Inteligencia Artificial

Práctica de Planificación

Adrian Cristian Crisan
Javier Castaño
Priyanka Amarnani
Mark Smithson

Departamento de Computer Science

4 de enero de 2023

Índice

1. Introducción

En esta práctica se nos plantea desarrollar un trabajo de planificación con PDDL. El problema que tenemos, consiste en que estamos asentados en Marte y se necesitan cubrir ciertas necesidades de transporte. Para cubrir las necesidades de los asentamientos, estos hacen peticiones tanto de personal especializado como de suministros a un organismo centralizado. Cada petición es de una unidad de suministro o una persona, si se quiere más se pueden hacer varias peticiones. Este organismo se encarga de planificar los desplazamientos de los rovers desde la base donde se encuentran aparcados, hasta recoger los suministros y el personal necesarios y finalmente para dejarlos en lugar donde se ha hecho la petición.

El objetivo de esta práctica es enfrentarse a un problema que necesita hacer un trabajo de planificación para ser solucionado.

Una vez tengamos la solución del problema con sus diferentes extensiones, generaremos problemas y exploraremos la escalabilidad en tiempo que tienen los diferentes parámetros.

2. Dominio

2.1. Nivel básico

- **Variables**
  - Rovers: Representa al Rover.
  - Base: Se compone de Almacenes y Asentamientos que son subtipos de base
  - Petición: Representa el concepto de petición
- **Predicados**
  - (estacionado ?r - Rover ?b - base): representa la base en la que esta estacionado el Rover.
  - (camino ?b1 - base ?b2 - base): representa la existencia de un camino entre ambas bases.
  - (peticion-abierta ?p - peticion ?a - asentamiento): representa que asentamiento ha hecho la petición.
  - (peticion-suministros ?p - peticion): representa que la petición es para un suministro.
  - (peticion-personal ?p - peticion): representa que la petición es para un personal.
- **Funciones**
  - (suministros-Rover ?r - Rover): representa el número de suministros que está transportando el Rover.
  - (personal-Rover ?r - Rover): representa la cantidad de personal que está transportando el Rover.
  - (suministros-almacen ?b - almacen): representa el número de suministros que hay en un almacén.
  - (personal-asentamiento ?b - asentamiento): representa la cantidad de personal que hay en un asentamiento.
  - (peticiones-cerradas): representa el número de peticiones llevadas a cabo.
- **Acciones**
  - Mover: dado un Rover y dos bases, mueve el Rover de la primera base dada a la segunda, siempre que el Rover esté en la primera base y exista un camino entre ambas.
  - Cargar-suministros: dado un Rover y un almacén, traslada las existencias de suministro de las que disponga la base, siempre que esta disponga de alguno y que el Rover se encuentre en dicha base.

- Embarcar-personal: dado un Rover y un asentamiento, traslada todo el personal de la base al Rover, siempre que exista personal disponible en la base y el Rover se encuentre en el asentamiento.
- Satisfacer-peticion-suministros: dado un Rover, un asentamiento y una petición, da por acabada la petición trasladando el suministro del Rover a la base, siempre que el Rover se encuentre en la base, la petición esté abierta, sea de un suministro y sea para dicha base y el Rover tenga suministros suficientes.
- Satisfacer-peticion-personal: dado un Rover, un asentamiento y una petición, da por acabada la petición trasladando el personal del Rover a la base, siempre que el Rover se encuentre en la base, la petición esté abierta, sea para un personal y sea para dicha base y el Rover tenga personal suficiente.

2.2. Extensión 1

- **Variables**
  - Rovers: Representa al Rover.
  - Base: Se compone de Almacenes y Asentamientos que son subtipos de base
  - Petición: Representa el concepto de petición
- **Predicados**
  - (estacionado ?r - Rover ?b - base): representa la base en la que esta estacionado el Rover.
  - (camino ?b1 - base ?b2 - base): representa la existencia de un camino entre ambas bases.
  - (peticion-abierta ?p - peticion ?a - asentamiento): representa que asentamiento ha hecho la petición.
  - (peticion-suministros ?p - peticion): representa que la petición es para un suministro.
  - (peticion-personal ?p - peticion): representa que la petición es para un personal.
- **Funciones**
  - (suministros-Rover ?r - Rover): representa el número de suministros que está transportando el Rover.
  - (personal-Rover ?r - Rover): representa la cantidad de personal que está transportando el Rover.
  - (suministros-almacen ?b - almacen): representa el número de suministros que hay en un almacén.

- (personal-asentamiento ?b - asentamiento): representa la cantidad de personal que hay en un asentamiento.
- (peticiones-cerradas): representa el número de peticiones llevadas a cabo.

- **Acciones**
  - Mover: dado un Rover y dos bases, mueve el Rover de la primera base dada a la segunda, siempre que el Rover esté en la primera base y exista un camino entre ambas.
  - Cargar-suministros: dado un Rover y una base, traslada un suministro de los que disponga la base, siempre que esta disponga de alguno, que el Rover se encuentre en dicha base y que el Rover no tenga ni personal ni suministro.
  - Descargar-suministros: dado un Rover y una base, traslada el suministro que tenga el Rover a la base, siempre que el Rover disponga de un suministro y el Rover esté en dicha base.
  - Embarcar-personal: dado un Rover y una base, traslada una persona de la base al Rover, siempre que exista personal disponible en la base, que el Rover se encuentre en la base, que el Rover no tenga ningún suministro cargado o más de 1 persona a bordo.
  - Desembarcar-personal: dado un Rover y una base, traslada todo el personal del Rover a la base, siempre que exista personal disponible en el Rover y que el Rover se encuentre en la base.
  - Satisfacer-peticion-suministros: dado un Rover, un asentamiento y una petición, da por acabada la petición trasladando el suministro del Rover a la base, siempre que el Rover se encuentre en la base, la petición esté abierta, sea de un suministro y sea para dicha base y el Rover tenga suministros suficientes.
  - Satisfacer-peticion-personal: dado un Rover, un asentamiento y una petición, da por acabada la petición trasladando el personal del Rover a la base, siempre que el Rover se encuentre en la base, la petición esté abierta, sea para un personal y sea para dicha base y el Rover tenga personal suficiente.

2.3. Extensión 2

- **Variables**
  - Rovers: Representa al Rover.
  - Base: Se compone de Almacenes y Asentamientos que son subtipos de base
  - Petición: Representa el concepto de petición
- **Predicados**

- (estacionado ?r - Rover ?b - base): representa la base en la que esta estacionado el Rover.
- (camino ?b1 - base ?b2 - base): representa la existencia de un camino entre ambas bases.
- (peticion-abierta ?p - peticion ?a - asentamiento): representa que asentamiento ha hecho la petición.
- (peticion-suministros ?p - peticion): representa que la petición es para un suministro.
- (peticion-personal ?p - peticion): representa que la petición es para un personal.

- **Funciones**
  - (suministros-Rover ?r - Rover): representa el número de suministros que está transportando el Rover.
  - (personal-Rover ?r - Rover): representa la cantidad de personal que está transportando el Rover.
  - (suministros-almacen ?b - almacen): representa el número de suministros que hay en un almacén.
  - (personal-asentamiento ?b - asentamiento): representa la cantidad de personal que hay en un asentamiento.
  - (peticiones-cerradas): representa el número de peticiones llevadas a cabo.
  - (combustible ?r - rover): representa la cantidad de combustible que tiene el Rover.
  - (combustible-total): representa la cantidad de combustible total consumida.
- **Acciones**
  - Mover: dado un Rover y dos bases, mueve el Rover de la primera base dada a la segunda, siempre que el Rover esté en la primera base, exista un camino entre ambas y quede suficiente combustible para efectuar el movimiento.
  - Cargar-suministros: dado un Rover y una base, traslada un suministro de los que disponga la base, siempre que esta disponga de alguno, que el Rover se encuentre en dicha base y que el Rover no tenga ni personal ni suministro.
  - Descargar-suministros: dado un Rover y una base, traslada el suministro que tenga el Rover a la base, siempre que el Rover disponga de un suministro y el Rover esté en dicha base.
  - Embarcar-personal: dado un Rover y una base, traslada una persona de la base al Rover, siempre que exista personal disponible en la base, que el Rover se encuentre en la base, que el Rover no tenga ningún suministro cargado o más de 1 persona a bordo.

- Desembarcar-personal: dado un Rover y una base, traslada todo el personal del Rover a la base, siempre que exista personal disponible en el Rover y que el Rover se encuentre en la base.
- Satisfacer-peticion-suministros: dado un Rover, un asentamiento y una petición, da por acabada la petición trasladando el suministro del Rover a la base, siempre que el Rover se encuentre en la base, la petición esté abierta, sea de un suministro y sea para dicha base y el Rover tenga suministros suficientes.
- Satisfacer-peticion-personal: dado un Rover, un asentamiento y una petición, da por acabada la petición trasladando el personal del Rover a la base, siempre que el Rover se encuentre en la base, la petición esté abierta, sea para un personal y sea para dicha base y el Rover tenga personal suficiente.

2.4. Extensión 3

- **Variables**
  - Rovers: Representa al Rover.
  - Base: Se compone de Almacenes y Asentamientos que son subtipos de base
  - Petición: Representa el concepto de petición
- **Predicados**
  - (estacionado ?r - Rover ?b - base): representa la base en la que esta estacionado el Rover.
  - (camino ?b1 - base ?b2 - base): representa la existencia de un camino entre ambas bases.
  - (peticion-abierta ?p - peticion ?a - asentamiento): representa que asentamiento ha hecho la petición.
  - (peticion-suministros ?p - peticion): representa que la petición es para un suministro.
  - (peticion-personal ?p - peticion): representa que la petición es para un personal.
- **Funciones**
  - (suministros-Rover ?r - Rover): representa el número de suministros que está transportando el Rover.
  - (personal-Rover ?r - Rover): representa la cantidad de personal que está transportando el Rover.
  - (suministros-almacen ?b - almacen): representa el número de suministros que hay en un almacén.

- (personal-asentamiento ?b - asentamiento): representa la cantidad de personal que hay en un asentamiento.
- (peticiones-cerradas): representa el número de peticiones llevadas a cabo.
- (combustible ?r - rover): representa la cantidad de combustible que tiene el Rover.
- (combustible-total): representa la cantidad de combustible total consumida.
- (prioridad-peticion ?p - petición): representa la prioridad que tiene una petición.
- (prioridades-totales): representa la suma de las prioridades de las peticiones satisfechas.

- **Acciones**
  - Mover: dado un Rover y dos bases, mueve el Rover de la primera base dada a la segunda, siempre que el Rover esté en la primera base, exista un camino entre ambas y quede suficiente combustible para efectuar el movimiento.
  - Cargar-suministros: dado un Rover y una base, traslada un suministro de los que disponga la base, siempre que esta disponga de alguno, que el Rover se encuentre en dicha base y que el Rover no tenga ni personal ni suministro.
  - Descargar-suministros: dado un Rover y una base, traslada el suministro que tenga el Rover a la base, siempre que el Rover disponga de un suministro y el Rover esté en dicha base.
  - Embarcar-personal: dado un Rover y una base, traslada una persona de la base al Rover, siempre que exista personal disponible en la base, que el Rover se encuentre en la base, que el Rover no tenga ningún suministro cargado o más de 1 persona a bordo.
  - Desembarcar-personal: dado un Rover y una base, traslada todo el personal del Rover a la base, siempre que exista personal disponible en el Rover y que el Rover se encuentre en la base.
  - Satisfacer-peticion-suministros: dado un Rover, un asentamiento y una petición, da por acabada la petición trasladando el suministro del Rover a la base, siempre que el Rover se encuentre en la base, la petición esté abierta, sea de un suministro y sea para dicha base y el Rover tenga suministros suficientes.
  - Satisfacer-peticion-personal: dado un Rover, un asentamiento y una petición, da por acabada la petición trasladando el personal del Rover a la base, siempre que el Rover se encuentre en la base, la petición esté abierta, sea para un personal y sea para dicha base y el Rover tenga personal suficiente.

3. Modelado del problema

Para resolver los distintos problemas, proponemos las siguientes modelizaciones. Como se trata de una resolución incremental del problema, para cada extensión sólo describiremos qué cambia respecto a la anterior y qué elementos del problema utilizamos para resolver dicha extensión.

3.1. Nivel básico

- **Objetos**
  - **Rovers:** representa el medio de transporte de los diferentes suministros y del personal especializado. De momento tienen capacidad ilimitada.
  - **Petición:** representan las diferentes peticiones de suministros de los almacenes. Cada almacén puede hacer tantas peticiones como quiera.
  - **Almacén:** representa el sitio donde se guardan los suministros y que se encarga de hacer las peticiones de estos.
  - **Base:** representan los diferentes asentamientos donde se aloja el personal especializado. Cada base se encarga de hacer todas las peticiones que necesite.
  - **Persona:** representa a las personas especializadas. Viajan entre las bases según las peticiones de estos.
- **Estado inicial**
  - El estado inicial tiene todos los rovers definidos en objects aparcados en alguna base (ya sea base o almacén).
  - Las personas van a estar en asentamientos mientras que los suministros se encontrarán en almacenes.
  - Las cargas que van a transportar los rovers, ya sean suministros o personas.
  - Se encuentran definidas las diferentes peticiones abiertas que hace cada base o almacén ya sea de personal o suministros.
- **Estado final**

```
(:goal
    (= (peticiones-cerradas) N)
)
```

  - Estas dos lineas hacen que se cumpla el estado final en caso de ser posible, básicamente queremos que **todas** las peticiones hechas por los diferentes almacenes y bases esten cerradas

3.2. Extensión 1

Para esta extensión se nos pedía que el número de personal a bordo de un Rover sea un máximo de 2 y 1 en el caso de los suministros, además, de la restricción de no poder llevar personal y suministros juntos. El único cambio realizado en esta extensión ha sido el haber añadido los siguientes casos:

- **En caso de cargar suministros:** que el Rover no tenga ni personal ni suministros a bordo.
- **En caso de embarcar personal:** que el Rover no tenga suministros y que tampoco contenga 2 personas a bordo.

Un pequeño cambio hecho ha sido qué en vez de usar los subtipos de base, almacén y asentamiento, en las operaciones embarcar-personal y cargar-suministros, se han cambiado por el tipo base, de esta manera podemos cargar y descargar tanto personal como suministros tantas veces como se necesite.

En resumen lo único que cambia del modelado es el estado inicial.

- **Estado inicial**
  - Ahora además de tener los rovers inicializados, tenemos que inicializar las variables de `suministros-rover` y `personal-rover` a 0 debido a que al principio el rover no lleva ni personal ni suministros.

3.3. Extensión 2

Para esta extensión se nos pedía añadir una capacidad limitada de combustible al Rover, por lo que limita el número de movimientos, también, se nos pedía la posibilidad de minimizar la cantidad de combustible usado. Los cambios realizados han sido los de al mover restar una unidad de combustible e ir añadiendo 1 unidad al combustible total gastado. Además, que en las restricciones no sea posible mover al Rover si ese ya no tiene más combustible. Por otra parte, hemos añadido la siguiente métrica para poder minimizar la cantidad de combustible usado:

```
(:metric minimize (combustible-total))
```

En resumen lo único que cambia del modelado es el estado inicial.

- **Estado inicial**
  - Ahora además de lo mismo de antes tenemos que inicializar el `combustible` de cada rover que determinará el número de movimientos que este puede hacer.
  - También debemos inicializar el `combustible-total` a 0 para poder minimizarlo.

3.4. Extensión 3

Para esta última extensión se nos pedía añadir una prioridad a las peticiones, de manera que se priorizaran las de más alta prioridad. A la hora de implementar la suma de las prioridades de las peticiones realizadas, se ha hecho de manera inversa, es decir, las peticiones de prioridad 3, se le suma 1 al total de prioridades satisfechas y a las peticiones de prioridad 1 se les suma 3 al total de prioridades satisfechas. De esta manera minimizamos siempre, tanto al añadir las ponderaciones con el combustible como cuando se obvia este.

Por otra parte, hemos añadido la siguiente métrica para poder minimizar la cantidad de combustible usado:

```
(:metric minimize (+
   (* (prioridades-totales) N)
    (* (combustible-total) M)
))
```

Dónde M y N son las ponderaciones que añadimos para el combustible y las prioridades, respectivamente.

En resumen lo único que cambia del modelado es el estado inicial.

- **Estado inicial**
  - Ahora además de lo mismo de antes tenemos que inicializar la `(prioridad-peticion ?p - petición)` que indica la prioridad que tiene cada petición.
  - También debemos inicializar la variable de `prioridades-totales` a 0 para poder minimizarlo, esta indica la suma de las prioridades de las peticiones satisfechas.

4. Desarrollo de los modelos

Para el desarrollo de los distintos hemos seguido la metodología explicada en el enunciado de ir haciendo extensiones incrementalmente empezando por un nivel muy básico. En cada extensión se han ido añadiendo diferentes funcionalidades que nos permitían cumplir con lo que se pedía en el enunciado.

Para el testeo de los diferentes modelos se han ido haciendo problemas específicos que enseñaban el buen funcionamiento de estos y más tarde se ha hecho un generador de problemas aleatorio para cada modelo para poder estudiar los diferentes comportamientos y tiempos de ejecución.

5. Juegos de prueba

Para comprobar el correcto funcionamiento del modelo inicial y sus diferentes extensiones hemos realizado los siguientes juegos de prueba que se han generado aleatoriamente mediante el `generator.py` que hemos desarrollado.

5.1. Juego 1 - Nivel básico

En este primer juego de pruebas generado aleatoriamente con la `SEED=1234` con 1 rover, 5 bases y 1 petición. Este problema se llama `probBase.pddl`.

- **Entrada**

```
(define (problem probBase) (:domain peticionador)
  (:objects
      r0 - rover
      b1 b2 - almacen
      b0 b3 b4 - asentamiento
      p0 - peticion
  )
  (:init
      (= (peticiones-cerradas) 0)
      (camino b0 b1)
      (camino b0 b2)
      (camino b1 b4)
      (camino b2 b3)
      (camino b2 b4)
      (= (suministros-almacen b1) 0)
      (= (suministros-almacen b2) 0)
      (= (personal-asentamiento b0) 1)
      (= (personal-asentamiento b3) 0)
      (= (personal-asentamiento b4) 0)
      (= (suministros-rover r0) 0) (= (personal-rover r0) 0)
          (estacionado r0 b2)
      (peticion-abierta p0 b0) (peticion-personal p0)
  )
  (:goal
      (= (peticiones-cerradas) 1)
  )
  )
```

- **Salida**

```
  step
      0: MOVER R0 B2 B0
      1: EMBARCAR-PERSONAL R0 B0
      2: SATISFACER-PETICION-PERSONAL R0 B0 P0
```

- **Justificación**

  Este juego de pruebas muestra como un problema con una sola petición se cumple, que es el rover moviendo al personal hasta la base 0.

5.2. Juego 2 - Extensión 1

En este segundo juego de pruebas generado aleatoriamente con la `SEED=200` con 4 rover, 7 bases y 6 petición. Este problema se llama `probExt1.pddl`.

- **Entrada**

```
(define (problem probExt1) (:domain peticionador)
    (:objects
        r0 r1 r2 r3 - rover
        b0 b1 b3 b5 b6 - almacen
        b2 b4 - asentamiento
        p0 p1 p2 p3 p4 p5 - peticion
    )
    (:init
        (= (peticiones-cerradas) 0)
        (camino b0 b6)
        (camino b0 b5)
        (camino b1 b2)
        (camino b2 b6)
        (camino b3 b4)
        (camino b4 b5)
        (camino b5 b6)
        (= (suministros-base b0) 1)
        (= (suministros-base b1) 0)
        (= (suministros-base b3) 0)
        (= (suministros-base b5) 0)
        (= (suministros-base b6) 0)
        (= (personal-base b2) 0)
        (= (personal-base b4) 1)
        (= (suministros-rover r0) 0) (= (personal-rover r0) 0)
            (estacionado r0 b2)
        (= (suministros-rover r1) 0) (= (personal-rover r1) 0)
            (estacionado r1 b3)
        (= (suministros-rover r2) 0) (= (personal-rover r2) 0)
            (estacionado r2 b3)
        (= (suministros-rover r3) 0) (= (personal-rover r3) 0)
            (estacionado r3 b1)
        (peticion-abierta p0 b2) (peticion-suministros p0)
        (peticion-abierta p1 b2) (peticion-suministros p1)
        (peticion-abierta p2 b4) (peticion-personal p2)
```

```
            (peticion-abierta p3 b4) (peticion-personal p3)
            (peticion-abierta p4 b4) (peticion-personal p4)
            (peticion-abierta p5 b2) (peticion-personal p5)
        )
        (:goal
            (= (peticiones-cerradas) 2)
        )
    )
```

- **Salida**

```
step
 0: MOVER R2 B3 B4
 1: EMBARCAR-PERSONAL R2 B4
 2: SATISFACER-PETICION-PERSONAL R2 B4 P2
 3: MOVER R2 B4 B5
 4: MOVER R2 B5 B0
 5: CARGAR-SUMINISTROS R2 B0
 6: MOVER R2 B0 B6
 7: MOVER R2 B6 B2
 8: SATISFACER-PETICION-SUMINISTROS R2 B2 P0
```

- **Justificación**

  Este juego de pruebas muestra como un problema con dos peticiones se cumple, que es el rover moviendo al personal hasta la base 4 y el personal hasta la base 2.

5.3. Juego 3 - Extensión 2 (sin optimizaciones)

En este segundo juego de pruebas generado aleatoriamente con la `SEED=754` con 4 rover, 7 bases y 6 petición. Este problema se llama `probExt2.pddl` sin optimizar el combustible usado.

- **Entrada**

```
(define (problem probExt2) (:domain peticionador)
 (:objects
     r0 r1 r2 r3 - rover
     b0 b1 b2 b4 b5 b6 - almacen
     b3 - asentamiento
     p0 p1 p2 p3 p4 p5 - peticion
 )
 (:init
     (= (combustible-total) 0) (= (peticiones-cerradas) 0)
     (camino b0 b3)
     (camino b1 b2)
     (camino b1 b4)
     (camino b2 b3)
```

```
        (camino b3 b5)
        (camino b4 b5)
        (camino b5 b6)
        (= (suministros-base b0) 2)
        (= (suministros-base b1) 0)
        (= (suministros-base b2) 0)
        (= (suministros-base b4) 0)
        (= (suministros-base b5) 0)
        (= (suministros-base b6) 0)
        (= (personal-base b3) 0)
        (= (suministros-rover r0) 0) (= (personal-rover r0) 0)
            (= (combustible r0) 21) (estacionado r0 b5)
        (= (suministros-rover r1) 0) (= (personal-rover r1) 0)
            (= (combustible r1) 21) (estacionado r1 b3)
        (= (suministros-rover r2) 0) (= (personal-rover r2) 0)
            (= (combustible r2) 21) (estacionado r2 b0)
        (= (suministros-rover r3) 0) (= (personal-rover r3) 0)
            (= (combustible r3) 21) (estacionado r3 b3)
        (peticion-abierta p0 b3) (peticion-suministros p0)
        (peticion-abierta p1 b3) (peticion-suministros p1)
        (peticion-abierta p2 b3) (peticion-suministros p2)
        (peticion-abierta p3 b3) (peticion-suministros p3)
        (peticion-abierta p4 b3) (peticion-suministros p4)
        (peticion-abierta p5 b3) (peticion-suministros p5)
    )
    (:goal
        (= (peticiones-cerradas) 2)
    ))
    ;(:metric minimize (combustible-total)))
```

- **Salida**

```
step
 0: CARGAR-SUMINISTROS R2 B0
 1: MOVER R2 B0 B3
 2: SATISFACER-PETICION-SUMINISTROS R2 B3 P5
 3: MOVER R2 B3 B0
 4: CARGAR-SUMINISTROS R2 B0
 5: MOVER R2 B0 B3
 6: SATISFACER-PETICION-SUMINISTROS R2 B3 P4
```

- **Justificación**

  Este juego de pruebas muestra como un problema con dos peticiones se cumple, teniendo en cuenta el combustible usado pero sin minimizarlo, como veremos en el ejemplo a posteriori.

5.4. Juego 4 - Extensión 2 (con optimizaciones)

En este segundo juego de pruebas generado aleatoriamente con la `SEED=754` con 4 rover, 7 bases y 6 petición. Este problema se llama `probExt20.pddl`.

- **Entrada**

```
(define (problem probExt20) (:domain peticionador)
    (:objects
        r0 r1 r2 r3 - rover
        b0 b1 b2 b4 b5 b6 - almacen
        b3 - asentamiento
        p0 p1 p2 p3 p4 p5 - peticion
    )
    (:init
        (= (combustible-total) 0) (= (peticiones-cerradas) 0)
        (camino b0 b3)
        (camino b1 b2)
        (camino b1 b4)
        (camino b2 b3)
        (camino b3 b5)
        (camino b4 b5)
        (camino b5 b6)
        (= (suministros-base b0) 2)
        (= (suministros-base b1) 0)
        (= (suministros-base b2) 0)
        (= (suministros-base b4) 0)
        (= (suministros-base b5) 0)
        (= (suministros-base b6) 0)
        (= (personal-base b3) 0)
        (= (suministros-rover r0) 0) (= (personal-rover r0) 0)
            (= (combustible r0) 21) (estacionado r0 b5)
        (= (suministros-rover r1) 0) (= (personal-rover r1) 0)
            (= (combustible r1) 21) (estacionado r1 b3)
        (= (suministros-rover r2) 0) (= (personal-rover r2) 0)
            (= (combustible r2) 21) (estacionado r2 b0)
        (= (suministros-rover r3) 0) (= (personal-rover r3) 0)
            (= (combustible r3) 21) (estacionado r3 b3)
        (peticion-abierta p0 b3) (peticion-suministros p0)
        (peticion-abierta p1 b3) (peticion-suministros p1)
        (peticion-abierta p2 b3) (peticion-suministros p2)
        (peticion-abierta p3 b3) (peticion-suministros p3)
        (peticion-abierta p4 b3) (peticion-suministros p4)
        (peticion-abierta p5 b3) (peticion-suministros p5)
    )
    (:goal
        (= (peticiones-cerradas) 2)
```

```
    )
    (:metric minimize (combustible-total)))
```

- **Salida**

```
step
 0: CARGAR-SUMINISTROS R2 B0
 1: MOVER R2 B0 B3
 2: MOVER R3 B3 B0
 3: CARGAR-SUMINISTROS R3 B0
 4: MOVER R3 B0 B3
 5: SATISFACER-PETICION-SUMINISTROS R2 B3 P0
 6: SATISFACER-PETICION-SUMINISTROS R3 B3 P1
```

- **Justificación**

  Este juego de pruebas muestra como un problema con dos peticiones se cumple, teniendo en cuenta el combustible usado pero esta vez minimizandolo.

5.5. Juego 5 - Extensión 3 (sin optimizaciones)

En este segundo juego de pruebas generado aleatoriamente con la `SEED=10` con 4 rover, 7 bases y 6 petición. Este problema se llama `probExt3.pddl`.

- **Entrada**

```
(define (problem probExt3) (:domain peticionador)
 (:objects
     r0 r1 r2 r3 - rover
     b1 b3 - almacen
     b0 b2 b4 b5 b6 - asentamiento
     p0 p1 p2 p3 p4 p5 - peticion
 )
 (:init
     (= (prioridades-totales) 0) (= (combustible-total) 0)
         (= (peticiones-cerradas) 0)
     (camino b0 b1)
     (camino b1 b4)
     (camino b1 b6)
     (camino b2 b6)
     (camino b2 b3)
     (camino b3 b5)
     (camino b4 b5)
     (= (suministros-base b1) 0)
     (= (suministros-base b3) 0)
     (= (suministros-base b0) 0)
     (= (suministros-base b2) 0)
     (= (suministros-base b4) 0)
```

```
    (= (suministros-base b5) 0)
    (= (suministros-base b6) 0)
    (= (personal-base b0) 0)
    (= (personal-base b2) 1)
    (= (personal-base b4) 1)
    (= (personal-base b5) 0)
    (= (personal-base b6) 0)
    (= (personal-base b1) 0)
    (= (personal-base b3) 0)
    (= (suministros-rover r0) 0) (= (personal-rover r0) 0)
        (= (combustible r0) 21) (estacionado r0 b5)
    (= (suministros-rover r1) 0)
        (= (personal-rover r1) 0) (= (combustible r1) 21)
        (estacionado r1 b1)
    (= (suministros-rover r2) 0) (= (personal-rover r2) 0)
        (= (combustible r2) 21) (estacionado r2 b3)
    (= (suministros-rover r3) 0) (= (personal-rover r3) 0)
        (= (combustible r3) 21) (estacionado r3 b5)
    (peticion-abierta p0 b0) (peticion-personal p0)
        (= (prioridad-peticion p0) 2)
    (peticion-abierta p1 b2) (peticion-personal p1)
        (= (prioridad-peticion p1) 3)
    (peticion-abierta p2 b4) (peticion-personal p2)
        (= (prioridad-peticion p2) 2)
    (peticion-abierta p3 b5) (peticion-personal p3)
        (= (prioridad-peticion p3) 2)
    (peticion-abierta p4 b4) (peticion-personal p4)
        (= (prioridad-peticion p4) 2)
    (peticion-abierta p5 b2) (peticion-personal p5)
        (= (prioridad-peticion p5) 3)
)
(:goal
    (= (peticiones-cerradas) 2)
)
;(:metric minimize (+
 ;   (* (prioridades-totales) 2)
  ;  (* (combustible-total) 3)
;)))
)
```

- **Salida**

```
step
    0: MOVER R3 B5 B3
    1: MOVER R3 B3 B2
    2: EMBARCAR-PERSONAL R3 B2
    3: SATISFACER-PETICION-PERSONAL R3 B2 P5
```

```
4: MOVER R0 B5 B4
5: EMBARCAR-PERSONAL R0 B4
6: SATISFACER-PETICION-PERSONAL R0 B4 P4
```

- **Justificación**

  En este juego de pruebas, a parte de probar la extensión 3 optimizada, se comprueba que teniendo solicitudes prioritarias.

5.6. Juego 6 - Extensión 3 (con optimizaciones)

En este segundo juego de pruebas generado aleatoriamente con la `SEED=10` con 4 rover, 7 bases y 6 petición. Este problema se llama `probExt30.pddl`.

- **Entrada**

```
            (define (problem probExt30) (:domain peticionador)
(:objects
    r0 r1 r2 r3 - rover
    b1 b3 - almacen
    b0 b2 b4 b5 b6 - asentamiento
    p0 p1 p2 p3 p4 p5 - peticion
)
(:init
    (= (prioridades-totales) 0) (= (combustible-total) 0) (= (peticiones-cerradas
    (camino b0 b1)
    (camino b1 b4)
    (camino b1 b6)
    (camino b2 b6)
    (camino b2 b3)
    (camino b3 b5)
    (camino b4 b5)
    (= (suministros-base b1) 0)
    (= (suministros-base b3) 0)
    (= (suministros-base b0) 0)
    (= (suministros-base b2) 0)
    (= (suministros-base b4) 0)
    (= (suministros-base b5) 0)
    (= (suministros-base b6) 0)
    (= (personal-base b0) 0)
    (= (personal-base b2) 1)
    (= (personal-base b4) 1)
    (= (personal-base b5) 0)
    (= (personal-base b6) 0)
    (= (personal-base b1) 0)
    (= (personal-base b3) 0)
```

```
    (= (suministros-rover r0) 0) (= (personal-rover r0) 0)
        (= (combustible r0) 21) (estacionado r0 b5)
    (= (suministros-rover r1) 0) (= (personal-rover r1) 0)
        (= (combustible r1) 21) (estacionado r1 b1)
    (= (suministros-rover r2) 0) (= (personal-rover r2) 0)
        (= (combustible r2) 21) (estacionado r2 b3)
    (= (suministros-rover r3) 0) (= (personal-rover r3) 0)
        (= (combustible r3) 21) (estacionado r3 b5)
    (peticion-abierta p0 b0) (peticion-personal p0)
        (= (prioridad-peticion p0) 2)
    (peticion-abierta p1 b2) (peticion-personal p1)
        (= (prioridad-peticion p1) 3)
    (peticion-abierta p2 b4) (peticion-personal p2)
        (= (prioridad-peticion p2) 2)
    (peticion-abierta p3 b5) (peticion-personal p3)
        (= (prioridad-peticion p3) 2)
    (peticion-abierta p4 b4) (peticion-personal p4)
        (= (prioridad-peticion p4) 2)
    (peticion-abierta p5 b2) (peticion-personal p5)
        (= (prioridad-peticion p5) 3)
)
(:goal
    (= (peticiones-cerradas) 2)
)
(:metric minimize (+
    (* (prioridades-totales) 2)
    (* (combustible-total) 3)
)))
```

- **Salida**

```
step
 0: MOVER R0 B5 B4
 1: EMBARCAR-PERSONAL R0 B4
 2: MOVER R2 B3 B2
 3: EMBARCAR-PERSONAL R2 B2
 4: SATISFACER-PETICION-PERSONAL R0 B4 P2
 5: SATISFACER-PETICION-PERSONAL R2 B2 P1
```

- **Justificación**

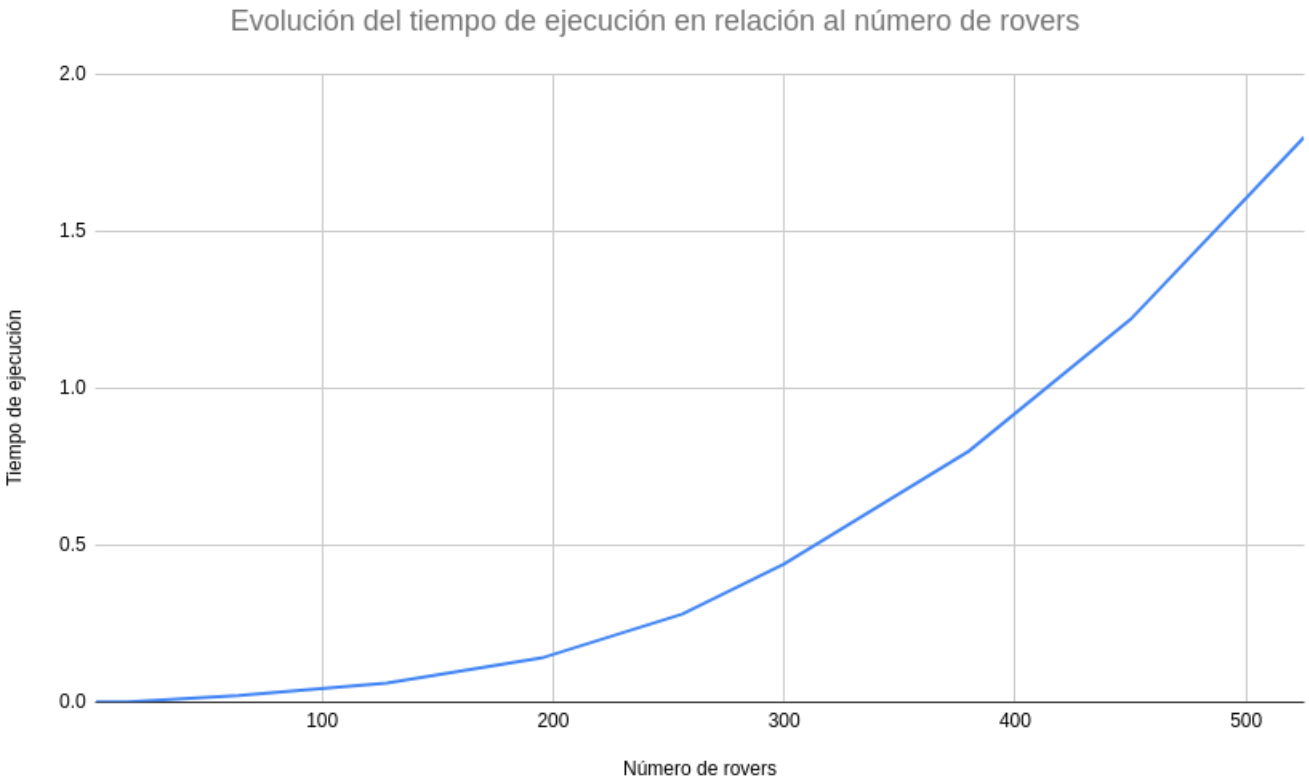
  En este juego de pruebas, a parte de probar la extensión 3 optimizada, se comprueba que teniendo solicitudes prioritarias, se cambian los ordenes lo mínimo posible, minimizando siempre el uso del combustible y las prioridades totales.

6. Tiempo de Ejecución

En este apartado nos centraremos en analizar la evolución del tiempo de ejecución. Para poder realizar este análisis hemos hecho pruebas con los 4 parámetros principales: `número de rovers, número de peticiones, número de personal/suministros(load) y el número de bases`. Es decir, para la prueba de un parámetro, los otros 3 se mantenían en valores fijos. Al acabar estas pruebas, ejecutabamos el programa resultante con el `domain.pddl` para obtener así el tiempo de ejecuión. En nuestro caso, para poder realizar las pruebas coómodamente, hemos trabajado con la `Extensión 1`.
Una vez acabamos de probar los 4 parámetros y ejecutamos sus pruebas, llegamos a la conclusión que el único parámetro que mostraba tener una evolución importante era ell número de rovers. Es decir el crecimiento del número de rovers y del tiempo de ejecución es proporcional. Es por esto que en este apartado nos centraremos en este parámetro.
A continuación se muestra una tabla que sintetiza los tiempos de ejecución obtenidos con el número de rovers usados:

Número de rovers	Tiempo de ejecución
2	0.00
16	0.00
64	0.02
128	0.06
195	0.14
256	0.28
300	0.44
380	0.80
450	1.22
525	1.80

Hemos creado un gráfico con estos resultados para poder visualizarlos mejor:



Mientras ejecutabamos estos programas, vimos que debido a las limitaciones del `Metric-FF`, los valores más altos de número de rovers no se podían ejecutar correctamente. Es por eso que tuvimos que modificar parámetros del fichero *ff.h* (concretamente`MAX_LENGTH` y `MAX_RELEVANT_FLUENTS`). Tal y como se observa en los resultados, la ejecución tiene valores muy pequeños, práctimente nulos, para valores pequeños y sigue un crecimiento gradual.
También se observa que la mayor parte del tiempo se emplea en la computación del algorítmo Left Neighbor First y en la búsqueda de la solución.
Todos los archivos de prueba creados con el `generator.py` (programado en Python) se incluyen en una carpeta dentro de la extensión 1.

7. Conclusiones

Con esta práctica, hemos observado como los sistemas basados en conocimiento y los planificadores tienen una inherente relación entre ellos. Los dos se sujetan a las bases de la lógica, diferenciándose por la autonomía que tiene uno respeto al otro, dado que los planificadores se ayudan de los heurísticos para llegar a una buena solución.

Es importante recalcar que el modelado de un problema y su dominio tiene un gran impacto en su resolución, influyendo des de la solución hasta el tiempo de ejecución. También hemos observado una importante impacto de la complejidad del problema y el tiempo de ejecución, sobretodo al analizar el crecimiento entre el tiempo de ejecución y el número de rovers utilizados.

Esta última práctica nos ha permitido conocer y usar un nuevo lenguaje de programación, como también un nuevo método de resolución de problemas. Con cada extensión hemos podido explorar nuevas complejidades y sacar nuestras propias conclusiones de ellas.