARGAR_PERSONA R01 AS03 P04
       13: MOVER_ROVER R01 AS03 AL05
       14: MOVER_ROVER R02 AL05 AS03
       15: DESCARGAR_PERSONA R02 AS03 P03
       16: MOVER_ROVER R02 AS03 AL04
       17: MOVER_ROVER R01 AL05 AS04
       18: DESCARGAR_PERSONA R01 AS04 P04
       19: MOVER_ROVER R01 AS04 AL04
       20: MOVER_ROVER R02 AL04 AS04
       21: MOVER_ROVER R02 AS04 AL01
       22: CARGAR_SUMINISTRO R02 AL01 S01
       23: MOVER_ROVER R02 AL01 AS01
       24: DESCARGAR_SUMINISTRO R02 AS01 S01
       25: MOVER_ROVER R01 AL04 AS04
       26: MOVER_ROVER R01 AS04 AL01
       27: CARGAR_SUMINISTRO R01 AL01 S04
```

```
28: MOVER_ROVER R01 AL01 AS04
29: DESCARGAR_SUMINISTRO R01 AS04 S04
30: MOVER_ROVER R01 AS04 AL01
31: CARGAR_SUMINISTRO R01 AL01 S03
32: MOVER_ROVER R01 AL01 AS04
33: MOVER_ROVER R01 AS04 AL05
34: MOVER_ROVER R01 AL05 AS03
35: DESCARGAR_SUMINISTRO R01 AS03 S03
36: MOVER_ROVER R02 AS01 AL05
37: MOVER_ROVER R02 AL05 AS02
38: CARGAR_PERSONA R02 AS02 P02
39: MOVER_ROVER R02 AS02 AL05
40: MOVER_ROVER R02 AL05 AS01
41: DESCARGAR_PERSONA R02 AS01 P02
42: MOVER_ROVER R02 AS01 AL01
43: CARGAR_SUMINISTRO R02 AL01 S02
44: MOVER_ROVER R02 AL01 AS04
45: MOVER_ROVER R02 AS04 AL05
46: MOVER_ROVER R02 AL05 AS02
47: DESCARGAR_SUMINISTRO R02 AS02 S02
```

Es veu clarament que encara mantenen aquesta diferència de moviments entre ells, però a part han augmentat els moviments. Això és degut al fet que si un *rover* està en una base, però és ple, no pot recollir, per tant, haurà de passar més tard o passar un altre *rover*.

7.3 Jocs de prova extensió 2

Com sens demana en l'explicació del problema, els rovers nomes poden funcionar mentre tinguin gasolina. Hem fet les dues versions demanades, una on nomes mira que els rovers no es quedin sense gasolina(prob_ext2.pddl) i una altre(prob_ext2_v2.pddl) on mira de gastar el mínim de gasolina. Les planificacions són les següents, per prob_ext2.pddl:

```
step    0: MOVER_ROVER R01 AS01 AL02
        1: MOVER_ROVER R01 AL02 AS02
        2: MOVER_ROVER R01 AS02 AL03
        3: CARGAR_SUMINISTRO R01 AL03 S02
        4: MOVER_ROVER R01 AL03 AS02
        5: DESCARGAR_SUMINISTRO R01 AS02 S02
        6: MOVER_ROVER R01 AS02 AL03
        7: MOVER_ROVER R01 AL03 AS02
        8: MOVER_ROVER R01 AS02 AL03
        9: MOVER_ROVER R01 AL03 AS02
       10: MOVER_ROVER R01 AS02 AL03
       11: MOVER_ROVER R01 AL03 AS02
       12: MOVER_ROVER R01 AS02 AL03
       13: MOVER_ROVER R01 AL03 AS02
       14: MOVER_ROVER R01 AS02 AL03
       15: MOVER_ROVER R01 AL03 AS02
       16: MOVER_ROVER R01 AS02 AL02
       17: MOVER_ROVER R02 AS04 AL04
       18: CARGAR_SUMINISTRO R02 AL04 S01
       19: MOVER_ROVER R02 AL04 AS04
       20: MOVER_ROVER R02 AS04 AL05
       21: MOVER_ROVER R02 AL05 AS01
```

```
22: DESCARGAR_SUMINISTRO R02 AS01 S01
23: MOVER_ROVER R02 AS01 AL01
24: CARGAR_SUMINISTRO R02 AL01 S03
25: MOVER_ROVER R02 AL01 AS04
26: MOVER_ROVER R02 AS04 AL05
27: MOVER_ROVER R02 AL05 AS03
28: DESCARGAR_SUMINISTRO R02 AS03 S03
29: CARGAR_PERSONA R02 AS03 P01
30: CARGAR_PERSONA R02 AS03 P02
31: MOVER_ROVER R02 AS03 AL05
32: MOVER_ROVER R02 AL05 AS04
33: DESCARGAR_PERSONA R02 AS04 P01
34: CARGAR_PERSONA R02 AS04 P03
35: MOVER_ROVER R02 AS04 AL05
36: MOVER_ROVER R02 AL05 AS03
37: DESCARGAR_PERSONA R02 AS03 P03
38: MOVER_ROVER R02 AS03 AL05
39: MOVER_ROVER R02 AL05 AS01
40: DESCARGAR_PERSONA R02 AS01 P02
```

i per prob_ext2_v2.pddl, on hem afegit "(:metric minimize(suma-combustible-gastado))":

```
step    0: MOVER_ROVER R01 AS01 AL02
        1: MOVER_ROVER R01 AL02 AS02
        2: MOVER_ROVER R01 AS02 AL03
        3: CARGAR_SUMINISTRO R01 AL03 S02
        4: MOVER_ROVER R01 AL03 AS03
        5: MOVER_ROVER R02 AS04 AL01
        6: CARGAR_SUMINISTRO R02 AL01 S03
        7: MOVER_ROVER R02 AL01 AS01
        8: MOVER_ROVER R02 AS01 AL05
        9: MOVER_ROVER R01 AS03 AL03
       10: MOVER_ROVER R01 AL03 AS02
       11: DESCARGAR_SUMINISTRO R01 AS02 S02
       12: MOVER_ROVER R02 AL05 AS03
       13: DESCARGAR_SUMINISTRO R02 AS03 S03
       14: MOVER_ROVER R01 AS02 AL02
       15: CARGAR_PERSONA R02 AS03 P01
       16: CARGAR_PERSONA R02 AS03 P02
       17: MOVER_ROVER R02 AS03 AL05
       18: MOVER_ROVER R02 AL05 AS04
       19: DESCARGAR_PERSONA R02 AS04 P01
       20: CARGAR_PERSONA R02 AS04 P03
       21: MOVER_ROVER R02 AS04 AL05
       22: MOVER_ROVER R02 AL05 AS01
       23: DESCARGAR_PERSONA R02 AS01 P02
       24: MOVER_ROVER R02 AS01 AL05
       25: MOVER_ROVER R02 AL05 AS03
       26: DESCARGAR_PERSONA R02 AS03 P03
       27: MOVER_ROVER R02 AS03 AL04
       28: CARGAR_SUMINISTRO R02 AL04 S01
       29: MOVER_ROVER R02 AL04 AS03
       30: MOVER_ROVER R02 AS03 AL05
       31: MOVER_ROVER R02 AL05 AS01
```

```
32: DESCARGAR_SUMINISTRO R02 AS01 S01
```

Evidentment en la primera versió es fan més moviments, ja que no li demanem cap optimització. En canvi, en la segona versió busca minimitzar la gasolina (que equival als moviments entre bases) i passem de 29 moviments a 21.

7.4 Jocs de prova extensió 3

En aquesta extensió també se'ns demanava dues versions i a més a més hem fet servir un dels jocs de prova generat pel nostre script, aquest últim hi aprofundirem en el següent subapartat. Dels jocs de prova que ens centrarem en aquest apartat, hem fet una amb una heurística tenint en compte el combustible utilitzat i les prioritats de les peticions i en l'altre només la prioritat de les peticions.

Primer veurem la planificació del problema que només utilitza les prioritats, en la qual fem servir "(:metric maximize(suma-prioridades))ï el problema de la qual està en prob_ext3.pddl:

```
step    0: MOVER_ROVER R02 AS04 AL01
        1: MOVER_ROVER R02 AL01 AS01
        2: MOVER_ROVER R01 AS01 AL01
        3: MOVER_ROVER R01 AL01 AS04
        4: CARGAR_PERSONA R01 AS04 P03
        5: MOVER_ROVER R01 AS04 AL01
        6: MOVER_ROVER R02 AS01 AL02
        7: MOVER_ROVER R01 AL01 AS01
        8: MOVER_ROVER R02 AL02 AS02
        9: MOVER_ROVER R02 AS02 AL03
       10: CARGAR_SUMINISTRO R02 AL03 S02
       11: MOVER_ROVER R02 AL03 AS02
       12: DESCARGAR_SUMINISTRO R02 AS02 S02
       13: MOVER_ROVER R02 AS02 AL02
       14: MOVER_ROVER R01 AS01 AL01
       15: MOVER_ROVER R01 AL01 AS04
       16: MOVER_ROVER R01 AS04 AL04
       17: MOVER_ROVER R01 AL04 AS03
       18: DESCARGAR_PERSONA R01 AS03 P03
       19: MOVER_ROVER R01 AS03 AL05
       20: MOVER_ROVER R01 AL05 AS04
       21: MOVER_ROVER R02 AL02 AS01
       22: MOVER_ROVER R01 AS04 AL04
       23: CARGAR_SUMINISTRO R01 AL04 S01
       24: MOVER_ROVER R01 AL04 AS04
       25: MOVER_ROVER R01 AS04 AL01
       26: MOVER_ROVER R02 AS01 AL01
       27: CARGAR_SUMINISTRO R02 AL01 S03
       28: MOVER_ROVER R02 AL01 AS01
       29: MOVER_ROVER R02 AS01 AL05
       30: MOVER_ROVER R02 AL05 AS03
       31: DESCARGAR_SUMINISTRO R02 AS03 S03
       32: MOVER_ROVER R01 AL01 AS01
       33: CARGAR_PERSONA R02 AS03 P01
       34: CARGAR_PERSONA R02 AS03 P02
       35: MOVER_ROVER R02 AS03 AL05
       36: MOVER_ROVER R02 AL05 AS01
       37: DESCARGAR_PERSONA R02 AS01 P02
```

```
        38: DESCARGAR_PERSONA R02 AS01 P01
        39: MOVER_ROVER R02 AS01 AL05
        40: DESCARGAR_SUMINISTRO R01 AS01 S01
```

i per altra banda tenim la generada per prob_ext3_v2.pddl fent servir

```
(:metric maximize(+ (suma-prioridades) (* 2 (suma-combustible-no-gastado))))

step    0: MOVER_ROVER R01 AS01 AL02
        1: MOVER_ROVER R01 AL02 AS02
        2: MOVER_ROVER R01 AS02 AL03
        3: CARGAR_SUMINISTRO R01 AL03 S02
        4: MOVER_ROVER R01 AL03 AS02
        5: DESCARGAR_SUMINISTRO R01 AS02 S02
        6: MOVER_ROVER R01 AS02 AL02
        7: MOVER_ROVER R01 AL02 AS01
        8: CARGAR_PERSONA R02 AS04 P03
        9: MOVER_ROVER R02 AS04 AL01
       10: MOVER_ROVER R02 AL01 AS01
       11: MOVER_ROVER R01 AS01 AL01
       12: CARGAR_SUMINISTRO R01 AL01 S03
       13: MOVER_ROVER R01 AL01 AS04
       14: MOVER_ROVER R01 AS04 AL04
       15: MOVER_ROVER R02 AS01 AL05
       16: MOVER_ROVER R02 AL05 AS03
       17: DESCARGAR_PERSONA R02 AS03 P03
       18: MOVER_ROVER R02 AS03 AL04
       19: MOVER_ROVER R01 AL04 AS03
       20: DESCARGAR_SUMINISTRO R01 AS03 S03
       21: MOVER_ROVER R01 AS03 AL03
       22: MOVER_ROVER R01 AL03 AS02
       23: MOVER_ROVER R01 AS02 AL02
       24: MOVER_ROVER R02 AL04 AS03
       25: CARGAR_PERSONA R02 AS03 P01
       26: CARGAR_PERSONA R02 AS03 P02
       27: MOVER_ROVER R02 AS03 AL05
       28: MOVER_ROVER R02 AL05 AS01
       29: DESCARGAR_PERSONA R02 AS01 P02
       30: DESCARGAR_PERSONA R02 AS01 P01
       31: MOVER_ROVER R02 AS01 AL05
       32: MOVER_ROVER R02 AL05 AS04
       33: MOVER_ROVER R02 AS04 AL04
       34: CARGAR_SUMINISTRO R02 AL04 S01
       35: MOVER_ROVER R02 AL04 AS03
       36: MOVER_ROVER R02 AS03 AL05
       37: MOVER_ROVER R02 AL05 AS01
       38: DESCARGAR_SUMINISTRO R02 AS01 S01
```

En aquest cas també tenim una petita millora de moviments en millorar la mètrica a la segona versió, passem de 29 moviments a 27 moviments. Això és degut al fet que en la primera versió fa el possible per sumar en prioritats, podent així anar fins a l'altra punta del mapa i consumint el combustible necessari.

7.5 Generador de jocs de prova

Hem fet un script en Python que genera problemes aleatoris. Aquest script per ser executar es fa amb la següent comanda

```
python punt-extra-1.py seed variable_incremental
```

on seed és la llavors del random per poder generar els mateixos casos que en l'experiment i en la variable_incremental posem "recursos", "peticions"o "rovers"en funció del que vulguem anar incrementant i veure com evoluciona la planificació. Dins del programa tenim unes variables a la part superior que és el nombre de coses de cada tipus que hi haurà abans d'incrementar cap cosa, l'estat inicial. També podem limitar els segons amb els quals avorta l'execució si el joc de prova tarda més. El mapa de les bases utilitzat per tots els problemes generats amb aquest script és amb 10 assentaments i 5 magatzems, formant un graf connex entre tots ells i que per més informació es pot consultar el codi pddl.

Per generar un sol joc de prova com és aquest cas, hem modificat aquests valors segons conveniència i hem posat un problema prou ambiciós, amb només una iteració. D'aquesta manera ens ha creat el problema i ha fet l'execució. El problema(prob_ext3_v2_random.pddl) generat era el següent:

```
   ...
(:objects
r1 r2 - rover
al1 al2 al3 al4 al5 - almacen
as1 as2 as3 as4 as5 as6 as7 as8 as9 as10 - asentamiento
p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 - persona
s1 s2 s3 s4 s5 s6 s7 s8 - suministro
)

(:init
(= (capacidad-combustible) 100)
(= (suma-prioridades) 54)
(= (suma-combustible-no-gastado) 200)

(= (cantidad-personas r1) 0)
(= (cantidad-suministros r1) 0)
(= (combustible-gastado r1) 0)

(= (cantidad-personas r2) 0)
(= (cantidad-suministros r2) 0)
(= (combustible-gastado r2) 0)

;ESTACIONAMENT DELS ROVERS
(estacionado r1 al2) (estacionado r2 as8)

;MATRIU D ADJACENCIES
(adyacente as8 as1) (adyacente as1 as8)
(adyacente al4 al2) (adyacente al2 al4)

           ...

(adyacente as9 as10) (adyacente as10 as9)
(adyacente as10 al1) (adyacente al1 as10)

;PETICIONS DE LES PERSONES I LES SEVES PRIORITATS
```

```
 (peticion p1 as9) (= (prioridad-peticion p1 as9) 1) (enBase p1 as9)
 (peticion p2 as9) (= (prioridad-peticion p2 as9) 2) (enBase p2 as10)

 ...

 (peticion p6 as9) (= (prioridad-peticion p6 as9) 3)
 (peticion p5 as9) (= (prioridad-peticion p5 as9) 2)

 ;PETICIONS DELS SUBMINISTRES I LES SEVES PRIORITATS
 (peticion s1 as2) (= (prioridad-peticion s1 as2) 3) (enBase s1 al4)
 (peticion s2 as1) (= (prioridad-peticion s2 as1) 2) (enBase s2 al1)

            ...

 (peticion s8 as7) (= (prioridad-peticion s8 as7) 2)
 (peticion s6 as2) (= (prioridad-peticion s6 as2) 3)
 )

 (:goal (forall (?rec - recursos) (servido ?rec)))
 (:metric maximize(+ (suma-prioridades) (* 2 (suma-combustible-no-gastado))))
)
```

i la planificació donada per aquest problema ha sigut la següent:

```
step    0: CARGAR_SUMINISTRO R1 AL2 S7
        1: MOVER_ROVER R1 AL2 AS2
        2: CARGAR_PERSONA R2 AS8 P6
        3: MOVER_ROVER R2 AS8 AS1
        4: DESCARGAR_PERSONA R2 AS1 P6
        5: MOVER_ROVER R2 AS1 AL1
        6: CARGAR_SUMINISTRO R2 AL1 S2
        7: SE_QUEDA P1 AS9
        8: MOVER_ROVER R2 AL1 AS1
        9: DESCARGAR_SUMINISTRO R2 AS1 S2
       10: MOVER_ROVER R2 AS1 AS8
       11: DESCARGAR_SUMINISTRO R1 AS2 S7
       12: MOVER_ROVER R1 AS2 AL2
       13: CARGAR_SUMINISTRO R1 AL2 S4
       14: MOVER_ROVER R1 AL2 AS2
       15: MOVER_ROVER R2 AS8 AL5
       16: MOVER_ROVER R1 AS2 AS1
       17: DESCARGAR_SUMINISTRO R1 AS1 S4
       18: MOVER_ROVER R1 AS1 AS2
       19: CARGAR_SUMINISTRO R2 AL5 S3
       20: MOVER_ROVER R2 AL5 AS8
       21: MOVER_ROVER R2 AS8 AS1
       22: DESCARGAR_SUMINISTRO R2 AS1 S3
       23: MOVER_ROVER R2 AS1 AS8
       24: MOVER_ROVER R2 AS8 AL5
       25: MOVER_ROVER R2 AL5 AS9
       26: CARGAR_PERSONA R2 AS9 P3
       27: MOVER_ROVER R2 AS9 AL5
       28: MOVER_ROVER R2 AL5 AS8
```

```
29: DESCARGAR_PERSONA R2 AS8 P3
30: MOVER_ROVER R2 AS8 AL5
31: MOVER_ROVER R2 AL5 AS9
32: MOVER_ROVER R2 AS9 AS10
33: MOVER_ROVER R1 AS2 AL2
34: CARGAR_PERSONA R2 AS10 P2
35: CARGAR_PERSONA R2 AS10 P7
36: MOVER_ROVER R2 AS10 AS9
37: MOVER_ROVER R1 AL2 AS3
38: DESCARGAR_PERSONA R2 AS9 P2
39: MOVER_ROVER R1 AS3 AS4
40: MOVER_ROVER R2 AS9 AS6
41: MOVER_ROVER R1 AS4 AL3
42: MOVER_ROVER R2 AS6 AL4
43: CARGAR_SUMINISTRO R1 AL3 S5
44: MOVER_ROVER R1 AL3 AS4
45: MOVER_ROVER R1 AS4 AL1
46: MOVER_ROVER R2 AL4 AS6
47: MOVER_ROVER R1 AL1 AS1
48: DESCARGAR_SUMINISTRO R1 AS1 S5
49: MOVER_ROVER R1 AS1 AL1
50: MOVER_ROVER R1 AL1 AS4
51: CARGAR_PERSONA R1 AS4 P9
52: MOVER_ROVER R1 AS4 AL1
53: MOVER_ROVER R1 AL1 AS1
54: MOVER_ROVER R1 AS1 AS2
55: DESCARGAR_PERSONA R1 AS2 P9
56: CARGAR_PERSONA R2 AS6 P10
57: MOVER_ROVER R2 AS6 AS5
58: DESCARGAR_PERSONA R2 AS5 P10
59: CARGAR_PERSONA R2 AS5 P4
60: MOVER_ROVER R2 AS5 AS6
61: MOVER_ROVER R2 AS6 AL4
62: MOVER_ROVER R2 AL4 AL2
63: MOVER_ROVER R2 AL2 AS2
64: DESCARGAR_PERSONA R2 AS2 P4
65: MOVER_ROVER R2 AS2 AL2
66: MOVER_ROVER R2 AL2 AL4
67: MOVER_ROVER R2 AL4 AS6
68: CARGAR_PERSONA R2 AS6 P8
69: MOVER_ROVER R2 AS6 AS9
70: DESCARGAR_PERSONA R2 AS9 P8
71: MOVER_ROVER R2 AS9 AS6
72: MOVER_ROVER R2 AS6 AL4
73: MOVER_ROVER R2 AL4 AS7
74: DESCARGAR_PERSONA R2 AS7 P7
75: CARGAR_PERSONA R2 AS7 P5
76: MOVER_ROVER R2 AS7 AL4
77: MOVER_ROVER R2 AL4 AS6
78: DESCARGAR_PERSONA R2 AS6 P5
79: MOVER_ROVER R2 AS6 AL4
80: CARGAR_SUMINISTRO R2 AL4 S8
81: MOVER_ROVER R2 AL4 AS6
```

```
82: DESCARGAR_SUMINISTRO R2 AS6 S8
83: MOVER_ROVER R2 AS6 AL4
84: CARGAR_SUMINISTRO R2 AL4 S1
85: MOVER_ROVER R2 AL4 AL2
86: MOVER_ROVER R2 AL2 AS2
87: MOVER_ROVER R1 AS2 AL2
88: MOVER_ROVER R1 AL2 AL4
89: CARGAR_SUMINISTRO R1 AL4 S6
90: MOVER_ROVER R1 AL4 AL2
91: MOVER_ROVER R1 AL2 AS2
92: DESCARGAR_SUMINISTRO R1 AS2 S6
93: DESCARGAR_SUMINISTRO R2 AS2 S1
```

on podem destacar la complexitat del problema i amb la facilitat amb la que ho ha resolt. Per altre banda podem veure un dels operadors poc utilitzats per la dificultat de produir-se la casuística, com es SE_QUEDA.

7.6 Experiments amb el generador de jocs de prova

Utilitzant el script del que hem parlat al subapartat anterior amb tot el seu potencial, hem augmentat la complexitat dels problemes de manera que veiéssim el comportament del planificador.

Els paràmetres amb els quals comença tot increment són els següents:

```
maxim_seg_iteracio = 300
domain_file = "domini_ext3"
num_rovers=2
num_persones=2
num_subministres=2
gasolina_inici=50
num_peticions_persones=4
num_peticions_subministres=4
limit_iteracions = 10
```

En el primer cas ho hem fet per les variables recursos, amb la següent comanda:

```
python punt-extra-1.py 1 "recursos"
```

i hem obtingut els següent resultats:

```
Analitzant que passa amb incrementar recursos (si en una iteracio esta mes de
300 segons s'atura)
    Per 2 persones i 2 subministres ha tardat 0.05941295623779297 segons.
    Per 3 persones i 3 subministres ha tardat 0.007480621337890625 segons.
    Per 4 persones i 4 subministres ha tardat 0.0646970272064209 segons.
    Per 5 persones i 5 subministres ha tardat 3.954686164855957 segons.
    Per 6 persones i 6 subministres ha tardat 0.5978763103485107 segons.
    Per 7 persones i 7 subministres ha tardat 63.13110971450806 segons.
    Per 8 persones i 8 subministres ha tardat 22.519129037857056 segons.
    Per 9 persones i 9 subministres ha tardat 300.01017212867737 segons.
```

on veiem la tendència esperada d'anar augmentant el temps d'execució, però per altra banda hi ha algun factor que sembla més complex, que fa que alguns problemes amb menys recursos tinguin temps d'execucions molt superiors. Aquests casos els atribuïm al fet que són problemes aleatoris i segons com hagi distribuït els recursos, els rovers. Per 9 persones i 9 subministres ha tardat més de 5 minuts i això a fet matar el procés, donant així per acabat aquesta secció de l'experiment i disposar-nos a fer la següent part. Així doncs ara hem fet un experiment semblant però mirant quina és la evolució en augmentar les peticions. Això ho hem fet amb la següent comanda:

```
python punt-extra-1.py 1 "peticions"
```

i hem obtingut els següents resultats:

```
Analitzant que passa amb incrementar peticions (si en una iteracio esta mes
de 300 segons s'atura)
    Per 4 peticions de persones i 4 peticions de subministres ha tardat
    0.0593719482421875 segons.
    Per 5 peticions de persones i 5 peticions de subministres ha tardat
    0.005164384841918945 segons.
    Per 6 peticions de persones i 6 peticions de subministres ha tardat
    0.007175445556640625 segons.
    Per 7 peticions de persones i 7 peticions de subministres ha tardat
    0.008069753646850586 segons.
    Per 8 peticions de persones i 8 peticions de subministres ha tardat
    0.041407108306884766 segons.
    Per 9 peticions de persones i 9 peticions de subministres ha tardat
    0.004355669021606445 segons.
    Per 10 peticions de persones i 10 peticions de subministres ha tardat
    0.004734992980957031 segons.
    Per 11 peticions de persones i 11 peticions de subministres ha tardat
    0.004592418670654297 segons.
    Per 12 peticions de persones i 12 peticions de subministres ha tardat
    0.0863654613494873 segons.
    Per 13 peticions de persones i 13 peticions de subministres ha tardat
    0.004177570343017578 segons.
```

pensàvem que seria l'experiment més fàcil, però no esperàvem que tant, ja que ha sigut pràcticament instantani, ha acabat les 10 iteracions amb menys 1 segon. Sembla fins i tot que el problema en comptes d'augmentar en complexitat, disminuir. Això pot ser causat per la facilitat de trobar una base amb una petició.

Ara farem el mateix augmentat el nombre de rovers disponibles:

```
python punt-extra-1.py 1 "rovers"
```

i hem obtingut els següents resultats:

```
Analitzant que passa amb incrementar rovers (si en una iteracio esta mes de 300 segons
s'atura)
    Per 2 rovers ha tardat 0.05301094055175781 segons.
    Per 3 rovers ha tardat 0.01284170150756836 segons.
    Per 4 rovers ha tardat 0.01685810089111328 segons.
    Per 5 rovers ha tardat 0.2015233039855957 segons.
    Per 6 rovers ha tardat 0.02639460563659668 segons.
    Per 7 rovers ha tardat 300.0106084346771 segons.
```

Hem vist que en la sisena iteració el programa ha hagut de ser tallat per la durada del planificador. Això pot haver sigut degut al factor de ramificació que ha adquirit.

Per concloure aquest apartat volem fer èmfasi en el fet que en augmentar dues de les variables fan que el problema augmenti de complexitat, mentre que per peticions passa el contrari.

8 Conclusions

Havent acabat ja el treball, ens hem reunit per extreure quin és el coneixement extret de la pràctica, i de l'assignatura en general. Ens hem adonat que *PDDL* és un llenguatge semblant a *CLIPS* però ens ha costat molt menys adaptar-nos-hi. A part, hem sabut distingir molt clarament tots els elements del problema i com relacionar-los.
Mirant enrere, els objectius de la pràctica considerem haver-los assolit:

Hem aconseguit implementar mitjançant un llenguatge de descripció el domini i problema d'una situació donada.

Hem aplicat també la metodologia de disseny incremental basat en prototipatge ràpid, i per tal de provar aquests dissenys hem fet ús de jocs de prova per identificar errors en el nostre codi.

Un altre objectiu que es comenta és el d'utilitzar un llenguatge que sigui compatible amb planificadors moderns i això sí que s'agraeix. Les principals dificultats que ens hem trobat han sigut identificar tots els possibles casos que es podien donar i arribar per arribar a una solució.

També hem tingut problemes per entendre com funcionava la mètrica, ja que no ho vam trobar gaire intuïtiu quan havíem de maximitzar un valor decreixent i minimitzar-ne un de creixent.

El fet de no poder *debuggar* el codi també ens ha dificultat bastant la feina, sobretot en l'aspecte d'optimitzacions com la del combustible gastat, ja que havíem de revisar tots els passos un a un per veure el resultat final. En casos de problemes senzills no afectava a causa del curt nombre d'accions, però en casos on els problemes generats eren més llargs podien complicar les coses.

Juntament amb la pràctica de SBC, ens hem topat amb un tipus de llenguatge al que no estàvem acostumats: el caràcter *declaratiu*, el funcionament d'aquests sistemes... ens ha semblat una experiència molt enriquidora.

En general estem molt satisfets amb la feina realitzada.