TERCERA PRÀCTICA

INTEL·LIGÈNCIA ARTIFICIAL

Planning Domain Definition Language

Borràs Civil, Bernat
Ramos Núñez, Oscar
Ros i Roger, Alexandre

2023.01.09

I. INRTODUCCIÓ

En aquesta tercera i última pràctica d'Intel·ligència Artificial, se'ns demana dissenyar el domini d'un problema en *PDDL*, i posteriorment crear instàncies del problema per a comprovar-ne l'eficiència.

En el problema en qüestió, nosaltres som els encarregats de planificar els moviments d'un seguit de ròvers a Mart, que hauran de moure béns i serveis entre bases per a servir peticions d'altres bases.

Concretament, tenim un conjunt finit de bases, d'on es construeix un graf no dirigit i connex, prenent les bases com a vèrtexs i les carreteres o connexions entre bases com a arestes d'aquest graf. Els nostres ròvers només podran moure's entre dues bases sense passar per cap altra si entre aquestes hi ha una carretera.

Posteriorment, se'ns descriu una partició del conjunt de bases en dos subconjunts que anomenarem "magatzems" i "assentaments". Les bases "magatzem" només contenen productes, que anomenarem "càrregues", i les bases "assentament" només contenen personal especialitzat necessitat, que anomenarem "persones".

A més a més, un subconjunt dels assentaments fa un nombre de peticions de càrrega i personal.

Nosaltres, a la nostra disposició, tenim un nombre finit de ròvers que es mouran entre les bases. Aquests ròvers poden transportar persones i càrregues, sense necessitat d'ésser tripulats.

El nostre objectiu és dissenyar el domini i problemes en PDDL per a aquest exercici, per tal de planificar els moviments exactes dels ròvers minimitzant una sèrie de criteris. A més a més, tindrem diferents restriccions segons l'extensió del problema.

II. DOMINI

En el domini base, no tenim restricció ni en la capacitat dels ròvers ni en la seva energia disponible.

Començarem amb les variables del nostre domini:

```
rover - object
base - object
asentamiento almacen - base
localizable - object
persona carga - localizable
```

Aquestes variables no són molt misterioses, però hem de comentar que, donat que les persones i càrregues es comportaran de forma idèntica en aquest domini base, hem ajuntat els dos tipus en un supertipus "localitzable". Així, no haurem de replicar accions idèntiques. També distingim entre els dos tipus de base (els magatzemts tenen inicialment les càrregues i els assentaments són els responsables de realitzar les peticions).

Seguidament, els predicats:

`(esta-en ?loc - localizable ?b - base)`
Aquest predicat ens indica la posició d'una persona o càrrega **a terra o base**. És a dir, si un ròver està transportant la càrrega *x*, llavors no existeix cap base per la qual el predicat sigui cert amb el localitzable *x*.

`(aparcado-en ?rov - rover ?b - base)`
Aquest predicat ens indica la posició d'un ròver. S'ha emprat l'ús de l'expressió "aparcat" per a distingir el predicat de l'anterior. Semlant a l'anterior predicat aquest és essencial per el nostre problema ja que no té sentit fer *load* o *unload* d'elements localitzables a bases on el rover no està.

`(esta-en-rover ?loc - localizable ?rov - rover)`
Aquest predicat ens indica sobre quin ròver està un cert localitzable. Si el localitzable no està carregat sobre un ròver, llavors no existeix cap ròver pel qual el predicat sigui cert amb dit localitzable. Aquest predicat ens permet identificar el rover en el qual l'objecte localitzable està carregat, per tal que no el pugui descarregar un altre rover a una posició que no és la del rover on (realment) està l'objecte.

`(hay-camino ?bas - base ?bas2 - base)`
Aquest predicat ens relaciona dues bases amb una carretera. És a dir, un ròver pot anar de la base *bas* a la base *bas2* amb un pas i viceversa si, i només si, `(or (hay-camino bas bas2) (hay-camino bas2 bas))` (donat que és un graf no dirigit). Malgrat és cert que el graf de bases és conex pot no ser complet, per tant no volem que els rovers facin un camí més curt dels que realment podrien fer i creem aquest predicat per tal que els rovers només es puguin moure entre bases on "hay camino" directe (relacionades en una aresta al graf).

`(petition ?loc - localizable ?b - base)`
Aquest predicat ens informa que la base *b* ha realitzat una petició d'un localitzable. Aquest predicat s'empra a totes les extensions (0, 1, 2) on les peticions no tenen prioritat i ens servirà per a determinar si un localitzable pot ser entregat a una base.

`(important-petition ?loc - localizable ?b - base)`
`(medium-petition ?loc - localizable ?b - base)`
`(low-petition ?loc - localizable ?b - base)`
Aquestes tres peticions s'utilitzen a l'extensió 3 i substitueixen al predicat anterior. Amb aquestes es distigueixen les prioritats de les peticions realitzades fent que el rover tendeixi a entregar els localitzables a bases que han fet peticions més importants. Aquests predicats són necessaris ja que, a diferència de les dues extensions anteriors, a l'extensió 3 no totes les peticions són iguals i hem de mirar d'entregar els objectes a on s'han realitzat les important peticions.

`(servido ?loc - localizable)`
Aquest predicat ens indica si un localitzable ja ha estat entregat a una base que l'ha demanat, o si encara està disponible. Possiblement un localitzable serà demanat a

més d'una base. Aquest predicat és útil per no donar la possibilitat a un rover de tornar a fer *load* d'un objecte entregat ja que aquest, anteriorment, ja s'ha tractat i entregat.

A continuació parlarem de les funcions:

`(current-capacity ?rov - rover)`
Aquesta funció és utilitzada a les extensions 1, 2 i 3 on els rovers tenen una capacitat limitada i ens permet saber, en cada moment, què es pot carregar a cada rover. A l'inici tots els rovers comencen amb la capacitat a 0. Donat que podem afegir fins a dues persones o una càrrega, hem considerat que la capacitat màxima d'un rover serà de dues unitats on una persona ocupa una unitat i una càrrega n'ocupa dos. Aquest fluent és necessari per a les precondicions per a fer el *load* a un rover.

`(gas-level ?rov - rover)`
Aquesta funció és utilitzada a les extensions 2 i 3 on s'ha de mantenir la benzina restant a cada rover, ja que cada moviment entre bases empra una unitat de benzina. El valor inicial de `gas-level` de cada rover és arbitrari i canvia entre problemes diferents provocant que per a certes configuracions el probleme quedi sense solució, tot i així el generador proposat de problemes, al inicialitzar els valors del problema, garantitza suficient benzina per tal que els rovers puguin servir les peticions de tots els localitzables.

`(sum-petitions)`
Aquesta funció s'utilitza a l'extensió 3, on no totes les peticions són igual d'importants. Nosaltres hem decidit que el valor inicial de `sum-petitions` sigui *3*(#càrregues + #persones)* i, quan entreguem cada càrrega o persona, depenent de l'importància de la petició feta en la base en la qual s'entrega, es resti 1, 2 o 3 unitats si la petició és *high, medium o low,* respectivament. Aquest valor inicial també provoca que el valor d'aquest fluent mai sigui negatiu.

Finalment, les accions possibles del domini:

`entrar (amb paràmetres:?loc - localitable, ?b - base)`
Per executar aquesta acció cal que ?loc estigui a la base ?b, que la base ?b tingui alguna petició (low, medium o high) de l'objecte ?loc i que aquest objecte no estigui "servido".
Al realitzar aquesta acció l'objecte passa a estar "servido" i no està a la base ?b; a més, depenent de l'importància de la petició feta per ?b sobre ?loc es decrementa 1, 2 o 3 per les high, medium i low petitions respectivament (per la mètrica).

`mover-rover (params.:?rov - rover, ?from - base, ?to - base)`
Per executar aquesta acció cal que el rover ?rov estigui aparcat a ?from, que hi hagi un camí de ?from a ?to (predicat "hay-camino") i que el *gas level* de ?rov sigui major a 1 (per el cas de les extensions 2 i 3).
Al realitzar aquesta acció ?rov passa a estar apartcat a ?to, es decrementa en 1 (extensions 2 i 3) el seu *gas level* i també ?rov passa a no estar aparcat a ?from.

`load-persona (params.:?rov - rover, ?loc - persona, ?b - base)`
Per executar aquesta acció cal que ?rov estigui aparcat a ?b, ?loc estigui a ?b i, per les extensions 2 i 3, que la capacitat actual del rover sigui inferior a 2 (0 o 1 permeten afegir una persona).
Al realitzar aquesta acció ?loc passa a estar en el rover ?rov (predicat esta-en-rover) i passa a no estar a ?b, a més, per les extensions 2 i 3, la capacitat actual del rover s'incrementa en 1.

`load-carga (params.:?rov - rover, ?loc - carga, ?b - base)`
Per executar aquesta acció cal que ?rov estigui aparcat a ?b, ?loc estigui a ?b i, per les extensions 1, 2 i 3, que la capacitat actual del rover sigui inferior a 1 (0 ja que una càrrega ocupa tota la capacitat d'un rover (considerem que ocupa les 2 espais)).
Al realitzar aquesta acció ?loc passa a estar en el rover ?rov (predicat esta-en-rover) i passa a no estar a ?b, a més, per les extensions 1, 2 i 3, la capacitat actual del rover s'incrementa en 2 (fent que ja no es pugui fer cap load més fins que es descarregui la càrrega).

`unload-persona(params:?rov - rover, ?loc - persona, ?b - base)`
Per executar aquesta acció cal que ?rov estigui aparcat a ?b i que ?loc estigui en el rover ?rov.
Al realitzar aquesta acció ?loc passa a estar a ?b i passa a no estar en el rover ?rov, a més, per les extensions 1, 2 i 3, la capacitat de ?rov es decrementa en 1.

`unload-carga (params.:?rov - rover, ?loc - persona, ?b - base)`
Per executar aquesta acció cal que ?rov estigui aparcat a ?b i que ?loc estigui en el rover ?rov.
Al realitzar aquesta acció ?loc passa a estar a ?b i passa a no estar en el rover ?rov, a més, per les extensions 1, 2 i 3, la capacitat de ?rov es decrementa en 2.

III. FITXER DE PROBLEMA

Extensió #0: Base
El fitxer de problema està dividit en 4 parts: els objectes, l'estat inicial, l'estat objectiu, i la mètrica. A l'extensió base no apareix la quarta part.

Un exemple de la part dels objectes d'aquesta extensió és la següent:

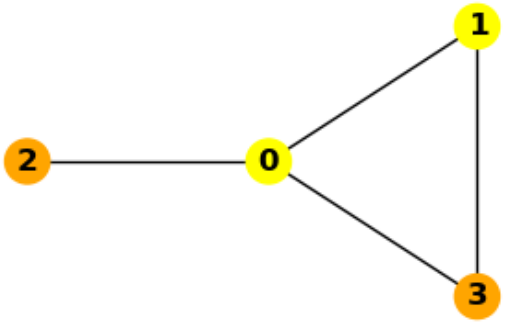
```
p0 p1 p2 - persona
c0 c1 c2 - carga
b0 b1 - asentamiento
b2 b3 - almacen
r0 r1 r2  - rover
```

En aquest exemple podem observar diversos objectes que formen part de les cinc variables comentades anteriorment al domini. En aquest cas podem observar que tenim tres objectes de la variable `persona`, tres objectes més de la variable `carga`, quatre `bases` de les quals dues són assentaments, i dues més de magatzems. Finalment tenim dos objectes de la variable `rover`.

La secció de l'estat inicial està composta pel conjunt de predicats que representen la situació actual del problema. A continuació tenim un exemple del primer predicat a l'estat inicial:

```
(hay-camino b3 b0)
(hay-camino b0 b2)
(hay-camino b3 b1)
(hay-camino b1 b0)
```

Ens indica quines bases estan connectades entre elles per un camí. En aquest exemple el graf resultant després d'afegir aquests camins és el següent:



Els colors dels nodes indiquen si la `base` és `asentamiento` (color groc) o `almacen` (color taronja).

El següent predicat per inicialitzar és `aparcado-en`, i l'exemple és el següent:

```
(aparcado-en r0 b2)
(aparcado-en r1 b1)
```

Aquí estem indicant que el rover 0 començarà a la base 2, i el rover 1 a la base 1.

També inicialitzem les posicions de les càrregues i persones, tal com es pot veure a continuació:

```
(esta-en p0 b1)
(esta-en p1 b1)
(esta-en p2 b0)
(esta-en c0 b3)
(esta-en c1 b2)
(esta-en c2 b2)
```

Aquí estem assignant a cada persona `p0`, `p1` i `p2`; als assentaments `b1`, `b1` i `b0` respectivament. Les càrregues `c0`, `c1` i `c2` són assignades als magatzems `b3`, `b2` i `b2`. Les càrregues també s'assginen utilitzant el mateix predicat ja que el predicat utilitza la variable `localizable`, que engloba `persona` i `carga`; i la variable `base`, que engloba `asentamiento` i `almacen`.

Els últims predicats que inicialitzem són els predicats de `petition`, a continuació veiem l'exemple:

```
(petition p2 b0)
(petition p2 b1)
(petition c0 b1)
(petition c0 b0)
(petition c1 b1)
(petition c2 b1)
(petition c2 b0)
(petition p0 b0)
```

El predicat de `petition` és el mateix per a `persona` i `carga` ja que utilitza les variables `localizable` i `base`, tal com ho fa el predicat `esta-en`. En aquest exemple es veu que no totes les persones tenen peticions (`persona p1`), i persones i càrrgues (`persona p1`, `carga c0` i `c1`) que tenen més d'una petició. Això és correcte ja que a l'enunciat especifica que típicament hi haurà més peticions que personal i càrrega, i al ser especialitzat podria ser que algun `localizable` sigui demanat vàries vegades, i algun altre que no sigui demanat per a cap.

La última secció de l'extensió base és l'estat objectiu, o goal. A l'exemple que hem posat, tenim el següent estat:

```
(servido p2)
(servido c0)
(servido c1)
(servido c2)
(servido p0)
```

Aquí podem observar que l'objectiu del problema és que totes les persones i càrregues estiguin servides. En el cas de la persona p1, que no hi havia cap petició, no apareix a l'estat (els rovers podran carregar-la i descarregar-la igualment).

Extensió #1: Capacitat

En aquesta extensió, el fitxer de problema és el mateix que a l'extensió 1, però inicialitzant les capacitats dels rovers a l'estat inicial, seguint l'exemple d'abans:

```
(= (current-capacity r0) 0)
(= (current-capacity r1) 0)
```

Aquí estem assignant que inicialment els dos rovers del problema no tenen cap capacitat ocupada. Com hem vist als operadors de càrrega i descàrrega aquests valors fluctuen d'entre el 0 i el 2 (on una persona ocupa 1 espai i una càrrega 2).

Extensió #2: Combustible

En aquesta extensió, el fitxer de problema és el mateix que a l'extensió 2, però afegint a l'estat inicial, el nivell de combustible que tenen els rovers, per exemple:

```
(= (gas-level r0) 5)
(= (gas-level r1) 7)
```

Els dos rovers de l'exemple d'abans tenen assignats un cert nivell de combustible. El número 5 i 7 indica el nombre de camins entre dues bases que els rovers poden fer.

En el cas que a l'extensió 2 vulguem optimitzar el nivell de combustible gastat (extensió 2.b), al final del fitxer de problema afegim l'apartat de la mètrica. El nostre exemple tindria:

```
(:metric maximize
(+ (gas-level r0)  (gas-level r1) ))
```

Aquí estem indicat que el planificador maximitzi la suma dels nivells de combustible restant dels dos rovers.

Extensió #3: Peticions amb prioritat

En aquesta extensió, utilitzem el fitxer de problema de l'extensió 2a, però amb algunes modificacions. A l'estat inicial substituirem tots els predicats `petition`, per predicats de `important-petition`, `medium-petition` o `low-petition`. A continuació veiem un exemple:

```
(low-petition p0 b1)
(low-petition p0 b0)
(low-petition c1 b1)
(important-petition c1 b0)
(medium-petition c2 b1)
(medium-petition p2 b1)
(low-petition p2 b0)
(medium-petition p1 b0)
```

Cada petició que teniem anteriorment se li a assignat una prioritat amb aquests nous predicats.

En el cas que vulguem optimitzar el problema tenint en compte les prioritats, però no el combustible utilitzat, l'apartat de la mètrica serà el següent:

```
(:metric maximize
(sum-petitions) )
```

Bàsicament li estem indicant al planificador que maximitzi la suma de peticions. Aquesta funció està explicada amb més detall a la part de domini.

En el cas que vulguem optimitzar el problema tenint en compte les prioritats com el nivell de combustible utilitzat, l'apartat de la mètrica serà (en el cas de tenir 2 rovers) el següent:

```
(:metric maximize
(+ (* (sum-petitions) 3) (+(gas-level r0)  (gas-level r1) )))
```

Podem observar que la mètrica és una combinació entre la utilitzada a l'extensió 2b amb la de l'extensió 3b. Bàsicament estem fent una ponderació entre aquestes dues. Hem trobat que multiplicar la suma de peticions per 3, el planificador troba solucions on les peticions respecten bastant les prioritats, sense tenir un consum de combustible gaire elevat. En el cas que es vulgui tenir una major importància a les prioritats de les peticions, es pot incrementar el multiplicador de la suma de peticions. En el cas contrari, es pot reduir aquest multiplicador, fins i tot per sota de 1.

IV. PROCÉS DE DESENVOLUPAMENT

En aquesta pràctica hem optat desenvolupar la pràctica de forma incremental. Vam començar construint l'extensió base. Aquí vam començar implementant el fitxer de domini amb els predicats i les operacions especificades anteriorment amb l'excepció del predicat `servido` i de l'operador `entrar`. Just després de fer la primera versió de la primera extensió vam fer un primer fitxer de problema per a comprovar que el domini es comportava de la manera adequada.

Tot seguit ens vam posar a fer el domini de la primera extensió a la vegada que formàvem el generador dels fitxers de problemes. Aquest domini també el vam fer com l'especificat, però sense el predicat `servido` i de l'operador `entrar`. Una vegada fet el primer domini vam ampliar el generador per a que suportés els dos dominis creats fins aleshores.

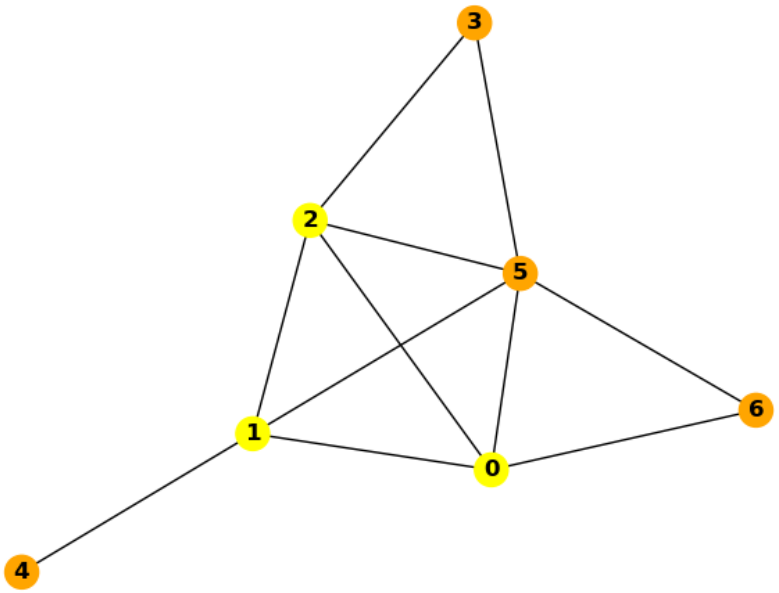
Una vegada teníem els dominis base i de la primera extensió juntament amb el generador que suportava aquests dos dominis ens vam posar a fer la segona extensió. Com els altres dominis, tal com s'especifiquen anterioriment i sense el predicat `servido` i de l'operador `entrar`. També vam ampliar el generador per suportar aquesta extensió.

Mentres formàvem i feiem proves de la tercera extensió ens vam adonar que el temps de resolució anava molt lent, i abans d'intentar optimitzar la tercera extensió vam pensar d'optimitzar des del domini base, i incrementar-ho fins a la tercera extensió. Les primeres optimitzacions consistia en reordenar les precondicions de les diverses operacions. El temps de resolució va millorar però no de forma significativa. Aleshores vam veure que les persones i les càrregues que ja estàven a destinació es seguien tenint en compte a l'hora de comprovar les precondicions. Vam decidir afegir el predicat `servido` i l'operador `entrar`. Aquesta millora ens ha permès reduir el temps de resolució de forma significativa. Per a problemes més grans, el planificador igualment trigava força estona. Aquestes optimitzacions les vam expandir fins el domini 3. Vam finalitzar el domini 3 juntament amb el generador, i vam optar per refinar el ponderador de la mètrica del domini 3 que vol optimitzar les prioritats juntament amb el combustible usat.

V. ESTUDI DEL TEMPS DE RESOLUCIÓ

Aquesta part de la pràctica estudiem com varia el temps de resolució dels problemes segons la mida d'aquest. Concretament veurem com varia el temps segons les variables següents: el número de rovers, el número de peticions, i el número de personal i subministres disponibles.

Utilitzant el nostre generador, hem creat un graf dels assentaments i magatzems, que serà utilitzat per a totes les execucions. Ja que el nostre generador el crea de manera aleatòria, després de generar un fitxer de problema, el copiarem per tal de reduir el nombre de variables externes. El graf utilitzat és el següent:



on els nodes taronges són magatzems, i els grocs assentaments.

Per a totes les execucions s'utilitzarà l'extensió 3, amb la variant on es combina les prioritats amb el combustible utilitzat.

Segons el número de rovers

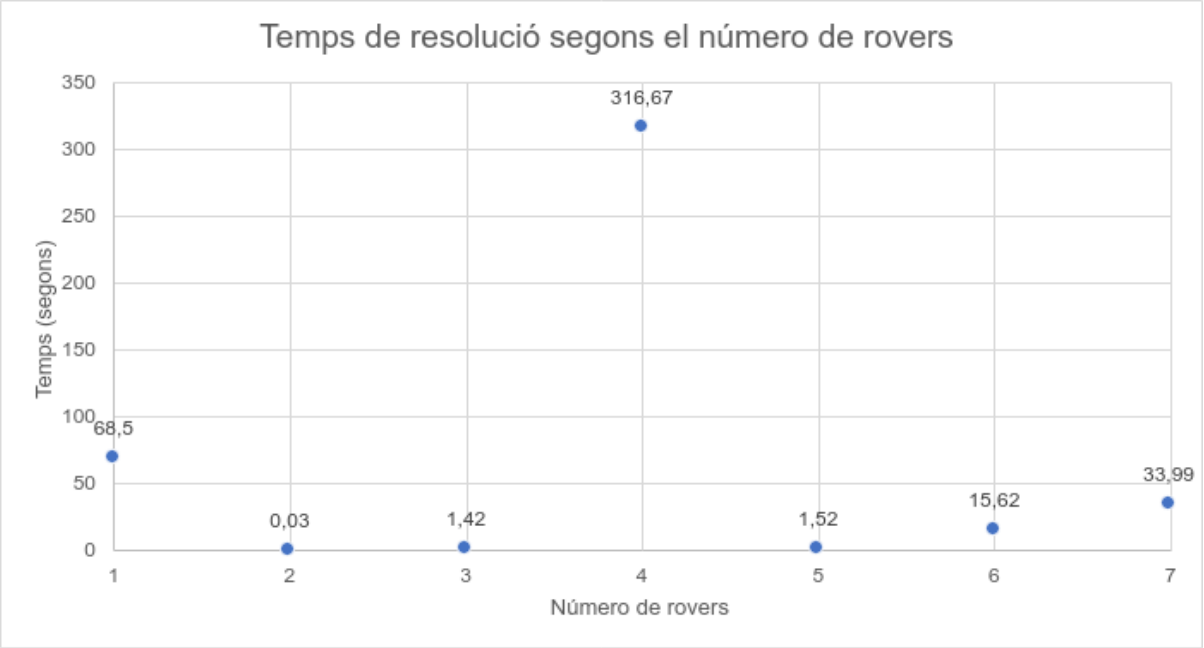
En aquesta part de l'estudi, hem decidit crear un fitxer de problema amb el graf indicat anteriorment, i amb els següents paràmetres:

6 persones
5 càrregues
16 peticions

La primera execució tindrà assignat un sòl rover, i n'afegirem un més a cada execució. Per a mantenir la màxima consistència possible, les peticions seran

copiades a totes les execucions d'aquest apartat ja que el generador les crea cada vegada de manera aleatòria.

Després de fer les execucions, obtenim el següent gràfic:



Quan el número de rovers és molt baix (1 rover) l'execució triga un minut. És bastant comparat amb el temps obtingut amb les altres (excepte amb 4 rovers). Això pot ser degut a que el planificador li costa trobar una solució vàlida si el rover no té gaire combustible. A continuació veiem que les següents dues execucions es fan de manera molt ràpida, degut a que al haver-hi més rovers es troben solucions vàlides més fàcilment. Amb 4 rovers, veiem que el temps s'ha disparat ja que deu haver de fer moltes més comparacions. Les següents execucions tornen a ser ràpides perquè al haver-hi molts rovers, és fàcil trobar una molt bona solució al permetre fer menys desplaçaments a cada rover. Veiem una tendència final on es va incrementant el temps degut al incrementar el número de rovers del problema a resoldre.

Segons el número de peticions

En aquesta part de l'estudi, hem decidit crear un fitxer de problema amb el graf indicat anteriorment, i amb els següents paràmetres:

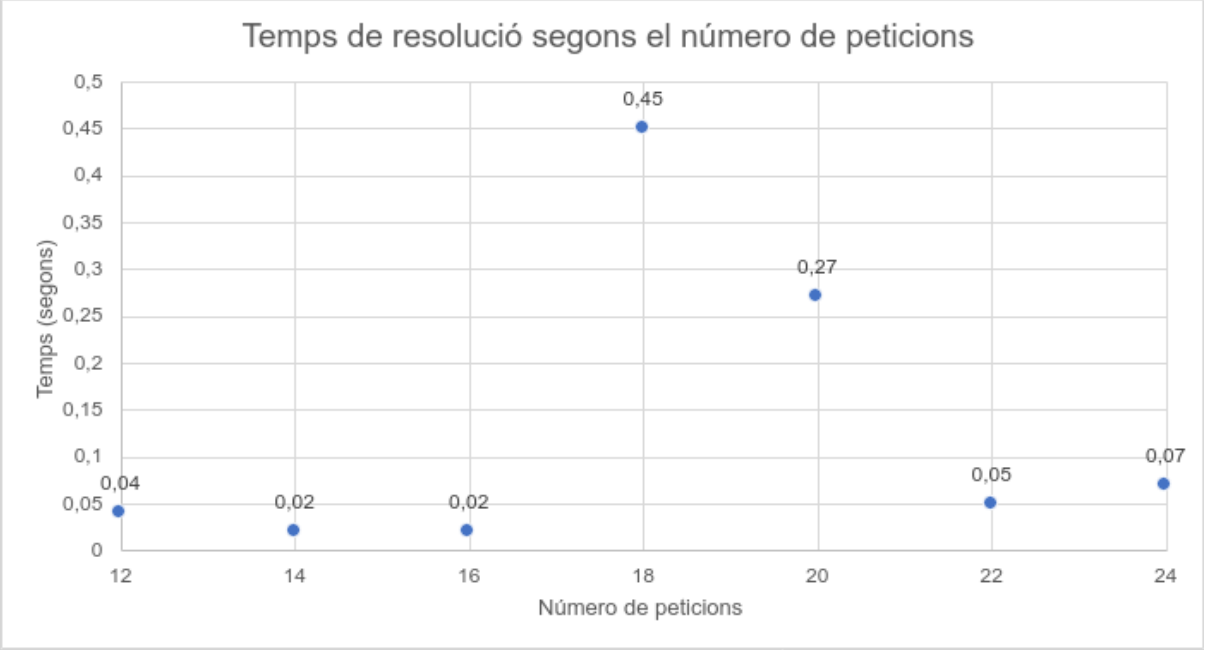
6 persones

5 càrregues

2 rovers

La primera execució tindrà assignat 12 peticions, i n'afegirem dues més a cada execució. Hem decidit començar amb 12 peticions ja que a l'enunciat especifica que hi ha més peticions que personal/càrrega disponible. Per a mantenir la màxima consistència possible, les posicions i el combustible dels rovers seran copiats ja que el generador ho assigna de forma aleatòria cada vegada que generem un fitxer de problema.

Després de fer les execucions, obtenim el següent gràfic:



Podem observar que quan hi ha poques peticions el temps de resolució és molt baix. Al incrementar aquest número, s'arriba a un punt (en aquest cas 18 peticions), on hi ha bastantes peticions (que fa augmentar el temps), però no proutes per tenir una saturació de peticions. Es pot observar a les següents execucions, que hi ha tantes peticions, que fa que la majoria de personal/càrrega només cal que es traslladi a una base pròxima a la d'orígen. Quan el nombre de peticions és molt gran, la solució donada és pràcticament la mateixa, i només el número de peticions a tenir en compte fa augmentar el temps de resolució.

Segons el número de personal/càrrega

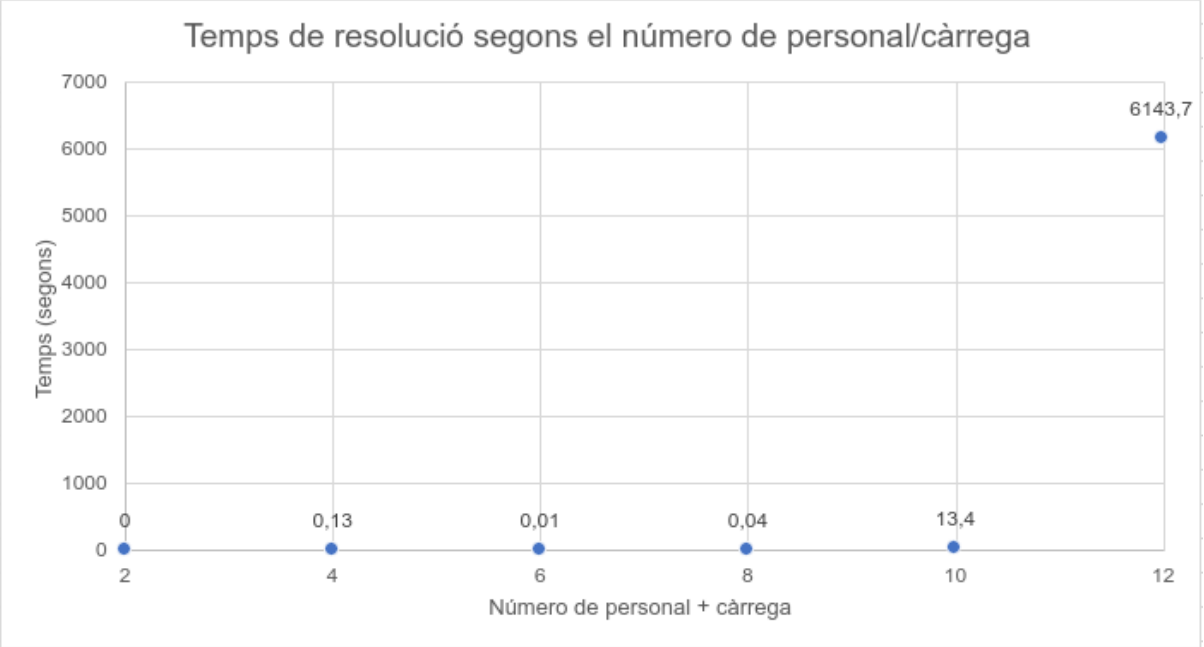
En aquesta part de l'estudi, hem decidit crear un fitxer de problema amb el graf indicat anteriorment, i amb els següents paràmetres:

2 rovers

30 peticions

La primera execució tindrà assignat una càrrega i una persona, i afegirem una persona i una càrrega més cada execució. Efectuarem la primera execució amb una sòla càrrega i una sòla persona. Per a mantenir la màxima consistència possible, les posicions i el combustible dels rovers seran copiats ja que el generador ho assigna de forma aleatòria cada vegada que generem un fitxer de problema.

Després de fer les execucions, obtenim el següent gràfic:



En aquest gràfic podem veure clarament que el temps de resolució vaira de manera exponencial respecte el número de personal i càrrega. Al inici les execucions són instantànies. Quan arribem a 5 càrregues i 5 persones, la resolució triga uns 13 segons, i amb una càrrega i una persòna més arribem a 1 hora 42 minuts.

VI. GENERADOR I JOCS DE PROVES

Per a aquesta pràctica hem desenvolupat un generador de jocs de proves amb Python, *generator.py*. Els únics requeriments per a executar-lo són instal·lar *numpy, matplotlib* i *networkx* (per a visualitzar el mapa de Mart).

El generador pregunta, inicialment, si l'usuari vol incloure cada extensió. En cas positiu, pregunta també possibles variants (per exemple, si l'usuari decideix incloure combustible alsròvers, el programa després pregunta si vol optimitzar aquest combustible). A més a més, es pregunta per el número de *persones*, *càrregues, bases, arestes* i *ròvers*.

Posteriorment, es genera aleatòriament un graf connex amb exactament el nombre d'arestes indicades (es comença generant un Spanning Tree i, després, continua afegint arestes aleatòries).

Finalment, mitjançant *NetworkX i Matplotlib*, es dibuixa el graf de les bases. En color taronja es dibuixen les bases *magatzem* i, en groc, els *asssentaments*. L'usuari pot tancar la finestra i s'haurà generat el fitxer de problema.

Amb aquest generador, hem generat sis jocs de prova:

I. Extensió 0: Base
II. Extensió 1: Capacitats
III. Extensió 2A: Combustible
IV. Extensió 2B: Optimització de Combustible
V. Extensió 3A: Prioritats
VI. Extensió 3B: Optimització de Prioritats i Combustible

Cadascun d'aquests, els hem executat amb *Metric-FF* i els resultats els teniu en els fitxers de Solució, per a cada problema (output del *Metric-FF*).

Hem escollit dos problemes d'aquest llistat per a explicar-los de manera més detallada. Els dos escollits són el 2A i el 3B perquè creiem que aquests dos representen tot el conjunt. El joc de proves de l'extensió 2A no optimitza cap mètrica i, en canvi, el 3B sí.

III. Extensió 2A: Combustible

L'estat inicial d'aquest joc de proves és el següent:

```
(hay-camino b3 b1) (hay-camino b1 b0)
(hay-camino b3 b2) (hay-camino b2 b1)
(aparcado-en r0 b0) (aparcado-en r1 b0)
(esta-en p0 b1) (esta-en p1 b1) (esta-en p2 b1)
(esta-en c0 b2) (esta-en c1 b2) (esta-en c2 b3)
(= (current-capacity r0) 0)
(= (current-capacity r1) 0)
(= (gas-level r0) 12) (= (gas-level r1) 9)
(petition p1 b0) (petition p1 b1) (petition p0 b0)
(petition c2 b0) (petition p2 b0) (petition c1 b1)
(petition c1 b0) (petition c0 b0) )
```

L'estat objectiu és:

```
(servido p1) (servido p0) (servido c2) (servido p2)
(servido c1) (servido c0)
```

El planificador ens ha donat els següents passos per resoldre'l:

```
step 0: ENTRAR P1 B1
1: MOVER-ROVER R1 B0 B1
2: LOAD-PERSONA R1 P0 B1
3: MOVER-ROVER R1 B1 B0
4: UNLOAD-PERSONA R1 P0 B0
5: ENTRAR P0 B0
6: MOVER-ROVER R1 B0 B1
7: LOAD-PERSONA R1 P2 B1
8: MOVER-ROVER R1 B1 B0
9: UNLOAD-PERSONA R1 P2 B0
10: ENTRAR P2 B0
11: MOVER-ROVER R1 B0 B1
12: MOVER-ROVER R1 B1 B3
13: LOAD-CARGA R1 C2 B3
14: MOVER-ROVER R1 B3 B2
```

```
15: MOVER-ROVER R1 B2 B1
16: MOVER-ROVER R1 B1 B0
17: UNLOAD-CARGA R1 C2 B0
18: ENTRAR C2 B0
19: MOVER-ROVER R0 B0 B1
20: MOVER-ROVER R0 B1 B2
21: LOAD-CARGA R0 C1 B2
22: MOVER-ROVER R0 B2 B1
23: UNLOAD-CARGA R0 C1 B1
24: ENTRAR C1 B1
25: MOVER-ROVER R0 B1 B2
26: LOAD-CARGA R0 C0 B2
27: MOVER-ROVER R0 B2 B1
28: MOVER-ROVER R0 B1 B0
29: UNLOAD-CARGA R0 C0 B0
30: ENTRAR C0 B0
```

Metric-FF minimitza la llargada del pla, i es compleix les restriccions de capacitat i combustible. Totes les càrregues i persones s'han servit a una base on hi havia una petició (totes les accions d'entrar). El rover 0 ha fet un total de 6 moviments, i al tenir 12 unitats de combustible, no ha fet més moviments dels permesos. El rover 1, tiena permès fer 9 moviments, i com que n'ha fet 9, no s'ha passat de moviments i per tant s'han respectat els límits de combustible.

VI. Extensió 3B: Optimització de Prioritats i Combustible

L'estat inicial d'aquest joc de proves és el següent:

```
(hay-camino b0 b3) (hay-camino b0 b1) (hay-camino b0 b2)
(hay-camino b2 b3)
(aparcado-en r0 b2) (aparcado-en r1 b2)
(esta-en p0 b1) (esta-en p1 b0) (esta-en p2 b1)
(esta-en c0 b3) (esta-en c1 b2) (esta-en c2 b2)
(= (current-capacity r0) 0) (= (current-capacity r1) 0)
(= (gas-level r0) 8) (= (gas-level r1) 7)
(= (sum-petitions) 18) (low-petition c1 b0)
(low-petition c1 b1) (important-petition p1 b1)
```

```
(low-petition p1 b0) (low-petition p0 b0)
(medium-petition p2 b0)
(medium-petition p2 b1) (important-petition c2 b1) )
```

L'estat objectiu és:

```
(:goal
(and (servido c1) (servido p1) (servido p0) (servido p2)
(servido c2) ) )

(:metric maximize (+ (* (sum-petitions) 3) (+ (gas-level r0)
(gas-level r1) )))
```

El planificador ens ha donat els següents passos per resoldre'l:

```
step 0: ENTRAR P1 B0
1: ENTRAR P2 B1
2: MOVER-ROVER R0 B2 B3
3: LOAD-CARGA R1 C1 B2
4: MOVER-ROVER R1 B2 B0
5: UNLOAD-CARGA R1 C1 B0
6: ENTRAR C1 B0
7: LOAD-CARGA R0 C0 B3
8: MOVER-ROVER R1 B0 B2
9: LOAD-CARGA R1 C2 B2
10: MOVER-ROVER R1 B2 B0
11: UNLOAD-CARGA R0 C0 B3
12: MOVER-ROVER R1 B0 B1
13: UNLOAD-CARGA R1 C2 B1
14: ENTRAR C2 B1
15: LOAD-CARGA R0 C0 B3
16: LOAD-PERSONA R1 P0 B1
17: MOVER-ROVER R1 B1 B0
18: UNLOAD-PERSONA R1 P0 B0
19: ENTRAR P0 B0
```

Es pot obvservar que la solució és correcta ja que cada objecte localitzable (càrrega i persona) acaba servida en una base que havia realitzat una petició i per tant es compleix l'objectiu.

El ròver r0 ha fet en total un sol moviment, i com que en podia fer fins a 8, es compleix la restricció de combustible. En canvi, el ròver 1 ha fet 5 moviments, però igualment entra dins dels seus límits. En total els rovers fan 6 moviments. La suma de les prioritats d'aquesta execució dóna 8. Utilitzar 6 unitats de combustible és una solució ja que té un límit de 15 unitats. També podem observar que la suma de prioritats no és la màxima, degut a que la mètrica té tendència a prioritzar l'ús de combustible.