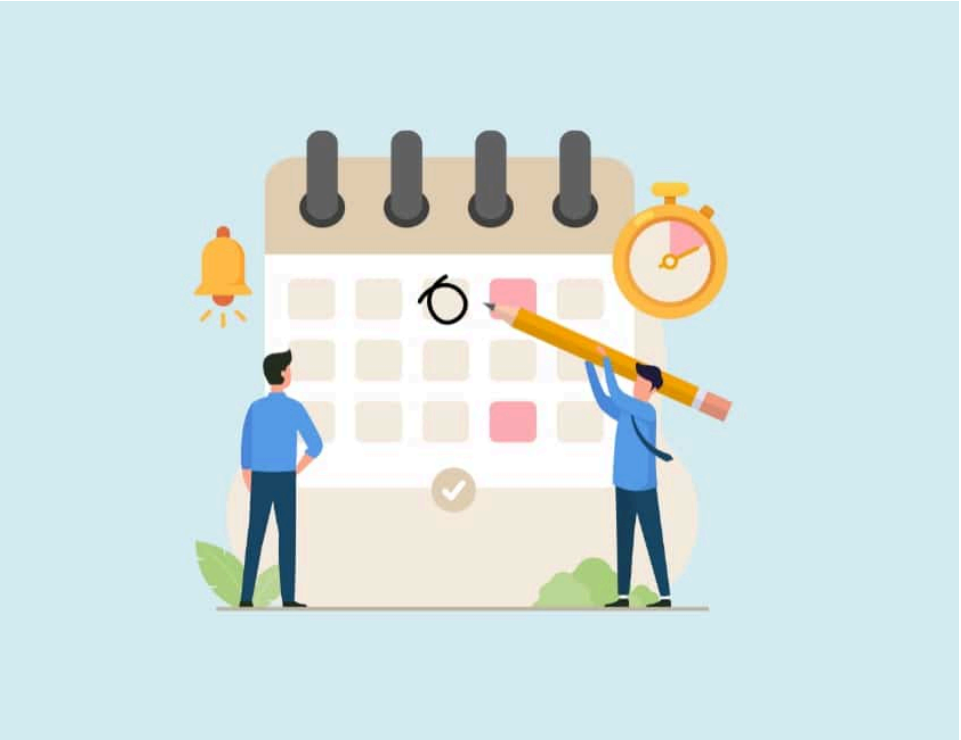
Práctica de Planificación

Hoteles y Reservas



Ishak Felfoul
Eduard Corrons
LAB12 - 2025/2026 Q1

ÍNDICE

1. Introducción

En esta práctica se ha desarrollado un planificador de reservas de hotel utilizando el lenguaje PDDL y el planificador Metric-FF. El objetivo es asignar un conjunto de reservas a distintas habitaciones respetando las restricciones del problema y, en las extensiones más avanzadas, optimizando además diferentes criterios de calidad del plan.

El dominio modela elementos típicos de un sistema de gestión de reservas: habitaciones con cierta capacidad, reservas con número de personas y rango de días, posibles solapamientos en el calendario y preferencias adicionales como la orientación de la habitación. A partir de esta modelización, el planificador debe decidir para cada reserva si se asigna a alguna habitación o se descarta, teniendo en cuenta tanto la factibilidad como los costes asociados a cada decisión.

La práctica se ha planteado de forma incremental en varios niveles. En el nivel básico sólo se exige construir un plan que asigne correctamente las reservas disponibles, mientras que en las extensiones se añaden nuevas restricciones y funciones de coste, como la orientación preferida, el desperdicio de plazas o el número de habitaciones utilizadas. Finalmente, en la última extensión se define una métrica que combina todos estos aspectos, de forma que el planificador busque no solo una solución válida, sino también una solución “buena” según los criterios establecidos.

2. Dominio

2.1 Estado inicial y final

En todos los niveles de la práctica el estado final tiene la misma forma general: para cada reserva del problema se exige que haya sido tratada, es decir, que esté asignada a alguna habitación o bien descartada explícitamente mediante la acción correspondiente. Esto se expresa mediante un objetivo de la forma (forall (?r - reserva) (or (asignada ?r) (descartada ?r))), que garantiza que no queda ninguna reserva sin procesar al final del plan. En el nivel básico esta condición se particulariza pidiendo que todas las reservas estén asignadas, mientras que en las extensiones métricas se permite también que algunas reservas terminen descartadas si así lo decide el planificador para optimizar el coste total.

En cambio, el estado inicial varía según el nivel y la extensión considerada. En el nivel básico se inicializan las habitaciones con su capacidad máxima y las reservas con su número de personas y el intervalo de días en que se alojan, además de los predicados que indican qué días ocupa cada reserva; ninguna reserva aparece inicialmente ni como asignada ni como descartada, y no se marca ninguna habitación como ocupada, lo que implica que todas están libres al inicio. En las extensiones se va enriqueciendo progresivamente este estado inicial: se añaden las preferencias de orientación de cada reserva y la orientación real de cada habitación, así como los valores iniciales (a 0) de las distintas funciones de coste introducidas en las versiones métricas, tales como coste-descartar, coste-orientacion, coste-desperdicio y coste-habitaciones.

Extensión 1: El estado inicial es básicamente el del nivel básico, pero el objetivo ya no exige que todas las reservas estén asignadas, sino que se introduce una función numérica que contabiliza las reservas asignadas y se busca maximizarla (o minimizar un coste equivalente).

Extensión 2: Al estado inicial se añaden las preferencias de orientación de cada reserva y la orientación de cada habitación, junto con la función de coste asociada a las orientaciones satisfechas/no satisfechas; el objetivo combina ahora el número de reservas asignadas con el cumplimiento de las orientaciones, penalizando más no asignar que asignar con orientación incorrecta.

Extensión 3: Se incorpora información sobre la capacidad de las habitaciones en relación con cada reserva mediante una función que mide el desperdicio de plazas; el objetivo combina el criterio de maximizar reservas asignadas con la minimización de este desperdicio, ponderando adecuadamente ambos aspectos.

Extensión 4: Se añade al estado inicial la función que cuenta cuántas habitaciones diferentes se utilizan, y el objetivo pasa a integrar tres criterios (reservas asignadas, habitaciones abiertas y desperdicio de plazas), priorizando que no queden reservas sin asignar y que se use el menor número de habitaciones posible frente a minimizar el desperdicio.

2.1 Variables

En el dominio del hotel se han definido cuatro tipos principales de objetos, que se mantienen constantes en todas las versiones de la práctica: reserva, habitacion, dia y direccion. Sobre estos tipos se construyen los predicados y funciones que permiten representar el estado del sistema y definir los objetivos y costes.

reserva
Este tipo representa cada petición de reserva realizada por un cliente. De cada reserva se almacena:

- El número de personas mediante la función num-personas.
- El intervalo de estancia, representado tanto por las funciones dia-inicio y dia-fin como por los hechos dia-de-reserva que indican explícitamente los días en que la reserva está activa.
- A partir de las extensiones con preferencias, una orientación deseada mediante el predicado preferencia, que se usará para penalizar asignaciones que no respeten dicha orientación.

habitacion
Modela cada habitación disponible en el hotel. Para cada habitación se registra:

- Su capacidad máxima con la función capacidad, que limita el número de personas que puede alojar.
- Su orientación física mediante el predicado orientacion, utilizado al comparar con las preferencias de las reservas en las extensiones.
- En la extensión 4, si la habitación ya ha sido utilizada alguna vez mediante el predicado habitacion-usada, lo que permite contabilizar cuántas habitaciones distintas se abren mediante la función de coste asociada.

dia
Representa los días considerados en la planificación. Se utilizan para:

- Indicar los días en que está activa una reserva mediante el predicado dia-de-reserva.
- Marcar los días en que una habitación está ocupada con el predicado ocupada, lo que permite controlar correctamente los solapamientos entre reservas en una misma habitación.

direccion
Este tipo recoge las posibles orientaciones (por ejemplo n, s, e, o). Interviene en las extensiones donde se tiene en cuenta la preferencia de orientación:

- En orientacion ?h ?dir para codificar la orientación real de cada habitación.
- En preferencia ?r ?dir para expresar la orientación deseada por cada reserva.

Estos tipos de objetos declarados en la sección :objects se han mantenido iguales en todas las extensiones, y lo que ha ido evolucionando entre versiones son los predicados y funciones definidos sobre ellos, así como los criterios de coste que se optimizan mediante la métrica del problema.

2.2 Predicados

En el dominio del hotel se han definido los siguientes predicados, algunos presentes desde el nivel básico y otros introducidos en las extensiones.

asignada ?r - reserva
Indica que la reserva ?r ya ha sido asignada a alguna habitación. Pasa a ser cierto en el efecto de la acción asignar-reserva y se utiliza en el objetivo final para comprobar que todas las reservas han sido tratadas, así como en las precondiciones para evitar asignar dos veces la misma reserva.

descartada ?r - reserva
Indica que se ha decidido no asignar la reserva ?r a ninguna habitación. Aparece a partir de la extensión 1 donde se permite dejar reservas sin asignar, se hace cierto mediante la acción descartar-reserva y, junto con asignada, forma parte de la condición de estado final (forall (?r - reserva) (or (asignada ?r) (descartada ?r))).

ocupada ?h - habitacion ?d - dia
Indica que la habitación ?h está ocupada el día ?d. Se actualiza en el efecto de asignar-reserva para todos los días activos de la reserva y se comprueba en la precondición de esa misma acción, garantizando que no se asignen dos reservas solapadas a la misma habitación.

dia-de-reserva ?r - reserva ?d - dia
Marca explícitamente los días en los que la reserva ?r está activa. Se declara en el estado inicial del problema y se usa tanto en la precondición (para comprobar disponibilidad) como en el efecto de asignar-reserva (para saber qué días hay que marcar la habitación como ocupada).

orientacion ?h - habitacion ?dir - direccion
Indica la orientación física de la habitación ?h (por ejemplo norte, sur, este u oeste). Se utiliza en las extensiones con preferencias de orientación, donde se compara con preferencia para determinar si una asignación respeta la orientación deseada y, en caso contrario, aplicar la penalización correspondiente en el coste.Añadidos en la extensión 2, donde se modelan preferencias de orientación

preferencia ?r - reserva ?dir - direccion
Expresa la orientación deseada por la reserva ?r. En combinación con orientacion, permite comprobar dentro de la acción asignar-reserva si existe alguna dirección que satisfaga la preferencia de la reserva; si no la hay, se incrementa coste-orientacion para reflejar que se ha asignado a una habitación con orientación no preferida. Añadidos en la extensión 2, donde se modelan preferencias de orientación

habitacion-usada ?h - habitacion
Señala que la habitación ?h ha sido utilizada al menos una vez en el plan. Se introduce en la extensión 4, se inicializa a falso en el estado inicial y pasa a cierto la primera vez que se asigna alguna reserva a esa habitación, mediante un when (not (habitacion-usada ?h)) ... en el efecto de asignar-reserva, lo que permite contar cuántas habitaciones distintas se han abierto a través de la función de coste coste-habitaciones.

2.3 Funciones

El dominio utiliza varias funciones numéricas (fluents) que se han ido introduciendo progresivamente a lo largo de los niveles de la práctica.
Funciones básicas de capacidad y tamaño de reserva
Estas funciones están presentes desde el nivel básico, ya que son necesarias para expresar las restricciones de capacidad y calendario.

- num-personas ?r - reserva
  Indica el número de personas que forman la reserva ?r. Se usa en las precondiciones para comprobar que la reserva cabe en la habitación elegida y, en extensiones posteriores, para calcular el desperdicio de plazas en combinación con capacidad.

- capacidad ?h - habitacion
  Almacena el número máximo de personas que puede alojar la habitación ?h. Permite restringir las asignaciones y sirve de base para medir el desperdicio de plazas cuando se introduce la función coste-desperdicio.

Funciones relacionadas con el intervalo de días
- dia-inicio ?r - reserva

- dia-fin ?r - reserva
  Guardan el primer y último día de estancia de la reserva ?r. En las instancias se complementan con los hechos dia-de-reserva para razonar día a día, mientras que estas funciones describen el intervalo de manera más compacta.

Funciones globales de coste
A partir de las extensiones de métricas se añaden funciones globales de coste acumulado, inicializadas a cero en el estado inicial y actualizadas en los efectos de las acciones:

- **coste-descartar**
  Acumula una penalización fija por cada reserva descartada mediante la acción descartar-reserva. Refleja que es costoso dejar reservas sin asignar respecto a asignarlas, aunque sea en condiciones no ideales.

- **coste-orientacion**
  Cuenta las veces que se asigna una reserva a una habitación cuya orientación no coincide con la preferencia indicada por el cliente. Se incrementa condicionalmente en asignar-reserva cuando no existe ninguna dirección común entre preferencia y orientacion para esa reserva y habitación.

- **coste-desperdicio**
  Suma, para cada asignación, la diferencia entre la capacidad de la habitación y el número de personas de la reserva, es decir, las plazas desaprovechadas. Materializa el criterio de “minimizar el desperdicio de plazas” de las extensiones avanzadas

- **coste-habitaciones**
  Cuenta cuántas habitaciones distintas se han utilizado en el plan. Solo se incrementa la primera vez que una habitación pasa a estar marcada como habitacion-usada, lo que permite optimizar el número total de habitaciones abiertas.

La combinación lineal de estas funciones de coste aparece en la cláusula (:metric minimize ...), que permite a MetricFF comparar planes y escoger aquel que mejor equilibra los criterios de la práctica: asignar reservas, respetar preferencias de orientación, reducir el desperdicio de plazas y minimizar el número de habitaciones abiertas.

2.4 Operadores

El dominio del hotel se basa en dos operadores principales, presentes desde el nivel básico, cuyos efectos se van enriqueciendo con información numérica en las extensiones métricas.

asignar-reserva
Esta acción asigna una reserva a una habitación concreta.

- **Precondiciones**
    - Estas condiciones garantizan que no se sobrecargan habitaciones y que no hay solapamientos de reservas en los mismos días.
      La reserva no debe estar ya asignada: not (asignada ?r).
    - La habitación debe tener capacidad suficiente: <= (num-personas ?r) (capacidad ?h).
    - Para todos los días de la reserva, la habitación no puede estar ocupada:
      forall (?d - dia) (not (and (dia-de-reserva ?r ?d) (ocupada ?h ?d))).
- **Efectos lógicos**
    - Marca la reserva como asignada: (asignada ?r).
    - Marca la habitación como ocupada en todos los días de la reserva:
      forall (?d - dia) (when (dia-de-reserva ?r ?d) (ocupada ?h ?d)).
      Con esto se actualiza de forma explícita el calendario de ocupación del hotel.
- **Efectos numéricos y condicionales (extensiones métricas)**

  A partir de las extensiones se añaden efectos sobre las funciones de coste:
    - **Ext2 – orientación**: si la orientación de la habitación no coincide con la preferencia de la reserva, se incrementa coste-orientacion en 1 mediante un when que comprueba que no existe ninguna dirección compartida entre preferencia y orientacion.
    - **Ext3 – desperdicio**: se incrementa coste-desperdicio en la cantidad (- (capacidad ?h) (num-personas ?r)), que representa las plazas desaprovechadas en esa asignación.
    - **Ext4 – habitaciones usadas**: si la habitación todavía no se había usado (not (habitacion-usada ?h)), se la marca como usada y se incrementa coste-habitaciones en 1, evitando contar varias veces la misma habitación.

Este operador es el responsable de materializar en el estado tanto la decisión de asignación como el impacto de esa decisión en los distintos criterios de calidad del plan.

descartar-reserva
Esta acción permite decidir que una reserva no se asignará a ninguna habitación.

- **Precondiciones**
    - La reserva no debe estar ya asignada ni descartada:
      not (asignada ?r) y not (descartada ?r).
- **Efectos**
    - Marca la reserva como descartada: (descartada ?r).
    - Incrementa coste-descartar en una cantidad fija (en el modelo, 10 unidades).

Gracias a este operador, el planificador puede optar por no asignar ciertas reservas cuando el coste de aceptarlas (en términos de orientación, desperdicio o habitaciones adicionales) sería mayor que la penalización por descartarlas, combinando así correctamente los criterios de las distintas extensiones según el enunciado de la práctica. Es decir, sólo se descartará una habitación si realmente se consigue un mayor beneficio asignando otra o otras reservas en su período de reserva.

3. Juego de Prueba

3.1 Nivel Básico:

Para este nivel se usan las variables y predicados ya comentados anteriormente, pero para entender la posible salida se repasará la única acción que tiene:

```
(:action asignar-reserva
   :parameters (?r - reserva ?h - habitacion)
   :precondition (and
      (not (asignada ?r))
      (<= (num-personas ?r) (capacidad ?h))
      ;; comprueba que ningún día esté ocupado
      (forall (?d - dia)
         (not (and (dia-de-reserva ?r ?d)
                   (ocupada ?h ?d))))
   )
   :effect (and
      (asignada ?r)
      ;; marca los días como ocupados
      (forall (?d - dia)
         (when (dia-de-reserva ?r ?d)
               (ocupada ?h ?d)))
   )
)
```

Lo que se hace simplemente es, teniendo en cuenta que la reserva aún no ha sido asignada y respeta el criterio de la capacidad, se itera por los objetos *dia*, y por cada uno de ellos mira si es uno de los que quiere la reserva y ese habitación no esta marcada como ocupada en ese día. A continuación, la marca como asignada y que esos días la habitación está ocupada.

Completado de Nivel:

```
(:goal
  (and
    (asignada r1)
    (asignada r2)
    (asignada r3)
  )
)
```

Con esto ya sabemos si se han asignado o no todas las reservas, tal y como nos pide el enunciado se deben asignar todas o ningunas y este *goal* tiene la función de asegurar eso.

a) Asignación correcta de Reservas a Habitaciones

Habitaciones	Reservas	Días de Reserva
H1, capacidad = 2 H2, capacidad = 4	R1, personas = 2 R2, personas = 1 R3, personas = 3	R1: 1-3 R2: 1-3 R3: 7-8

Resultado:

step 0: ASIGNAR-RESERVA R3 H2
1: ASIGNAR-RESERVA R1 H2
2: ASIGNAR-RESERVA R2 H1

Según la planificación, vemos que tanto R1 como R2 están en habitaciones diferentes(respetando el solapamiento) y R3 también se asigna(de forma indiferente a cualquiera de las dos) lo que es correcto.

b) Asignación parcial de Reservas a Habitaciones

Lo que se quería comprobar era que, tal y como establece el enunciado básico, si no se pueden asignar todas las reservas , no se asigna ninguna. Con la siguiente entrada:

Habitaciones	Reservas	Días de Reserva
H1, capacidad = 2 H2, capacidad = 4	R1, personas = 2 R2, personas = 1 R3, personas = 3	R1: 1-3 R2: 1-3 R3: 1-3

Resultado:
advancing to distance: 3
2

best first search space empty! problem proven unsolvable.

Evidentemente, al haber 3 reservas sobre dos habitaciones en el mismo período, una no se asignaría entonces, no existe ninguna asignación válida para todas las reservas.

Adicionalmente, también se ha probado el juego de pruebas a), pero cambiado el número de personas de R3 a 5, y obtenemos lo siguiente:
ff: goal can be simplified to FALSE. No plan will solve it → Tampoco existe solución válida.

c) Juego de Pruebas Generado

Reserva	Personas	Días
R1	2	7-9
R2	1	21
R3	3	25-30
R4	1	28-30
R5	2	15-16

Resultado:
ff: found legal plan as follows

step 0: ASIGNAR-RESERVA R3 H3
1: ASIGNAR-RESERVA R2 H3
2: ASIGNAR-RESERVA R5 H3
3: ASIGNAR-RESERVA R1 H3
4: ASIGNAR-RESERVA R4 H2

Lo primero que se puede observar del plan es que nunca se usa la Habitación 1, cosa que puede extrañar pero en realidad no existe ninguna necesidad de usarla, ya que además de tener capacidad de dos personas(sólo R2 y R4 pueden acceder), sólo existe un solapamiento pero afecta a 2 habitaciones, entonces es indiferente cuál de ellas usar.

3.2 Extensión 1

En este caso es necesario hacer varios cambios, primero de todo añadir una métrica:

(:metric minimize (coste))

Inicializando en el :init con coste 0, y será a minimizar, ya que el problema se planteará que al descartar una reserva es cuando realmente implica un coste(queremos el máximo número de reservas posibles para el hotel).

El encargado de incrementar esta variable de coste será una nueva acción:

```
(:action descartar-reserva
   :parameters (?r - reserva)
   :precondition (and
      (not (asignada ?r))
      (not (descartada ?r))
   )
   :effect (and
      (descartada ?r)
      (increase (coste) 1)
   )
)
```

Fig A

Dado que el flujo de plan de ejecución encuentra una secuencia de pasos válidos que minimizan el coste, esto es aceptable ya que no se usará si antes lo puede asignar.

Completado de Nivel:

Al introducir métricas de optimización, el criterio de completado del problema cambia. Ya no deben de estar asignadas todas las reservas, sino que todas ellas hayan sido tratadas. Esto se hace mediante un objetivo que requiere que cada reserva esté marcada como asignada o descartada(acción de la Fig A). De este modo, el *goal* indica únicamente cuándo el plan está completo, mientras que la métrica se encarga de evaluar la calidad del plan, penalizando descartes, desperdicio u otros criterios definidos.

```
(:goal
  (and
    (forall (?r - reserva)
      (or (asignada ?r) (descartada ?r)))
  )
)
```

Este completado de nivel se usará para todas las extensiones posteriores, así que no se volverá a explicar.

a) Descarte de reservas

Para la primera prueba el objeto es ver si realmente se asignan el máximo número de reservas posibles y se prefiere descartar alguna que no convendría para el plan.

Reserva	Personas	Días
R1	1	1-7
R2	2	1-7
R3	1	1-4
R4	2	5-8

step 0: ASIGNAR-RESERVA R1 H2
1: DESCARTAR-RESERVA R2
2: ASIGNAR-RESERVA R4 H1
3: ASIGNAR-RESERVA R3 H1
4: REACH-GOAL

El plan encontrado prefiere descartar R2 para poder asignar R3 y R4 en el mismo espacio de tiempo, antes que tener simplemente dos reservas, cosa que es lo que se quería. Adicionalmente, se puede observar respecto al nivel básico que se quedan reservas sin asignar.

b) Juego de Pruebas Generado

A partir de esta extensión se amplía ligeramente el número de reservas y habitaciones. La idea es, una vez ya comprobado su correcto funcionamiento, ver cómo se comporta en un problema de mayor magnitud.

Habitación	Capacidad
h1	2
h2	4
h3	3

Reserva	Personas	Días ocupados
r1	2	17–21
r2	3	28
r3	1	13–14
r4	2	1–2
r5	1	19
r6	1	6–12
r7	3	23–25
r8	3	21–24
r9	3	13–19
r10	3	22–24

Resultado:

step 0: ASIGNAR-RESERVA R1 H3
1: DESCARTAR-RESERVA R8
2: ASIGNAR-RESERVA R10 H2
3: ASIGNAR-RESERVA R9 H2
4: ASIGNAR-RESERVA R7 H3
5: ASIGNAR-RESERVA R6 H1
6: ASIGNAR-RESERVA R5 H1
7: ASIGNAR-RESERVA R4 H1
8: ASIGNAR-RESERVA R3 H1
9: ASIGNAR-RESERVA R2 H2
10: REACH-GOAL

Como vemos en esta nueva extensión el plan dice que se ha descartado la reserva R8, el plan sólo muestra la secuencia válida con el coste mínimo en este caso se ha descartado esta ya que en las fechas 21-24 hay un total de 4 intersecciones y sólo 3 habitaciones, y R8 es la que forma mayor parte del rango de solapamiento, cosa que justificaría su descarte.

3.3 Extensión 2

En la extensión 2 se introducen preferencias de orientación. Cada habitación tiene una orientación fija y cada reserva una orientación preferida. Si una reserva se asigna a una habitación con orientación distinta a su preferencia, se incrementa el coste de orientación.

El problema optimiza conjuntamente el coste por descartar reservas y el coste por no respetar preferencias.

```
(:objects
   ;;orientacion
   n s e o - direccion
 )
```

```
(:init
;; orientación de las habitaciones
   (orientacion h1 n)
   (orientacion h2 s)

   ;; preferencias de las reservas
   (preferencia r1 n)
   (preferencia r2 e)
   (preferencia r3 o)
   (preferencia r4 s)
)
```

Para probar su correcto funcionamiento mediante se elaboraba, se han hecho 3 pruebas iniciales. Antes recordar la *métrica* usada:

(:metric minimize (+ (coste-descartar) (coste-orientacion)))

Dónde cada vez que se descarta se suma +10 al coste-descartar y si la orientación no es la adecuada +1 a coste-orientación. Sigue siendo a minmizar ya que sumaremos +1 si la orientación no es la correspondiente.

```
(:action asignar-reserva
    …
    :effect (and
        …
        ;; penalización si orientación no coincide
        (when (not (exists (?dir - direccion)
              (and (preferencia ?r ?dir) (orientacion ?h ?dir))))
          (increase (coste-orientacion) 1))
    )
 )
```

Esto se resuelve sin la igualdad, sinón con inferencia de la variable, se mira que si no existe alguna dirección *dir* que no sea la misma tanto para preferencia de la reserva como para orientación de la habitación, el coste se incrementa.

Completado de Nivel: El mismo que en la extensión 1.

a) Comprobar sólo Orientación Favorable frente otras reservas

La idea es ver simplemente si en un caso de solapamiento si se escogen las reservas que tengan la misma preferencia de orientación que la propia habitación.

Orientación de las habitaciones (orientacion h1 n) (orientacion h2 s)	Preferencias de las reservas (preferencia r1 n) (preferencia r2 e) (preferencia r3 o) (preferencia r4 s)

Y todas las reservas r1,r2,r3,r4 van del día 1 al 3. Entonces obtenemos:

step 0: ASIGNAR-RESERVA R4 H2
1: DESCARTAR-RESERVA R3
2: DESCARTAR-RESERVA R2
3: ASIGNAR-RESERVA R1 H1
4: REACH-GOAL

Dónde podemos ver que al sólo poder asignar dos habitaciones para ese período se han asignado las que tenían la orientación preferida.

b) Comprobar que es mejor asignar una reserva a que no tenga la orientación preferida

El objeto de la prueba es ver si la métrica realmente prioriza los casos donde se asignan más reservas a sólo la orientación. Simplemente cambiando los días de reserva para que no hayan intersecciones(R3 y R4 quieren reservar del 4 al 7) y conservando las orientaciones y preferencias anteriores:

step 0: ASIGNAR-RESERVA R2 H2
1: ASIGNAR-RESERVA R3 H1
2: ASIGNAR-RESERVA R4 H2
3: ASIGNAR-RESERVA R1 H1
4: REACH-GOAL

Vemos claramente cómo asigna todas las reservas aunque no haya habitaciones orientadas a Este ni Oeste.

c) Juego de Pruebas Generado

Al haber ya hecho dos juegos de prueba manualmente para probar el correcto funcionamiento aquí el objeto es intentar ver si realmente se escoge la mejor solución y si son coherentes los posibles descartes.

Entrada:

Habitación	Capacidad	Orientación
h1	2	o
h2	4	n
h3	3	s

Reserva	Personas	Días ocupados	Preferencia
r1	2	8–10	n
r2	1	4–8	n
r3	1	6–12	e
r4	3	21–24	o
r5	3	22–24	e
r6	1	11–17	e
r7	3	6	n
r8	3	4–6	e
r9	1	21–27	e
r10	1	12–15	n

Resultado:

step 0: ASIGNAR-RESERVA R3 H3
1: ASIGNAR-RESERVA R9 H1
2: ASIGNAR-RESERVA R6 H1
3: ASIGNAR-RESERVA R5 H3
4: ASIGNAR-RESERVA R4 H2
5: ASIGNAR-RESERVA R2 H1
6: ASIGNAR-RESERVA R1 H2
7: DESCARTAR-RESERVA R8
8: ASIGNAR-RESERVA R7 H2
9: ASIGNAR-RESERVA R10 H2
10: REACH-GOAL

El plan obtenido es coherente con la extensión 2. El planificador prioriza asignar el máximo número de reservas, permitiendo violaciones de orientación cuando es necesario, y descarta únicamente aquellas reservas cuya asignación provocaría un coste global mayor. De hecho hay 3 casos en los que coincide cosa que tiene sentido desde el punto de vista probabilístico ya que hay un 25% de tener una orientación concreta y sólo hay 3 habitaciones con orientación.

3.4 Extensión 3

Para la extensión 3 se añade un nuevo fluente numérico coste-desperdicio y se modifica la acción asignar-reserva para que incremente este valor con las plazas que sobran en cada asignación:

```
(:functions
  ...
  (coste-desperdicio)
)

(:action asignar-reserva
  :parameters (?r - reserva ?h - habitacion)
  :precondition (and
    (not (asignada ?r))
    (<= (num-personas ?r) (capacidad ?h))
    (forall (?d - dia)
      (not (and (dia-de-reserva ?r ?d)
                (ocupada ?h ?d))))
  )
  :effect (and
    (asignada ?r)
    (forall (?d - dia)
      (when (dia-de-reserva ?r ?d)
            (ocupada ?h ?d)))
    (when (not (exists (?dir - direccion)
                (and (preferencia ?r ?dir)
                     (orientacion ?h ?dir))))
      (increase (coste-orientacion) 1))
    ;; Extensión 3: desperdicio de plazas
    (increase (coste-desperdicio)
      (- (capacidad ?h) (num-personas ?r)))
  )
)
```

La métrica pasa a ser:

```
(:metric minimize
  (+ (coste-descartar)
     (coste-orientacion)
     (coste-desperdicio)))
```

De esta forma, además de evitar descartar reservas y penalizar orientaciones incorrectas, el planificador intenta asignar cada reserva a la habitación cuya capacidad esté más ajustada al número de personas de la reserva, minimizando las plazas sobrantes.

Completado de Nivel: El mismo que en la extensión 1.

Juego de prueba para la extensión 3

a) Manual

Para comprobar el correcto funcionamiento de la extensión 3 se ha definido un problema muy sencillo en el que una única reserva puede ser asignada a dos habitaciones distintas:

Habitaciones:
- h2 con capacidad 2.
- h4 con capacidad 4.

Reserva:
- rA con 2 personas, que ocupa los días d1 y d2.

Orientación:
- Ambas habitaciones (h2 y h4) tienen la misma orientación que la preferida por la reserva, de forma que coste-orientacion siempre vale 0 y no influye en la decisión.

Inicialmente:

```
(= (capacidad h2) 2)
(= (capacidad h4) 4)
(= (num-personas rA) 2)
(dia-de-reserva rA d1)
(dia-de-reserva rA d2)
(orientacion h2 n)
(orientacion h4 n)
(preferencia rA n)

(= (coste-descartar) 0)
(= (coste-orientacion) 0)
(= (coste-desperdicio) 0)
```

El objetivo es simplemente que la reserva esté asignada:
```
(:goal (asignada rA))
```

y la métrica:
```
(:metric minimize
  (+ (coste-descartar)
     (coste-orientacion)
     (coste-desperdicio)))
```

En este escenario, si rA se asigna a h2 el desperdicio es 2−2=0, mientras que si se asigna a h4 el desperdicio es 4−2=2. Al ejecutar Metric-FF se obtiene el siguiente plan:
```
step 0: ASIGNAR-RESERVA RA H2
```

Lo que muestra que el planificador elige siempre la habitación con menor desperdicio de plazas, tal como exige la extensión 3.

b) Juego de Pruebas Generado

Cómo en las otras extensiones, lo que se intenta ver si el plan final es coherente con un ejemplo más cercano a un caso real o de mayor tamaño.

Habitación	Capacidad	Orientación
h1	2	n
h2	4	o
h3	3	n

Reserva	Personas	Días ocupados	Preferencia
r1	2	5–10	n
r2	1	4–8	s
r3	3	25–26	s
r4	3	2–7	n
r5	1	8–12	o
r6	3	14–15	n
r7	1	1–7	o
r8	2	7–8	e
r9	2	12–16	s
r10	1	24–25	o

Resultado:

step 0: ASIGNAR-RESERVA R10 H1
1: ASIGNAR-RESERVA R7 H1
2: ASIGNAR-RESERVA R5 H1
3: ASIGNAR-RESERVA R2 H2
4: ASIGNAR-RESERVA R9 H2
5: ASIGNAR-RESERVA R3 H3
6: DESCARTAR-RESERVA R8
7: DESCARTAR-RESERVA R1
8: ASIGNAR-RESERVA R4 H3
9: ASIGNAR-RESERVA R6 H3

Al tener varios criterios se podrá comentar mejor la solución con una tabla:

Reserva	Personas	Habitación	Capacidad	Desperdicio
r10	1	h1	2	1
r7	1	h1	2	1
r5	1	h1	2	1
r2	1	h2	4	3
r9	2	h2	4	2
r3	3	h3	3	0
r4	3	h3	3	0
r6	3	h3	3	0

Lo primero que se ve es que tanto R8 cómo R1 se descartan. Teniendo en cuenta que el descarte es lo peor que hay es imprescindible saber porque para ver si es coherente.

R1: En su período la habitación que tiene menos desperdicio y además tiene la orientación que prefiere es h1, lo que la hace un candidato perdecto para la reserva, no obstante, si nos fijamos en la útlima tabla adjuntada, los períodos de reserva de R5 y R7 son contínuos, es decir, se pueden obtener más reservas en H1 si se permiten a esas dos entrar por encima

de H1. En cuanto a las otras habitaciones, R1 genera más desperdicio que las asignadas, entonces se descarta definitivamente.

R8: Coincide con r4 (h3), r7 (h1), r2 (h2). El caso de H1 ya se ha comentado, y las otras simplemente es por desperdicio y orientación.

En conclusión, el plan obtenido en la extensión 3 es coherente. El planificador prioriza asignar reservas grandes a habitaciones con capacidad ajustada para minimizar el coste de desperdicio, incluso aunque esto suponga incumplir preferencias de orientación. Las reservas descartadas son aquellas cuya asignación supondría un mayor coste total debido a conflictos temporales, desperdicio de capacidad y penalizaciones de orientación.

3.4 Extensión 4

Para la extensión 4 se añade un nuevo criterio de optimización: además de evitar descartar reservas, penalizar asignaciones con orientación incorrecta y minimizar el desperdicio de plazas, se quiere minimizar el número de habitaciones distintas que se usan durante el mes.

Cambios en el modelo

Se introduce un predicado para marcar si una habitación ya se ha usado alguna vez y un fluente numérico que cuenta cuántas habitaciones distintas se han abierto:

```
(:predicates
  ...
  (habitacion-usada ?h - habitacion)    ;; verdadero si la habitación ya se ha utilizado
)

(:functions
  ...
  (coste-habitaciones)                  ;; coste por abrir habitaciones nuevas
)
```

En la acción asignar-reserva se incrementa este coste únicamente la primera vez que se asigna algo a una habitación, usando una condición when sobre el predicado habitacion-usada:

```
(:action asignar-reserva
  :parameters (?r - reserva ?h - habitacion)
  :precondition (and
    (not (asignada ?r))
    (<= (num-personas ?r) (capacidad ?h))
    (forall (?d - dia)
      (not (and (dia-de-reserva ?r ?d)
                (ocupada ?h ?d))))
  )
  :effect (and
    (asignada ?r)
    (forall (?d - dia)
      (when (dia-de-reserva ?r ?d)
            (ocupada ?h ?d)))

    ;; Extensión 2: orientación incorrecta
    (when (not (exists (?dir - direccion)
                (and (preferencia ?r ?dir)
                     (orientacion ?h ?dir))))
      (increase (coste-orientacion) 1))

    ;; Extensión 3: desperdicio de plazas
    (increase (coste-desperdicio)
```

```
      (- (capacidad ?h) (num-personas ?r)))

    ;; Extensión 4: penalizar abrir una habitación nueva
    (when (not (habitacion-usada ?h))
      (and
        (habitacion-usada ?h)
        (increase (coste-habitaciones) 1)))
  )
)
```

La métrica global se actualiza para incluir este nuevo criterio, dando más peso a coste-habitaciones que al desperdicio, tal como indica el enunciado (“tiene prioridad no abrir habitaciones respecto a desperdiciar plazas”):

```
(:metric minimize

  (+ (coste-descartar)

     (+ (coste-orientacion)

        (+ (* 2 (coste-habitaciones))

           (coste-desperdicio))))

)
```

Completado de Nivel: El mismo que en la extensión 1.

Juego de pruebas para la extensión 4

a) Manual

Para comprobar el funcionamiento de la extensión 4 se ha diseñado un problema en el que varias reservas se pueden acomodar en una única habitación a lo largo del tiempo, o se pueden repartir entre varias habitaciones diferentes.

En este juego de prueba se definen:

Habitaciones:

- hBig con capacidad 4.
- hSmall1, hSmall2, hSmall3 con capacidad 2.

Reservas:

r1, r2, r3, todas de 2 personas, pero en intervalos de días que no se solapan entre ellas:

- r1: días 1–3.
- r2: días 4–6.
- r3: días 7–9.

Orientaciones:

Todas las habitaciones tienen la misma orientación n y todas las reservas prefieren n, de forma que coste-orientacion siempre es 0 y no influye en la decisión.

Fragmento relevante del estado inicial:

;; capacidades

(= (capacidad hBig) 4)

(= (capacidad hSmall1) 2)

(= (capacidad hSmall2) 2)

(= (capacidad hSmall3) 2)

;; reservas de 2 personas

(= (num-personas r1) 2)

(= (num-personas r2) 2)

(= (num-personas r3) 2)

;; r1: días 1-3

(dia-de-reserva r1 d1)

(dia-de-reserva r1 d2)

(dia-de-reserva r1 d3)

;; r2: días 4-6

(dia-de-reserva r2 d4)

(dia-de-reserva r2 d5)

(dia-de-reserva r2 d6)

;; r3: días 7-9

(dia-de-reserva r3 d7)

(dia-de-reserva r3 d8)

(dia-de-reserva r3 d9)

;; orientaciones compatibles

(orientacion hBig n)

(orientacion hSmall1 n)

(orientacion hSmall2 n)

(orientacion hSmall3 n)

(preferencia r1 n)

(preferencia r2 n)

(preferencia r3 n)

;; costes iniciales

(= (coste-descartar) 0)

(= (coste-orientacion) 0)

(= (coste-desperdicio) 0)

(= (coste-habitaciones) 0)

El objetivo exige que todas las reservas estén asignadas:

```
(:goal
  (and
    (asignada r1)
    (asignada r2)
    (asignada r3)
  )
)
```

y se utiliza la métrica anterior que penaliza abrir habitaciones nuevas y el desperdicio de plazas.

Al ejecutar Metric-FF sobre este problema, se obtiene un plan de la forma:

step 0: ASIGNAR-RESERVA R3 HSMALL1

step 1: ASIGNAR-RESERVA R2 HSMALL1

step 2: ASIGNAR-RESERVA R1 HSMALL1

En este plan todas las reservas se asignan a la misma habitación (hSmall1), por lo que coste-habitaciones vale 1. Asignar las reservas a varias habitaciones distintas (por ejemplo, hSmall1, hSmall2, hSmall3 o mezclando con hBig) daría el mismo número de reservas asignadas pero incrementaría coste-habitaciones, resultando un coste total mayor según la métrica.

Este comportamiento muestra que la extensión 4 está funcionando correctamente: el planificador busca soluciones que asignan las reservas utilizando el menor número posible de habitaciones distintas, priorizando no abrir habitaciones nuevas frente a reducir ligeramente el desperdicio de plazas.

b) Juego de Pruebas Generado

Para esta extensión se ha hecho una modificación en el número de habitaciones respecto al número de reservas ya que al tener 10 reservas y sólo 3 habitaciones realmente se usarían todas y no podremos ver muchas diferencias(de hecho para el caso generado de la extensión 3 obtenemos el mismo resultado) así que aumentando ya a 4 habitaciones se pueden apreciar los cambios.

Lo que se intentará ver es si realmente hay reincidencia en las habitaciones en vez de "abrir" nuevas innecesariamente.

Entrada:

Habitación	Capacidad	Orientación
h1	2	n
h2	4	o
h3	3	n
h4	4	o

Reserva	Personas	Días ocupados	Preferencia
r1	1	22	s
r2	1	18–19	s
r3	1	24	n
r4	1	8–12	o
r5	3	14–15	n
r6	1	1–7	o
r7	2	7–8	e
r8	2	12–16	s
r9	1	24–25	o
r10	3	22–26	n

Resultado:

step 0: ASIGNAR-RESERVA R9 H4
1: ASIGNAR-RESERVA R8 H4
2: ASIGNAR-RESERVA R7 H3
3: ASIGNAR-RESERVA R2 H1
4: ASIGNAR-RESERVA R1 H1
5: ASIGNAR-RESERVA R6 H1
6: ASIGNAR-RESERVA R4 H1
7: ASIGNAR-RESERVA R3 H1
8: ASIGNAR-RESERVA R10 H3
9: ASIGNAR-RESERVA R5 H3
10: REACH-GOAL

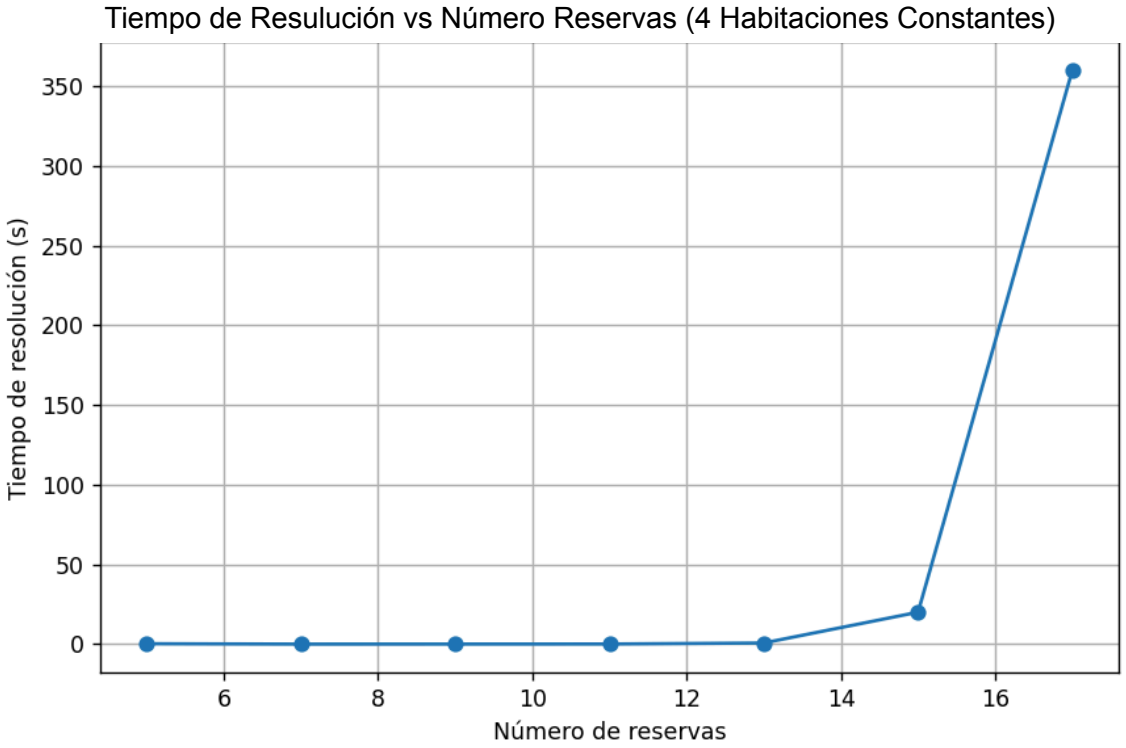
En este caso no se descarta ninguna habitación, que tiene sentido por lo que se ha comentado de la proporción de habitaciones respecto a reservas, pero se puede ver cómo claramente se han respetado los otros criterios. El primero que se puede observar es el añadido en la extensión 4, respecto a usar nuevas habitaciones o no, en este caso a pesar de tener 4 habitaciones la H2 nunca se usa porque no es necesario. En cuanto a los otros criterios se puede comentar que el de asignar el máximo número de reservas se sigue cumpliendo además de la orientación y el desperdicio(H3 alberga a todos con preferencia norte y con 3 personas en la reserva).

El plan tiene sentido con respecto a capacidades, preferencias y metric de la extensión 4. Algunas preferencias no coinciden exactamente con la orientación de la habitación o el desperdicio aceptado, pero eso es esperado. Las habitaciones se usan de manera eficiente y no hay descartes, lo que reduce el coste total.

4. Tiempos de ejecución y análisis del planificador

Habitaciones Constantes:

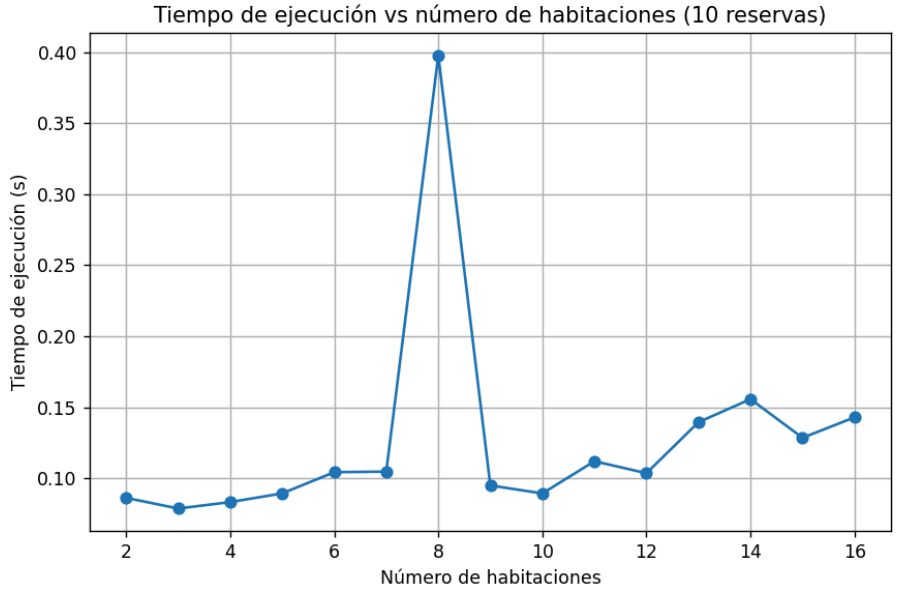
Para comparar los tiempos de ejecución se ha parametrizado el generador de juegos de prueba de la extensión 4, ya que incluye todas las otras, según el número de habitaciones y el número de reservas. De esta forma se ha podido comprobar cómo crece el tiempo de ejecución frente al tamaño del problema. El *script* se adjunta con la entrega. Aquí se detallan los resultados:

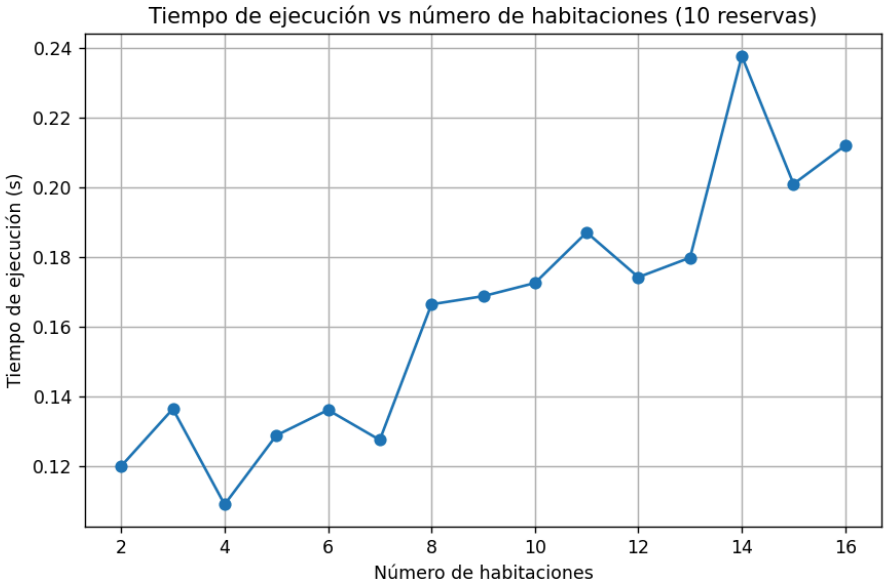


En este caso comprobamos sólo el número de reservas, se puede observar que el tiempo crece de forma no lineal aproximándose a un crecimiento exponencial en función de la entrada, cosa que tiene sentido teniendo en cuenta las acciones que disponemos.

Es decir, cada nueva reserva introduce nuevas combinaciones posibles de asignación a habitaciones. Teniendo en cuenta que el número de habitaciones se mantiene constante, el número de acciones posibles crece rápidamente con el número de reservas, ya que cada reserva interactúa potencialmente con todas las demás a través de las restricciones temporales.

Reservas Constantes:





El hecho de cambiar el número de habitaciones no tiene porque impactar tanto en el tiempo de ejecución ya que es más fácil tener una asignación válida para cualquiera de las secuencias de los planes hechos(menos conflictos de capacidad y/o solapamiento). No obstante, sí que cambia el número de acciones posibles ya que los criterios de capacidad y orientación se deben seguir comprobando.

En cuanto a los gráficos(la escala vertical es diferente) se puede observar que hasta 16 habitaciones el tiempo crece de forma más o menos lineal con algún outlier y variabilidad que dependen principalmente de la aleatoriedad de la seed escogida(constante para cada incremento de habitaciones). Como en el caso del 8 de la primera figura, dónde seguramente sea una combinación dónde haya muchos planes válidos pero sólo uno sea el mínimo.

5. Conclusión

En esta práctica se ha modelado un dominio de reservas de hotel en PDDL y se ha resuelto con Metric-FF, empezando por un nivel básico puramente lógico y añadiendo progresivamente funciones numéricas y métricas de calidad del plan. A lo largo de las cuatro extensiones se han incorporado decisiones de descarte, preferencias de orientación, minimización del desperdicio de plazas y penalización por abrir habitaciones nuevas, de forma que el planificador no solo encuentra planes válidos, sino que también optimiza distintos criterios en conflicto.

Los juegos de prueba diseñados para cada nivel muestran que el modelo se comporta según lo esperado: en el nivel básico solo acepta planes donde todas las reservas se asignan, mientras que en las extensiones métricas el sistema descarta reservas únicamente cuando ello reduce el coste total, elige orientaciones preferidas cuando es posible y prefiere habitaciones con capacidad ajustada y ya utilizadas antes de abrir nuevas. Los experimentos de rendimiento evidencian el crecimiento rápido del tiempo de búsqueda con el número de reservas y, en menor medida, con el número de habitaciones, ilustrando el aumento de complejidad combinatoria típico de los problemas de planificación a medida que crece el espacio de estados y de acciones.

En conjunto, la práctica permite comprobar en un contexto realista cómo un planificador numérico como Metric-FF utiliza la información lógica y los fluents de coste para guiar la búsqueda hacia soluciones eficientes, y cómo el diseño del dominio (predicados, funciones, operadores y métrica) influye directamente tanto en la calidad de los planes obtenidos como en el esfuerzo de cómputo necesario.