Pràctica de Planificació

Intel·ligència Artificial Primavera 2022/23

Rubén Aciego

Mariona Jaramillo

Francesc Pifarré

Contingut

1. Introducció

En aquesta pràctica se'ns planteja desenvolupar un treball de planificació amb PDDL.
El problema consisteix en assignar un conjunt de tasques de programació que tenen un grau de dificultat assignat i un temps estimat de realització a un conjunt de programadors que tenen associada una qualitat. Depenent de la qualitat del programador, es necessitarà després realitzar una tasca de revisió de més o menys temps.
Així doncs, l'objectiu de la pràctica és utilitzar PDDL per poder resoldre un problema que necessita tractar un problema de planificació per ser resolt.
Per poder arribar a trobar una solució, anirem implementant extensions a la pràctica constructivament fins a arribar a la solució final. Aleshores serà quan generarem problemes i explorarem l'escabilitat en temps que tenen els diferents paràmetres.

2. Domini

A continuació s'explicaran detalladament les característiques del domini: variables, predicats, funcions i accions per cada extensió proposada a l'enunciat. Com que se'ns demana que diferenciem en extensions i al cap i a la fi, les extensions serveixen per arribar de manera constructiva a la solució del problema, pot ser que hi hagi informació que es repeteixi entre subapartats, però és la millor manera que hem trobat per poder documentar-ho.

2.1. Nivell bàsic

- Variables
  - Programador: representa el concepte de programador
  - Tasca: representa el concepte de tasca
- Predicats
  - (tasca_oberta ?t - tasca): representa que la tasca encara no ha estat tractada ni assignada
  - (tasca_assignada ?t - tasca ?p - programador): representa que la tasca ja ha estat tractada i assignada
- Funcions
  - (habilitat ?p - programador): representa l'habilitat d'un programador, és a dir, un valor del conjunt {1,2,3}
  - (qualitat ?p - programador): representa la qualitat d'un programador, és a dir, un valor del conjunt {1,2}
  - (dificultat ?t - tasca): representa la dificultat d'una tasca, és a dir, un valor del conjunt {1,2,3}
  - (duracio_tasca ?t - tasca): representa el temps estimat de realització d'una tasca, en hores
  - (tasques_assignades): representa el nombre total de tasques que ja han estat assignades. Es considera que una tasca ha estat assignada quan ja s'ha realitzat també la seva revisió
- Accions
  - assignar_tasca: donada una tasca i un programador, verifica primer en la precondició que la tasca estigui oberta, és a dir, que encara no hagi estat assignada i que la dificultat de la tasca és com a molt una unitat més gran que l'habilitat del programador donat.

    Un cop s'han dut a terme aquestes verificacions, significa que es pot dur a

terme l'assignació de la tasca al programador. Per aquest motiu, la tasca passa a estar tancada (no oberta), assignada al programador en qüestió i es suma una unitat a les tasques_assignades, és a dir, aquelles tasques que ja s'han desenvolupat.

2.2. Extensió 1

- Variables
  - Programador: representa el concepte de programador
  - Tasca: representa el concepte de tasca
- Predicats
  - (tasca_oberta ?t - tasca): representa que la tasca encara no ha estat tractada ni assignada
  - (tasca_assignada ?t - tasca ?p - programador): representa que la tasca ja ha estat tractada i assignada (pot ser però que encara no hagi passat la revisió)
  - (revisio_oberta ?t): representa que la revisió encara no ha estat tractada
  - (revisio_assignada ?t - tasca ?p - programador): representa que la revisió ja ha estat tractada i assignada
- Funcions
  - (habilitat ?p - programador): representa l'habilitat d'un programador, és a dir, un valor del conjunt {1,2,3}
  - (qualitat ?p - programador): representa la qualitat d'un programador, és a dir, un valor del conjunt {1,2}
  - (dificultat ?t - tasca): representa la dificultat d'una tasca, és a dir, un valor del conjunt {1,2,3}
  - (duracio_tasca ?t - tasca): representa el temps estimat de realització d'una tasca, en hores
  - (duracio_revisio ?t - tasca): representa el temps de revisió d'una tasca. Pot ser 1 o 2 hores depenent de la qualitat del programador que hagi realitzat la tasca
  - (tasques_assignades): representa el nombre total de tasques que ja han estat assignades. Es considera que una tasca ha estat assignada quan ja s'ha realitzat també la seva revisió
- Accions
  - assignar_tasca: donada una tasca i un programador, verifica primer en la precondició que la tasca estigui oberta, és a dir, que encara no hagi estat assignadai que la dificultat de la tasca és com a molt una unitat més gran que

l'habilitat del programador donat.

Un cop s'han dut a terme aquestes verificacions, significa que es pot dur a terme l'assignació de la tasca al programador. Per aquest motiu, la tasca passa a estar tancada (no oberta), assignada al programador en qüestió, s'obre una tasca de revisió i s'assigna la qualitat del programador que ha fet la tasca a la duració de la revisió de la tasca.

- ➢ assignar_revisió: donada una tasca i un programador, verifica primer en la precondició que la revisió estigui oberta, és a dir, que encara no hagi estat assignada, que la dificultat de la tasca és com a molt una unitat més gran que l'habilitat del programador donat i que la mateixa tasca no ha estat assignada ja al mateix programador (per assegurar que qui revisa la tasca sigui un programador diferent a qui l'ha desenvolupat).

  Un cop s'han dut a terme aquestes verificacions, significa que es pot dur a terme l'assignació de la tasca de revisió al programador. Per aquest motiu, la revisió passa a estar tancada, assignada al programador en qüestió i es suma una unitat a les tasques_assignades, és a dir, aquelles tasques que ja s'han desenvolupat i revisat.

2.3. Extensió 2

- ❖ Variables
  - ➢ Programador: representa el concepte de programador
  - ➢ Tasca: representa el concepte de tasca
- ❖ Predicats
  - ➢ (tasca_oberta ?t - tasca): representa que la tasca encara no ha estat tractada ni assignada
  - ➢ (tasca_assignada ?t - tasca ?p - programador): representa que la tasca ja ha estat tractada i assignada (pot ser però que encara no hagi passat la revisió)
  - ➢ (revisio_oberta ?t): representa que la revisió encara no ha estat tractada
  - ➢ (revisio_assignada ?t - tasca ?p - programador): representa que la revisió ja ha estat tractada i assignada
- ❖ Funcions
  - ➢ (habilitat ?p - programador): representa l'habilitat d'un programador, és a dir, un valor del conjunt {1,2,3}
  - ➢ (qualitat ?p - programador): representa la qualitat d'un programador, és a dir, un valor del conjunt {1,2}
  - ➢ (dificultat ?t - tasca): representa la dificultat d'una tasca, és a dir, un valor del

conjunt {1,2,3}

- (duracio_tasca ?t - tasca): representa el temps estimat de realització d'una tasca, en hores
- (duracio_revisio ?t - tasca): representa el temps de revisió d'una tasca. Pot ser 1 o 2 hores depenent de la qualitat del programador que hagi realitzat la tasca
- (tasques_assignades): representa el nombre total de tasques que ja han estat assignades. Es considera que una tasca ha estat assignada quan ja s'ha realitzat també la seva revisió
- (suma_hores): representa el nombre total d'hores (excepte la suma de la duració de les tasques) que es necessiten per completar les tasques i revisions amb l'assignació donada. Aquesta suma no té en compte les possibles tasques realitzades en paral·lel, per tant és la suma total d'hores de treball

❖ Accions

- assignar_tasca: donada una tasca i un programador, verifica primer en la precondició que la tasca estigui oberta, és a dir, que encara no hagi estat assignada i que la dificultat de la tasca és com a molt una unitat més gran que l'habilitat del programador donat.

  Un cop s'han dut a terme aquestes verificacions, significa que es pot dur a terme l'assignació de la tasca al programador. Per aquest motiu, la tasca passa a estar tancada (no oberta), assignada al programador en qüestió i s'obre una tasca de revisió. En cas que la dificultat de la tasca sigui superior a l'habilitat del programador, el temps per realitzar la tasca s'incrementa en dues hores, per aquest motiu, en aquest cas, s'incrementa en dues unitats suma_hores. Finalment, s'assigna la qualitat del programador que ha fet la tasca a la duració de la revisió de la tasca.
- assignar_revisió: donada una tasca i un programador, verifica primer en la precondició que la revisió estigui oberta, és a dir, que encara no hagi estat assignada, que la dificultat de la tasca és com a molt una unitat més gran que l'habilitat del programador donat i que la mateixa tasca no ha estat assignada ja al mateix programador (per assegurar que qui revisa la tasca sigui un programador diferent a qui l'ha desenvolupat).

  Un cop s'han dut a terme aquestes verificacions, significa que es pot dur a terme l'assignació de la tasca de revisió al programador. Per aquest motiu, la revisió passa a estar tancada, assignada al programador en qüestió, s'incrementa en una o dues unitats la funció suma_hores, depenent de la

qualitat del programador que havia desenvolupat la tasca inicial i es suma una unitat a les tasques_assignades, és a dir, aquelles tasques que ja s'han desenvolupat i revisat.

2.4. Extensió 3

- Variables
  - Programador: representa el concepte de programador
  - Tasca: representa el concepte de tasca
- Predicats
  - (tasca_oberta ?t - tasca): representa que la tasca encara no ha estat tractada ni assignada
  - (tasca_assignada ?t - tasca ?p - programador): representa que la tasca ja ha estat tractada i assignada (pot ser però que encara no hagi passat la revisió)
  - (revisio_oberta ?t): representa que la revisió encara no ha estat tractada
  - (revisio_assignada ?t - tasca ?p - programador): representa que la revisió ja ha estat tractada i assignada
- Funcions
  - (habilitat ?p - programador): representa l'habilitat d'un programador, és a dir, un valor del conjunt {1,2,3}
  - (qualitat ?p - programador): representa la qualitat d'un programador, és a dir, un valor del conjunt {1,2}
  - (noves_assginacions ?p - programador): representa el nombre de tasques o revisions que encara poden ser assignades a un programador, per tant, el seu valor ha de ser >= 0
  - (dificultat ?t - tasca): representa la dificultat d'una tasca, és a dir, un valor del conjunt {1,2,3}
  - (duracio_tasca ?t - tasca): representa el temps estimat de realització d'una tasca, en hores
  - (duracio_revisio ?t - tasca): representa el temps de revisió d'una tasca. Pot ser 1 o 2 hores depenent de la qualitat del programador que hagi realitzat la tasca
  - (tasques_assignades): representa el nombre total de tasques que ja han estat assignades. Es considera que una tasca ha estat assignada quan ja s'ha realitzat també la seva revisió
  - (suma_hores): representa el nombre total d'hores (excepte la suma de la duració de les tasques) que es necessiten per completar les tasques i

revisions amb l'assignació donada. Aquesta suma no té en compte les possibles tasques realitzades en paral·lel, per tant és la suma total d'hores de treball

- Accions
  - assignar_tasca: donada una tasca i un programador, verifica primer en la precondició que la tasca estigui oberta, és a dir, que encara no hagi estat assignada, que al programador encara se li puguin assignar noves tasques (comprovant que el valor de la funció de noves_assignacions és superior a 0) i que la dificultat de la tasca és com a molt una unitat més gran que l'habilitat del programador donat.

    Un cop s'han dut a terme aquestes verificacions, significa que es pot dur a terme l'assignació de la tasca al programador. Per aquest motiu, la tasca passa a estar tancada (no oberta), assignada al programador en qüestió i s'obre una tasca de revisió. En cas que la dificultat de la tasca sigui superior a l'habilitat del programador, el temps per realitzar la tasca s'incrementa en dues hores, per aquest motiu, en aquest cas, s'incrementa en dues unitats suma_hores. També es resta una unitat a les assignacions que encara pot tenir un programador i finalment, s'assigna la qualitat del programador que ha fet la tasca a la duració de la revisió de la tasca.
  - assignar_revisió: donada una tasca i un programador, verifica primer en la precondició que la revisió estigui oberta, és a dir, que encara no hagi estat assignada, que al programador encara se li puguin assignar noves tasques, que la dificultat de la tasca és com a molt una unitat més gran que l'habilitat del programador donat i que la mateixa tasca no ha estat assignada ja al mateix programador (per assegurar que qui revisa la tasca sigui un programador diferent a qui l'ha desenvolupat).

    Un cop s'han dut a terme aquestes verificacions, significa que es pot dur a terme l'assignació de la tasca de revisió al programador. Per aquest motiu, la revisió passa a estar tancada, assignada al programador en qüestió i s'incrementa en una o dues unitats la funció suma_hores, depenent de la qualitat del programador que havia desenvolupat la tasca inicial. També es resta una unitat a les assignacions que encara pot tenir un programador i es suma una unitat a les tasques_assignades, és a dir, aquelles tasques que ja s'han desenvolupat i revisat.

2.5. Extensió 4

- Variables
  - Programador: representa el concepte de programador
  - Tasca: representa el concepte de tasca
- Predicats
  - (tasca_oberta ?t - tasca): representa que la tasca encara no ha estat tractada ni assignada
  - (tasca_assignada ?t - tasca ?p - programador): representa que la tasca ja ha estat tractada i assignada (pot ser però que encara no hagi passat la revisió)
  - (revisio_oberta ?t): representa que la revisió encara no ha estat tractada
  - (revisio_assignada ?t - tasca ?p - programador): representa que la revisió ja ha estat tractada i assignada
- Funcions
  - (habilitat ?p - programador): representa l'habilitat d'un programador, és a dir, un valor del conjunt {1,2,3}
  - (qualitat ?p - programador): representa la qualitat d'un programador, és a dir, un valor del conjunt {1,2}
  - (noves_assginacions ?p - programador): representa el nombre de tasques o revisions que encara poden ser assignades a un programador, per tant, el seu valor ha de ser >= 0
  - (dificultat ?t - tasca): representa la dificultat d'una tasca, és a dir, un valor del conjunt {1,2,3}
  - (duracio_tasca ?t - tasca): representa el temps estimat de realització d'una tasca, en hores
  - (duracio_revisio ?t - tasca): representa el temps de revisió d'una tasca. Pot ser 1 o 2 hores depenent de la qualitat del programador que hagi realitzat la tasca
  - (tasques_assignades): representa el nombre total de tasques que ja han estat assignades. Es considera que una tasca ha estat assignada quan ja s'ha realitzat també la seva revisió
  - (suma_hores): representa el nombre total d'hores (excepte la suma de la duració de les tasques) que es necessiten per completar les tasques i revisions amb l'assignació donada. Aquesta suma no té en compte les possibles tasques realitzades en paral·lel, per tant és la suma total d'hores de treball
  - (programadors): representa el nombre total de programadors que s'han

necessitat per poder fer l'assignació de tasques i revisions a aquests programadors

- Accions
  - assignar_tasca: donada una tasca i un programador, verifica primer en la precondició que la tasca estigui oberta, és a dir, que encara no hagi estat assignada, que al programador encara se li puguin assignar noves tasques (comprovant que el valor de la funció de noves_assignacions és superior a 0) i que la dificultat de la tasca és com a molt una unitat més gran que l'habilitat del programador donat.

    Un cop s'han dut a terme aquestes verificacions, significa que es pot dur a terme l'assignació de la tasca al programador. Per aquest motiu, la tasca passa a estar tancada (no oberta), assignada al programador en qüestió i s'obre una tasca de revisió. En cas que la dificultat de la tasca sigui superior a l'habilitat del programador, el temps per realitzar la tasca s'incrementa en dues hores, per aquest motiu, en aquest cas, s'incrementa en dues unitats suma_hores. A més a més, s'incrementa en 1 els programadors utilitzats en cas que encara tingués el paràmetre de noves_assignacions per defecte (és a dir, 2) perquè això vol dir que no s'ha utilitzat mai abans aquell programador. També es resta una unitat a les assignacions que encara pot tenir un programador i finalment, s'assigna la qualitat del programador que ha fet la tasca a la duració de la revisió de la tasca.
  - assignar_revisió: donada una tasca i un programador, verifica primer en la precondició que la revisió estigui oberta, és a dir, que encara no hagi estat assignada, que al programador encara se li puguin assignar noves tasques, que la dificultat de la tasca és com a molt una unitat més gran que l'habilitat del programador donat i que la mateixa tasca no ha estat assignada ja al mateix programador (per assegurar que qui revisa la tasca sigui un programador diferent a qui l'ha desenvolupat).

    Un cop s'han dut a terme aquestes verificacions, significa que es pot dur a terme l'assignació de la tasca de revisió al programador. Per aquest motiu, la revisió passa a estar tancada, assignada al programador en qüestió i s'incrementa en una o dues unitats la funció suma_hores, depenent de la qualitat del programador que havia desenvolupat la tasca inicial. A més a més, s'incrementa en 1 els programadors utilitzats en cas que encara tingués el paràmetre de noves_assignacions per defecte (és a dir, 2). També es resta 1 a les assignacions que encara poden tenir un programador i es suma 1 a les tasques_assignades (aquelles tasques que s'han desenvolupat i revisat)

3. Modelat del problema

Per resoldre les diferents extensions, proposem les següents modelitzacions del problema. Com havíem dit abans, es tracta d'una solució constructiva a mesura que es va avançant en les extensions, per tant, descriurem els elements que varien en comparació a l'extensió anterior.

3.1.Nivell bàsic

- Objectes
    - Tasca: representa una tasca d'un projecte informàtic i s'assigna a un programador
    - Programador: representa una persona amb una habilitat (menor o mejor) per programar i que programa amb una certa qualitat. Se li assignen tasques a completar.
- Estat inicial
    - L'estat inicial té totes les tasques i programadors sense assignar
    - Per tant, totes les tasques són obertes i les tasques assignades són 0
    - Es defineix la dificultat de les tasques i l'habilitat dels programadors
    - Es defineix la duració de les tasques
- Estat final: totes les tasques han estat assignades

3.2. Extensió 1

- Estat inicial
    - Es defineix la qualitat dels programadors

3.3. Extensió 2

- Estat inicial
    - La suma d'hores s'incialitza a 0
- Mètrica: a banda de l'estat final, s'afegeix una mètrica, que és minimitzar el temps total que es fa servir per resoldre totes les tasques (suma de les hores):
  (:metric minimize (suma_hores))

3.4. Extensió 3

- Estat inicial
    - S'inicialitzen les noves assignacions de cada programador a 2, de manera que cada programador pot completar com a molt dues tasques

3.5. Extensió 4

- Estat inicial
    - S'inicialitzen els programdors utilitzats per completar les tasques a 0
- Mètrica: es busca optimitzar la suma ponderada entre el número de persones que estan fent les tasques i el temps total que tarden en resoldre-les. Per fer això, hem considerat la ponderació:
  suma_hores/suma_hores_tasques + programadors/programadors_totals
  Com que suma_hores_tasques i programadors_totals són constants pel problema, si fem denominador comú, és el mateix que minimitzar el numerador: *suma_hores*programadors_totals + programadors*suma_hores_tasques*
  En aquestes expressions, suma_hores es refereix a la suma d'hores extra que s'afegeixen a la duració de les tasques per temes de revisió o dhabilitat del programador; suma_hores_tasques es refereix a la suma de la duració de totes les tasques; programadors es refereix a tots els programadors als quals se'ls ha assignat alguna tasca; programadors_totals es refereix a tots els programadors que hi ha disponibles per desenvolupar tasques.

4. Desenvolupament dels models

Per desenvolupar els models, hem fet servir la metodologia constructiva i iterativa, a través de totes les extensions, començant des d'un nivell molt bàsic, com ja hem vist en els apartats anteriors.

A més, hem anat fent tests sobre els models per poder veure si funcionaven correctament. Per tal de no haver de generar manualment els testos, hem escrit un generador de problemes aleatori que, a més a més, és únic per totes les extensions: es pot indicar per quina extensió volem generar un problema aleatori i el generador genera un test amb els objectes, incialitzacions, goal i metric corresponent a l'extensió seleccionada.

5. Jocs de prova

Per comprovar el correcte funcionament dels models per resoldre el problema en totes les seves extensions, s'ha programat un script de Python per redactar jocs de prova i instàncies del problema.

Aquest script rep els següents paràmetres:

- Problem: el nom del problema
- Programadors: el nombre de programadors a assignar
- Tasques: el nombre de tasques a completar
- Extension: el nivell d'extensió del model per al que es vol generar un problema
- Seed: la seed per al generador de nombres aleatoris.

L'script inicialitza el nombre de programadors i tasques donades, amb un nivell d'habilitat i qualitat aleatori per cada programador i un nombre aleatori per a les dificultats i durades de les tasques.

Amb aquest script hem generat els següents jocs de prova que procedim a desenvolupar:

5.1 Joc de prova 1 - Cas bàsic

Per la primera implementació del problema, es genera un joc de prova amb dos programadors i dos tasques:

- **Entrada:**

```
(define (problem jocProva1) (:domain planificador1)
(:objects
   p0 p1 - programador
   t0 t1 - tasca
)
(:init
   (= (tasques_assignades) 0)
   (= (habilitat p0) 1)
   (= (habilitat p1) 2)
   (= (qualitat p0) 2)
   (= (qualitat p1) 1)
   (= (dificultat t0) 2)
   (= (dificultat t1) 2)
   (= (duracio_tasca t0) 2)
   (= (duracio_tasca t1) 2)
```

```
    (tasca_oberta t0)
    (tasca_oberta t1)
)
(:goal
    (= (tasques_assignades) 2)
)
)
```

- **Sortida:**

```
step 0: ASSIGNAR_TASCA T1 P1
     1: ASSIGNAR_REVISIO T1 P0
     2: ASSIGNAR_TASCA T0 P0
     3: ASSIGNAR_REVISIO T0 P1
```

Així, el problema es resol correctament assignant una tasca i una revisió a cada un dels programadors.

5.2 Joc de prova 2 - Extensió 1

Considerem, per la extensió bàsica del problema, un joc de prova de mida més gran que el bàsic anterior, amb 4 programadors i 4 tasques a realitzar.

- **Entrada:**

```
(define (problem jocProva2) (:domain planificador1)
(:objects
    p0 p1 p2 p3 - programador
    t0 t1 t2 t3 - tasca
)
(:init
    (= (tasques_assignades) 0)
    (= (habilitat p0) 2)
    (= (habilitat p1) 1)
    (= (habilitat p2) 1)
    (= (habilitat p3) 1)
    (= (qualitat p0) 3)
    (= (qualitat p1) 1)
    (= (qualitat p2) 3)
    (= (qualitat p3) 3)
```

```
    (= (dificultat t0) 1)
    (= (dificultat t1) 1)
    (= (dificultat t2) 2)
    (= (dificultat t3) 1)
    (= (duracio_tasca t0) 1)
    (= (duracio_tasca t1) 1)
    (= (duracio_tasca t2) 1)
    (= (duracio_tasca t3) 6)
    (tasca_oberta t0)
    (tasca_oberta t1)
    (tasca_oberta t2)
    (tasca_oberta t3)
)
(:goal
    (= (tasques_assignades) 4)
)
)
```

- **Sortida:**

```
step 0: ASSIGNAR_TASCA T3 P3
     1: ASSIGNAR_REVISIO T3 P0
     2: ASSIGNAR_TASCA T2 P3
     3: ASSIGNAR_REVISIO T2 P0
     4: ASSIGNAR_TASCA T1 P3
     5: ASSIGNAR_REVISIO T1 P0
     6: ASSIGNAR_TASCA T0 P0
     7: ASSIGNAR_REVISIO T0 P1
```

De nou, tot i ser un problema més complex amb una mida més gran, s'assignen correctament les tasques i revisions a realitzar.

5.3 Joc de prova 3 - Extensió 2

Considerem, per el canvi d'extensió, un joc de prova amb les mateixes característiques que l'anterior per veure si la solució, en tenir en compte l'Extensió 2 minimitzar el temps total per resoldre les tasques, canvia per adaptar-se a aquest canvi.

- **Entrada:**

```
(define (problem jocProva3) (:domain planificador2)
```

```
(:objects
   p0 p1 p2 p3 - programador
   t0 t1 t2 t3 - tasca
)
(:init
   (= (tasques_assignades) 0)
   (= (suma_hores) 0)
   (= (habilitat p0) 2)
   (= (habilitat p1) 1)
   (= (habilitat p2) 1)
   (= (habilitat p3) 1)
   (= (qualitat p0) 3)
   (= (qualitat p1) 1)
   (= (qualitat p2) 3)
   (= (qualitat p3) 3)
   (= (dificultat t0) 1)
   (= (dificultat t1) 1)
   (= (dificultat t2) 2)
   (= (dificultat t3) 1)
   (= (duracio_tasca t0) 1)
   (= (duracio_tasca t1) 1)
   (= (duracio_tasca t2) 1)
   (= (duracio_tasca t3) 6)
   (tasca_oberta t0)
   (tasca_oberta t1)
   (tasca_oberta t2)
   (tasca_oberta t3)
)
(:goal
   (= (tasques_assignades) 4)
)
(:metric minimize (suma_hores))
)
```

- **Sortida:**

```
step 0: ASSIGNAR_TASCA T0 P0
      1: ASSIGNAR_REVISIO T0 P1
```

2: ASSIGNAR_TASCA T1 P0
3: ASSIGNAR_REVISIO T1 P1
4: ASSIGNAR_TASCA T2 P0
5: ASSIGNAR_REVISIO T2 P1
6: ASSIGNAR_TASCA T3 P0
7: ASSIGNAR_REVISIO T3 P1

El problema, de nou, es resol correctament i dona una assignació diferent a la que s'havia trobat en l'anterior joc de prova. Es pot comprovar, desfent les assignacions, que correspon a un nombre d'hores totals menors a la solució proposada en el joc de prova anterior, tal com es demana de l'Extensió 2.

5.4 Joc de prova 4 - Extensió 3

Per a l'Extensió 3, hem considerat un joc de prova de nou amb 4 programadors i 4 tasques generat amb una seed diferent als dos anteriors. Volem comprovar, per aquest joc, que es resol correctament segons les especificacions d'aquesta Extensió: limitant el nombre de tasques que pot realitzar un programador a 2.

- **Entrada:**

```
(define (problem jocProva4) (:domain planificador3)
(:objects
   p0 p1 p2 p3 - programador
   t0 t1 t2 t3 - tasca
)
(:init
   (= (tasques_assignades) 0)
   (= (suma_hores) 0)
   (= (habilitat p0) 2)
   (= (habilitat p1) 2)
   (= (habilitat p2) 2)
   (= (habilitat p3) 1)
   (= (qualitat p0) 2)
   (= (qualitat p1) 1)
   (= (qualitat p2) 2)
   (= (qualitat p3) 2)
   (= (noves_assginacions p0) 2)
```

```
    (= (noves_assginacions p1) 2)
    (= (noves_assginacions p2) 2)
    (= (noves_assginacions p3) 2)
    (= (dificultat t0) 2)
    (= (dificultat t1) 3)
    (= (dificultat t2) 2)
    (= (dificultat t3) 3)
    (= (duracio_tasca t0) 4)
    (= (duracio_tasca t1) 1)
    (= (duracio_tasca t2) 3)
    (= (duracio_tasca t3) 8)
    (tasca_oberta t0)
    (tasca_oberta t1)
    (tasca_oberta t2)
    (tasca_oberta t3)
)
(:goal
    (= (tasques_assignades) 4)
)
(:metric minimize (suma_hores))
)
```

- **Sortida:**

```
step 0: ASSIGNAR_TASCA T3 P0
     1: ASSIGNAR_TASCA T0 P0
     2: ASSIGNAR_REVISIO T3 P1
     3: ASSIGNAR_TASCA T2 P2
     4: ASSIGNAR_REVISIO T2 P3
     5: ASSIGNAR_TASCA T1 P1
     6: ASSIGNAR_REVISIO T1 P2
     7: ASSIGNAR_REVISIO T0 P3
```

El problema es resol correctament i es veu de l'assignació proposada per el model que compleix les especificacions de evitar que un programador realitzi més de dues tasques.

5.5 Joc de prova 5 - Extensió 4

Per a l'Extensió 4, hem procedit com en els jocs de prova 2-3 i generat un problema amb les mateixes característiques per comprovar que, en aquest cas, la solució és diferent que la proposada en el joc de prova 4 en tenir en compte aquesta Extensió una heurística per minimitzar la suma entre les hores invertides i el nombre de programadors treballant.

- **Entrada:**

```
(define (problem jocProva5) (:domain planificador4)
(:objects
   p0 p1 p2 p3 - programador
   t0 t1 t2 t3 - tasca
)
(:init
   (= (tasques_assignades) 0)
   (= (suma_hores) 0)
   (= (programadors) 0)
   (= (habilitat p0) 2)
   (= (habilitat p1) 2)
   (= (habilitat p2) 2)
   (= (habilitat p3) 1)
   (= (qualitat p0) 2)
   (= (qualitat p1) 1)
   (= (qualitat p2) 2)
   (= (qualitat p3) 2)
   (= (noves_assginacions p0) 2)
   (= (noves_assginacions p1) 2)
   (= (noves_assginacions p2) 2)
   (= (noves_assginacions p3) 2)
   (= (dificultat t0) 2)
   (= (dificultat t1) 3)
   (= (dificultat t2) 2)
   (= (dificultat t3) 3)
   (= (duracio_tasca t0) 4)
   (= (duracio_tasca t1) 1)
   (= (duracio_tasca t2) 3)
   (= (duracio_tasca t3) 8)
```

```
    (tasca_oberta t0)
    (tasca_oberta t1)
    (tasca_oberta t2)
    (tasca_oberta t3)
)
(:goal
    (= (tasques_assignades) 4)
)
(:metric minimize (+ (* 4 (suma_hores)) (* 16 (programadors))))
)
```

- **Sortida:**

```
step 0: ASSIGNAR_TASCA T3 P1
     1: ASSIGNAR_TASCA T1 P1
     2: ASSIGNAR_REVISIO T1 P2
     3: ASSIGNAR_TASCA T2 P2
     4: ASSIGNAR_REVISIO T3 P0
     5: ASSIGNAR_TASCA T0 P0
     6: ASSIGNAR_REVISIO T0 P3
     7: ASSIGNAR_REVISIO T2 P3
```

El problema arriba de nou a una solució que, com podíem preveure, és diferent a la proposada pel model de l'Extensió 3. Es pot comprovar que l'assignació d'hores és menor que la proposada en aquell cas, funcionant correctament l'Extensió.

6. Temps d'execució

Per estudiar la complexitat temporal del model hem considerat problemes de diferents mides i estudiat com evoluciona el temps necessari per resoldre'ls en l'extensió bàsica del model. Considerem la mida d'un problema el nombre de programadors i tasques a assignar, que fixem en el mateix valor.

Per obtenir els temps de resolució, resolem 5 problemes d'aquest tamany i calculem la mitjana d'aquests temps. Degut al format que ens dona el FastForward, els resultats son significatius en segons només fins a la segona xifra decimal, fent que en mides petites es pugui perdre informació sobre el temps de resolució colapsant els valors en 0.

D'aquestes figures podem concloure que la complexitat del problema es exponencial respecte el nombre de programadors i tasques a assignar.

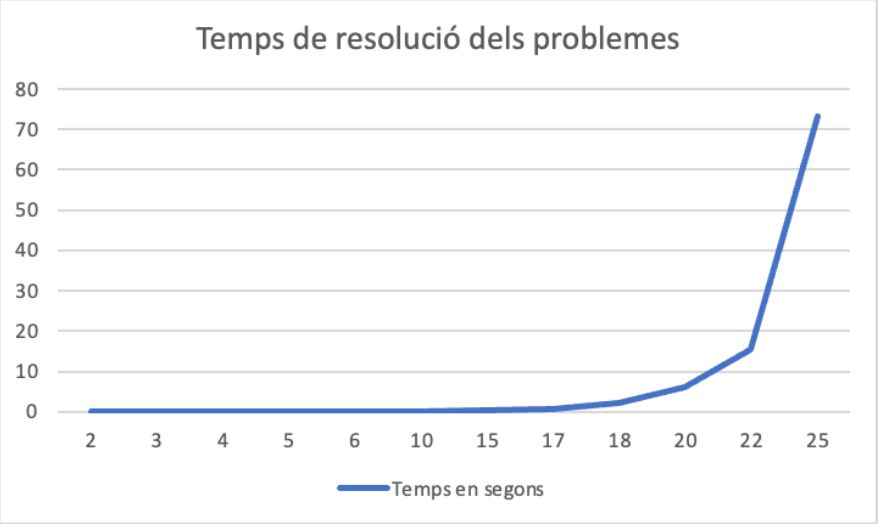

Figura 1: Temps de resolució obtingut pels problemes, en escala lineal.

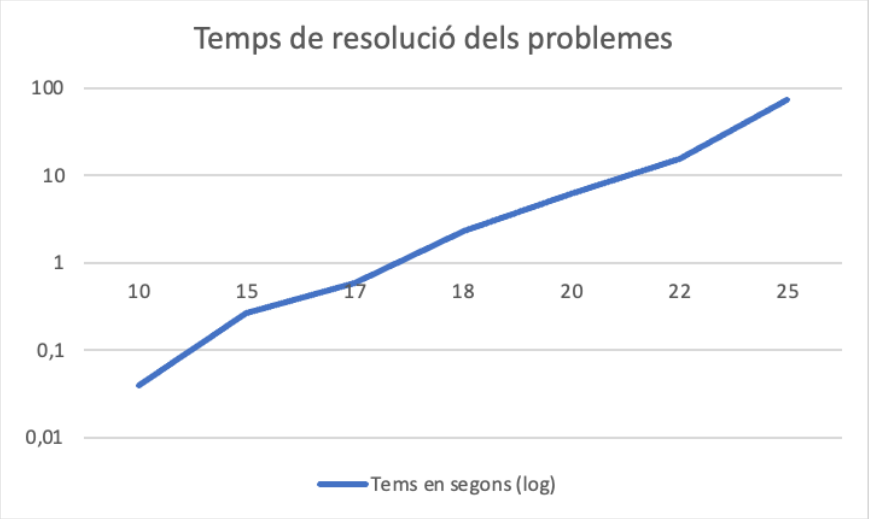

Figura 2: Temps de resolució obtingut pels problemes, en escala logarítmica.

7. Conclusions

Aquesta pràctica ens ha permès desenvolupar un model per solucionar un problema de planificació, podent entendre com funcionen aquests sistemes amb restriccions lògiques i objectius modelats en forma de heurístiques.
El modelat del problema i el seu domini son la part més important de cara a enfocar un problema de planificació, ja que el procediment de resolució del mateix es sustenta en les decisions preses en aquest moment.
Desenvolupant el model iterativament usant PDLL i el FastForward, afegint complexitat amb cada extensió, permet un mode de treball constructiu en el que cada extensió considera solucions més complexes o amb diferents objectius que les anteriors.
Comprovant la correctesa aquestes solucions amb diferents jocs de prova i mesurant el seu cost a nivell temporal, podem entendre com s'estan solucionant els diferents problemes proposats i evaluar si compleixen correctament els objectius proposats. La mesura de la complexitat temporal també permet veure la complexitat dels problemes de planificació, en el nostre cas fins i tot en l'extensió més bàsica, que fa que el seu cost de resolució es dispari exponencialment quan creix també la mida del problema.