Pràctica de Planificació

Sistema d'Assignació de Reserves d'Hotel amb PDDL

Grau IA – Q1 Curs 2025-2026

Departament de Ciències de la Computació
Universitat Politècnica de Catalunya

Anel Ademovic
anel.suljic@estudiantat.upc.edu

Aleix Pitarch
aleix.pitarch@estudiantat.upc.edu

Joan Solina
joan.solina@estudiantat.upc.edu

7 de gener de 2026

Índex

1 Introducció **4**

2 Modelatge del Domini **5**

2.1 Tipus d'Objectes 5

2.2 Predicats 5

2.3 Funcions 6

3 Operadors **8**

3.1 Acció: assignar-reserva 8

3.2 Acció: descartar-reserva 9

4 Problema **11**

4.1 Estat Inicial 11

4.2 Estat Final (Goal) 11

4.3 Mètrica 12

5 Jocs de Prova **13**

5.1 Criteris de Disseny dels Jocs de Prova 13

5.2 Nivell Bàsic 13

5.2.1 Problema 1 13

5.2.2 Problema 2 14

5.3 Extensió 1 15

5.3.1 Problema 1 15

5.3.2 Problema 2 16

5.4 Extensió 2 17

5.4.1 Problema 1 17

5.4.2 Problema 2 18

5.5 Extensió 3 19

5.5.1 Problema 1 19

5.5.2 Problema 2 20

5.6 Extensió 4 21

5.6.1 Problema 1 21

5.6.2 Problema 2 22

6 Experimentació **24**

6.1 Experiment 1 24

6.1.1 Configuració 24

6.1.2 Resultats 24

6.2 Experiment 2 25

6.2.1 Configuració 25

6.2.2 Resultats 25

6.3 Comparació entre els experiments 1 i 2 26

6.4 Experiment 3 27

6.4.1 Configuració 27

6.4.2 Resultats 28

7 Conclusions **29**

1 Introducció

La planificació automàtica és una disciplina de la intel·ligència artificial que s'ocupa de trobar seqüències d'accions que permetin assolir objectius específics en entorns complexos. En aquest treball, apliquem tècniques de planificació automàtica a un problema real de gestió hotelera: l'assignació òptima de reserves d'allotjament a habitacions disponibles durant un període de 30 dies.

El problema d'assignació de reserves presenta diversos reptes computacionals interessants. Primer, existeixen restriccions dures que han de ser sempre satisfetes: cada habitació té una capacitat màxima de persones, i dues reserves no poden ocupar la mateixa habitació en dates que se solapin. Segon, més enllà d'aquestes restriccions bàsiques, hi ha múltiples criteris d'optimització que poden entrar en conflicte: maximitzar el nombre de reserves servides, respectar les preferències dels clients sobre orientació de les habitacions, minimitzar el desperdici d'espai assignant reserves a habitacions de mida adequada, i minimitzar els costos operatius utilitzant el menor nombre possible d'habitacions diferents.

Per abordar aquest problema, hem desenvolupat una sèrie de models en el llenguatge PDDL (Planning Domain Definition Language) que capturen els diferents aspectes del problema mitjançant predicats, funcions numèriques i operadors. Hem implementat cinc nivells de complexitat creixent:

- **Nivell Bàsic**: Assignació simple de reserves respectant només capacitat i no solapament temporal.
- **Extensió 1**: Introducció de la possibilitat de descartar reserves i optimització per maximitzar el nombre de reserves assignades.
- **Extensió 2**: Incorporació de preferències d'orientació de les habitacions, amb optimització multi-criteri que equilibra maximitzar assignacions i respectar preferències.
- **Extensió 3**: Minimització del desperdici de places lliures, incentivant assignacions ajustades a la capacitat necessària.
- **Extensió 4**: Optimització del nombre d'habitacions diferents utilitzades, amb l'objectiu de concentrar reserves i reduir costos operatius.

2 Modelatge del Domini

2.1 Tipus d'Objectes

El domini defineix tres tipus principals d'objectes que representen les entitats fonamentals del problema:

- **habitacio**: representa una de les habitacions de l'hotel disponibles per assignar reserves.
- **reserva**: representa una petició de reserva que cal processar i assignar.
- **dia**: representa un dels 30 dies del mes durant els quals es poden fer reserves.

2.2 Predicats

Predicats base:

- `(pendent ?r - reserva)`: indica que la reserva `?r` està pendent de processar. Serveix per controlar quines reserves encara no han estat tractades i per garantir que cada reserva es processi exactament una vegada.
- `(assignada ?r - reserva ?h - habitacio)`: indica que l'habitació `?h` ha estat assignada a la reserva `?r`. Aquest predicat es imprescindible per mantenir la traçabilitat de les assignacions i poder consultar posteriorment quina habitació s'ha assignat a cada reserva.
- `(ocupada ?h - habitacio ?d - dia)`: indica que l'habitació `?h` està ocupada durant el dia `?d`. Aquest predicat és crucial per evitar solapaments entre reserves, garantint que dues reserves no comparteixin la mateixa habitació en dates coincidents.
- `(dia-reserva ?r - reserva ?d - dia)`: indica que el dia `?d` forma part del període de la reserva `?r`. Aquest predicat és necessari per definir l'interval temporal de cada reserva i poder comprovar els solapaments amb altres reserves.

Predicats afegits a l'extensió 2 (orientacions):

A l'extensió 2, s'afegeixen dos predicats addicionals per gestionar les preferències d'orientació de les habitacions:

- `(te-orientacio ?h - habitacio ?o - orientacio)`: indica l'orientació cardinal (N/S/E/O) de l'habitació `?h`. Aquest predicat permet al planificador conèixer les característiques de cada habitació.

- `(prefereix-orientacio ?r - reserva ?o - orientacio)`: indica la preferència d'orientació de la reserva `?r`. Aquest predicat és necessari per poder avaluar si una assignació compleix amb les preferències dels clients.

Predicats afegits a l'extensió 4 (habitacions obertes):

A l'extensió 4, s'afegeix un predicat per controlar quines habitacions s'han utilitzat durant el mes:

- `(habitacio-oberta ?h - habitacio)`: indica que l'habitació `?h` ha estat utilitzada almenys una vegada durant el mes. Aquest predicat és fonamental per poder optimitzar el nombre d'habitacions diferents utilitzades, ja que permet detectar la primera vegada que s'assigna una habitació i incrementar el comptador corresponent.

2.3 Funcions

Les funcions defineixen atributs numèrics dels objectes i variables de control per a l'optimització.

Funcions bàsiques (presents en totes les extensions):

- `(capacitat ?h - habitacio)`: retorna el nombre màxim de persones que pot allotjar l'habitació `?h` (entre 1 i 4). Aquesta funció és imprescindible per verificar que una habitació té capacitat suficient per una reserva donada.
- `(num-persones ?r - reserva)`: retorna el nombre de persones de la reserva `?r` (entre 1 i 4). Aquesta funció és necessària per comprovar la compatibilitat entre reserves i habitacions.

Funcions d'optimització:

Aquestes funcions s'utilitzen com a comptadors per construir la funció objectiu a minimitzar en les diferents extensions:

- `(reserves-assignades)`: comptador utilitzat a partir de l'extensió 1 per penalitzar les reserves no assignades. S'incrementa en 1 cada vegada que es descarta una reserva, per tant un nom més encertat hagués estat **reserves-no-assignades**, però el vem determinar així al principi i ja no ho hem canviat a la resta d'extensions
- `(orientacions-incorrectes)`: utilitzada a l'extensió 2 per comptar el nombre d'assignacions que no compleixen amb la preferència d'orientació del client. S'incrementa en 1 cada vegada que s'assigna una reserva a una habitació amb orientació diferent a la preferida.

- `(places-lliures)`: utilitzada a les extensions 3 i 4 per comptabilitzar el desperdici de places. S'incrementa amb la diferència entre la capacitat de l'habitació i el nombre de persones de la reserva en cada assignació.
- `(habitacions-obertes)`: utilitzada a l'extensió 4 per comptar el nombre total d'habitacions diferents utilitzades durant el mes. S'incrementa en 1 la primera vegada que s'utilitza cada habitació.

La justificació de cadascuna d'aquestes funcions rau en la necessitat d'expressar els diferents criteris d'optimització requerits per cada extensió del problema.

3 Operadors

El domini defineix dues accions principals (`assignar-reserva` i `descartar-reserva`) que permeten al planificador construir solucions vàlides al problema d'assignació de reserves.

No es requereixen operadors addicionals perquè el problema no presenta accions més complexes: totes les decisions es redueixen a assignar o no assignar cada reserva, i el planificador pot explorar diferents combinacions d'aquestes decisions per trobar la solució òptima segons el criteri especificat.

3.1 Acció: assignar-reserva

Aquesta és l'acció principal del domini i està present en totes les extensions. Permet assignar una reserva pendent a una habitació que compleix els requisits necessaris.

Paràmetres:

- `?r - reserva`: la reserva a assignar.
- `?h - habitacio`: l'habitació on s'assignarà la reserva.

Precondicions:

1. `(pendent ?r)`: la reserva ha d'estar pendent de processar. Aquesta precondició és fonamental per evitar assignar la mateixa reserva múltiples vegades.
2. `(>= (capacitat ?h) (num-persones ?r))`: l'habitació ha de tenir capacitat suficient per allotjar totes les persones de la reserva. Aquesta comprovació és essencial per garantir que la solució és físicament viable.
3. `(forall (?d - dia) (imply (dia-reserva ?r ?d) (not (ocupada ?h ?d))))`: cap dels dies de la reserva pot coincidir amb una ocupació prèvia de l'habitació. Aquest quantificador universal és crucial per evitar solapaments temporals entre reserves a la mateixa habitació.

Aquestes precondicions són necessàries i suficients per garantir que qualsevol assignació realitzada compleix amb les restriccions dures del problema (capacitat i no solapament).

Efectes:

1. `(assignada ?r ?h)`: estableix la relació d'assignació entre la reserva i l'habitació. Aquest efecte és necessari per mantenir el registre de les assignacions.
2. `(not (pendent ?r))`: elimina la reserva de les pendents. Aquest efecte serveix per evitar reprocessar la mateixa reserva. i sobretot per crear l'objectiu (goal) del problema, que requereix que totes les reserves hagin estat processades almenys un cop.

3. `(forall (?d - dia) (when (dia-reserva ?r ?d) (ocupada ?h ?d)))`: marca tots els dies de la reserva com a ocupats a l'habitació assignada. Aquest efecte condicional és imprescindible per prevenir assignacions futures que causin solapaments.

Efectes específics per extensions:

A l'extensió 2, s'afegeix un efecte addicional per comptabilitzar les orientacions incorrectes:

- `(forall (?o - orientacio) (when (and (te-orientacio ?h ?o) (not (prefereix-orientacio ?r ?o))) (increase (orientacions-incorrectes) 1)))`: si l'orientació de l'habitació no coincideix amb la preferència de la reserva, s'incrementa el comptador d'orientacions incorrectes.

A l' extensió 3, s'afegeix l'efecte per comptabilitzar el desperdici de places:

- `(increase (places-lliures) (- (capacitat ?h) (num-persones ?r)))`: incrementa el comptador de places lliures amb la diferència entre la capacitat de l'habitació i el nombre de persones de la reserva.

A l'extensió 4, s'afegeix l'efecte per controlar l'obertura d'habitacions:

- `(when (not (habitacio-oberta ?h)) (and (habitacio-oberta ?h) (increase (habitacions-obertes) 1)))`: si és la primera vegada que s'utilitza l'habitació, es marca com a oberta i s'incrementa el comptador d'habitacions obertes.

3.2 Acció: descartar-reserva

Aquesta acció s'introdueix a l'extensió 1 i es manté a totes les extensions posteriors. Permet al planificador deixar reserves sense assignar quan no és possible trobar una assignació vàlida o quan assignar-les empitjoraria excessivament la funció objectiu. Al nivell bàsic, simplement no soluciona el problema i no dona cap assignació

Paràmetres:

- `?r - reserva`: la reserva a descartar.

Precondicions:

- `(pendent ?r)`: la reserva ha d'estar pendent. Aquesta és l'única precondició necessària, ja que qualsevol reserva pendent pot ser descartada.

Efectes:

1. `(not (pendent ?r))`: elimina la reserva de les pendents, permetent complir l'objectiu de l'extensió 1 que requereix que no hi hagi reserves pendents.

2. `(increase (reserves-assignades) 1)`: incrementa el comptador de reserves no assignades.

4 Problema

4.1 Estat Inicial

L'estat inicial defineix tota la informació coneguda al començament del problema. Bàsicament hem de declarar els objectes del problema (dies, habitacions i reserves) i incialitzar paràmetres

1. **Capacitat de les habitacions**: mitjançant la funció numèrica `(= (capacitat ?h) N)`, s'especifica el nombre màxim de persones que pot allotjar cada habitació.
2. **Característiques de les reserves**: per cada reserva es defineix:
   - `(= (num-persones ?r) N)`: nombre de persones de la reserva.
   - `(pendent ?r)`: predicat que indica que la reserva està pendent de processar.
   - `(dia-reserva ?r ?d)`: conjunt de predicats que estableixen quins dies ocupa la reserva.
3. **Inicialitzar a zero els contadors** necessàris per optimitzar la solució

Elements afegits per extensions:

A l'extensió 2, s'afegeix informació sobre orientacions, tant de les preferides per la reserva com la de les habitacions:

- `(te-orientacio ?h ?o)`: indica l'orientació de cada habitació.
- `(prefereix-orientacio ?r ?o)`: indica la preferència d'orientació de cada reserva.

4.2 Estat Final (Goal)

L'objectiu del problema és idèntic en totes les extensions i requereix que totes les reserves hagin estat processades:

- `(forall (?r - reserva) (not (pendent ?r)))`

Aquest objectiu obliga el planificador a prendre una decisió per cada reserva, ja sigui assignant-la a una habitació (acció `assignar-reserva`) o descartant-la (acció `descartar-reserva`, disponible a partir de l'extensió 1). Al nivell bàsic, sense l'acció de descartar, el planificador ha de trobar assignacions vàlides per a totes les reserves o el problema no tindrà solució.

4.3 Mètrica

La mètrica defineix el criteri d'optimització que guiarà el planificador cap a la millor solució possible. Cada extensió incorpora nous factors a considerar:

Nivell bàsic: No hi ha mètrica. El planificador només ha de trobar qualsevol pla vàlid que compleixi l'objectiu.

Extensió 1: S'introdueix la primera mètrica per minimitzar les reserves no assignades:

- `(:metric minimize (reserves-assignades))`

El valor del comptador s'incrementa cada reserva descartada, penalitzant les solucions que no assignen reserves.

Extensió 2: La mètrica combina dos objectius amb pesos diferents:

- `(:metric minimize (+ (orientacions-incorrectes) (* 100 (reserves-assignades))))`

El pes de 100 sobre les reserves no assignades garanteix que primer es prioritza assignar totes les reserves possibles, i només després s'optimitza la satisfacció de preferències d'orientació.

Extensió 3: S'afegeix l'objectiu de minimitzar el desperdici de places:

- `(:metric minimize (+ (places-lliures) (* 100 (reserves-assignades))))`

El comptador `places-lliures` s'incrementa amb la diferència entre la capacitat de l'-habitació i el nombre de persones de cada reserva assignada. El pes de 100 manté la prioritat d'assignar reserves sobre l'eficiència d'ocupació.

Extensió 4: La mètrica final incorpora tres criteris jeràrquics:

- `(:metric minimize (+ (places-lliures) (+ (* 100 (habitacions-obertes)) (* 1000 (reserves-assignades)))))`

Els pesos estableixen una jerarquia de decisió clara: es prioritza assignar reserves, minimitzant el nombre d'habitacions obertes, i finalment l'últim criteri és l'aprofitament de places i ajustar el nombre de persones a la reserva amb la capacitat de l'habitació. Els pesos fan que tot i que haguem d'obrir una habitació nova i penalitzi, sempre surti més rentable que rebutjar una assignació vàlida, o que igualment que tinguem habitacions ajustades per totes les reserves, no les obrim totes.

Aquesta estructura permet al planificador concentrar les reserves en el menor nombre d'habitacions possible sense sacrificar assignacions.

5 Jocs de Prova

En aquesta secció presentem els jocs de prova dissenyats per a cada nivell i extensió del problema d'assignació de reserves a habitacions d'hotel.

Tots els problemes presentats han estat generats utilitzant el generador automàtic desenvolupat en Python.

5.1 Criteris de Disseny dels Jocs de Prova

El disseny dels jocs de prova s'ha basat en diversos criteris per garantir que les proves són representatives i no trivials:

Satisfactibilitat garantida: Tots els problemes generats tenen solució vàlida, com demostren els resultats obtinguts amb el planificador FF. Això permet validar el comportament correcte del sistema sense ambigüitats.

Complexitat no trivial: Malgrat ser satisfactibles, els problemes requereixen exploració significativa de l'espai de cerca, com evidencien els temps de resolució i el nombre d'estats avaluats. Els escenaris plantegen solapaments temporals complexos i restriccions de capacitat que interactuen entre si.

Progressió de dificultat: Els problemes dins de cada extensió tenen nivells de complexitat diferents, permetent observar com el planificador gestiona des de casos moderats fins a escenaris computacionalment més exigents.

Cobertura funcional: Cada extensió té almenys dos problemes que proven aspectes diferents de la funcionalitat implementada, demostrant l'efecte dels pesos en les funcions objectiu i la robustesa del modelatge.

5.2 Nivell Bàsic

El nivell bàsic del problema requereix que el planificador assigni totes les reserves a habitacions, respectant les restriccions de capacitat i no solapament temporal, sense cap criteri d'optimització.

5.2.1 Problema 1

Aquest problema ha estat generat amb la comanda:

```
python problem_generator.py -e 5 -H 6 -r 8 -s 100
       -o nivell_basic_problema1.pddl
```

Característiques del problema:

- 6 habitacions amb capacitats variades
- 8 reserves amb diferents durades i nombre de persones

Objectiu del test:

Aquest problema té com a objectiu verificar que el planificador és capaç de trobar una assignació vàlida en un escenari amb una ràtio habitacions/reserves equilibrada ($6/8 = 0.75$). Volem comprovar que el sistema gestiona correctament els solapaments temporals i les restriccions de capacitat de manera simultània.

Per què és no trivial:

Amb 8 reserves i 6 habitacions, el planificador ha d'explorar l'espai de combinacions per trobar una assignació on cap habitació tingui solapaments. Les reserves tenen durades variades (generades aleatòriament entre 1 i 15 dies) i capacitats diferents (1-4 persones), creant restriccions que redueixen l'espai de solucions vàlides.

Resultats obtinguts:

El planificador FF ha trobat una solució vàlida assignant totes les 8 reserves:

- Temps de resolució: 0.15 segons
- Estats avaluats: 144
- Longitud del pla: 8 accions (una per reserva)
- Totes les reserves assignades correctament sense solapaments

La solució trobada demostra que el modelatge amb predicats `ocupada` i `dia-reserva` funciona correctament per evitar conflictes temporals. La baixa complexitat computacional (144 estats) indica que el problema és manejable però requereix cerca no trivial.

5.2.2 Problema 2

Aquest problema ha estat generat amb:

```
python problem_generator.py -e 5 -H 6 -r 7 -s 250
       -o nivell_basic_problema2.pddl
```

Característiques del problema:

- 6 habitacions amb capacitats variades

- 7 reserves amb intervals temporals diversos
- Llavor aleatòria: 250

Objectiu del test:

Aquest problema prova un escenari amb una ràtio encara més favorable ($6/7 = 0.86$), verificant que el sistema troba solucions eficients quan la pressió sobre els recursos és moderada. És útil per validar que el planificador no només funciona en casos límit, sinó també en situacions més còmodes.

Per què és no trivial:

Malgrat tenir més habitacions que reserves, els solapaments temporals aleatoris i les restriccions de capacitat segueixen creant un problema de cerca no trivial. No totes les combinacions d'assignacions són vàlides, i el planificador ha de trobar una que respecti totes les restriccions.

Resultats obtinguts:

El planificador ha resolt el problema de manera eficient:

- Temps de resolució: 0.00 segons (gairebé instantani)
- Estats avaluats: 73
- Longitud del pla: 7 accions
- Totes les reserves assignades sense conflictes

La baixa quantitat d'estats explorats (73) indica que, amb aquesta configuració particular, el planificador ha trobat ràpidament una solució vàlida. Això no treu complexitat al problema, sinó que demostra l'efectivitat de les heurístiques utilitzades pel planificador FF.

5.3 Extensió 1

A l'extensió 1 s'introdueix la possibilitat de descartar reserves i una mètrica d'optimització que penalitza cada reserva no assignada.

5.3.1 Problema 1

Aquest problema ha estat dissenyat manualment per crear un escenari específic amb conflictes temporals i de capacitat interessants:

Característiques del problema:

- 3 habitacions amb capacitats 1, 2 i 3 places
- 6 reserves amb solapaments temporals intensos
- Inclou una reserva de 4 persones (r4) que és incompatible per capacitat

Objectiu del test:

Aquest problema està dissenyat específicament per forçar el planificador a descartar reserves. Amb una reserva impossible d'assignar per capacitat i diversos solapaments temporals, el sistema ha de decidir quines reserves descartar per minimitzar el cost.

Per què és no trivial:

La combinació de restriccions temporals i de capacitat crea un problema on no totes les reserves poden ser servides. El planificador ha d'explorar l'espai de decisions per trobar quina combinació de reserves assignades i descartades minimitza el cost global.

Resultats obtinguts:

El planificador ha trobat una solució òptima:

- Temps de resolució: 0.00 segons
- Estats avaluats: 60
- Reserves assignades: 5 (r1, r2, r3, r5, r6)
- Reserves descartades: 1 (r4)

Com s'esperava, la reserva r4 (4 persones) ha estat descartada per incompatibilitat de capacitat. Les altres 5 reserves s'han pogut assignar respectant els solapaments temporals. Aquest resultat valida que el planificador minimitza correctament el nombre de descarts.

5.3.2 Problema 2

Aquest problema ha estat generat amb:

```
python problem_generator.py -e 1 -H 4 -r 8 -s 400
       -o ext1_problema2.pddl
```

Característiques del problema:

- 4 habitacions amb capacitats variades
- 8 reserves generades aleatòriament

Objectiu del test:

Amb una ràtio de 4 habitacions per 8 reserves (0.5), aquest problema prova un escenari amb més demanda que oferta. El planificador ha de decidir quines reserves descartar per maximitzar el nombre d'assignades, considerant tant solapaments temporals com restriccions de capacitat.

Per què és no trivial:

L'espai de cerca és significativament més gran que en el problema manual: amb 8 reserves i múltiples opcions de descart, el planificador ha d'explorar moltes combinacions possibles per trobar la que minimitza el cost. A més, la distribució aleatòria de durades i capacitats crea un problema genuïnament complex.

Resultats obtinguts:

El planificador ha trobat una solució després d'una cerca substancial:

- Temps de resolució: 37.03 segons
- Estats avaluats: 44,582
- Reserves assignades: 6 (r1, r2, r3, r5, r6, r7)
- Reserves descartades: 2 (r4, r8)

El temps de resolució significatiu (37 segons) i l'alt nombre d'estats explorats (44,582) demostren que aquest és un problema computacionalment exigent. El planificador ha hagut de descartar 2 reserves per trobar una assignació vàlida per a les 6 restants, validant que el cost de 5 unitats per descart motiva el sistema a maximitzar les assignacions.

5.4 Extensió 2

En l'extensió 2, a més de maximitzar reserves assignades, es té en compte l'orientació preferida per cada reserva

5.4.1 Problema 1

Aquest problema ha estat generat amb:

```
python problem_generator.py -e 2 -H 6 -r 10 -s 500
       -o ext2_problema1.pddl
```

Característiques del problema:

- 6 habitacions amb diferents orientacions i capacitats

- 10 reserves amb preferències d'orientació variades

Objectiu del test:

Aquest problema planteja situacions on el planificador ha d'equilibrar tres dimensions: capacitat, temps i orientació. L'objectiu és verificar que el sistema respecta la jerarquia de prioritats: primer no descartar, després minimitzar orientacions incorrectes.

Per què és no trivial:

La combinació de solapaments temporals, restriccions de capacitat i preferències d'orientació genera un espai de decisions ric on el planificador ha de fer trade-offs complexos. Cada decisió d'assignació afecta les opcions futures en múltiples dimensions.

Resultats obtinguts:

El planificador ha trobat una solució després d'una cerca exhaustiva:

- Temps de resolució: 1842.34 segons (aproximadament 31 minuts)
- Estats avaluats: 133,741
- Reserves assignades: 9
- Reserves descartades: 1 (r1)

Aquest problema és el més computacionalment exigent de tots els jocs de prova. El temps de resolució de més de 30 minuts i l'exploració de més de 133,000 estats demostren la complexitat de gestionar simultàniament tres criteris. El fet que només s'hagi descartat una reserva valida que el planificador prioritza correctament les assignacions.

5.4.2 Problema 2

Aquest problema ha estat generat amb:

```
python problem_generator.py -e 2 -H 7 -r 12 -s 600
       -o ext2_problema2.pddl
```

Característiques del problema:

- 7 habitacions amb diferents orientacions
- 12 reserves amb preferències variades

Objectiu del test:

Aquest és el problema més gran de l'extensió 2 en termes de nombre de reserves (12). L'objectiu és validar que el sistema escala correctament i pot gestionar problemes de mida substancial.

Per què és no trivial:

Amb 12 reserves i 7 habitacions, hi ha moltes possibles assignacions i l'espai de cerca és ampli. Les restriccions d'orientació afegeixen complexitat addicional, obligant el planificador a considerar múltiples dimensions simultàniament.

Resultats obtinguts:

Contràriament al problema anterior, aquest s'ha resolt molt més ràpidament:

- Temps de resolució: 0.31 segons
- Estats avaluats: 396
- Reserves assignades: 12 (totes)
- Reserves descartades: 0
- Longitud del pla: 12 accions

Sorprenentment, aquest problema més gran s'ha resolt molt més ràpidament que el Problema 1. Això demostra que la complexitat computacional no depèn només de la mida, sinó també de la distribució específica de solapaments i preferències generada per la llavor aleatòria. En aquest cas, les heurístiques del planificador han trobat ràpidament una solució que assigna totes les reserves.

5.5 Extensió 3

L'extensió 3 afegeix el criteri de minimitzar el nombre de places lliures (desperdiciades) quan s'assignen reserves a habitacions més grans del necessari.

5.5.1 Problema 1

Aquest problema ha estat generat amb:

```
python problem_generator.py -e 3 -H 6 -r 11 -s 700
       -o ext3_problema1.pddl
```

Característiques del problema:

- 6 habitacions amb capacitats variades (1-4 places)

- 11 reserves amb diferents nombre de persones

Objectiu del test:

Aquest problema prova situacions on el planificador ha de decidir entre matchings perfectes de capacitat que poden crear conflictes temporals posteriors, versus matchings subòptims que deixen més flexibilitat. L'objectiu és verificar que el sistema fa raonament global, no només local.

Per què és no trivial:

El planificador ha de considerar el trade-off entre minimitzar desperdici de places (pes 1) i evitar descarts de reserves (pes 100). Una decisió primerenca d'assignar una reserva a l'habitació de capacitat exacta pot bloquejar opcions millors posteriors, obligant el planificador a explorar l'espai de cerca estratègicament.

Resultats obtinguts:

El planificador ha trobat una solució eficient:

- Temps de resolució: 0.77 segons
- Estats avaluats: 4,063
- Reserves assignades: 11 (totes)
- Longitud del pla: 11 accions
- Cost final: depèn del desperdici total de places

El temps de resolució moderat i l'exploració de més de 4,000 estats indiquen que el planificador ha fet una cerca no trivial per trobar una bona solució. L'assignació de totes les 11 reserves demostra que el pes de 100 per descart és suficient per motivar el planificador a acceptar cert desperdici de places abans de descartar reserves.

5.5.2 Problema 2

Aquest problema ha estat generat amb:

```
python problem_generator.py -e 3 -H 8 -r 14 -s 800
       -o ext3_problema2.pddl
```

Característiques del problema:

- 8 habitacions amb tota la gamma de capacitats
- 14 reserves amb tots els tamanys possibles

Objectiu del test:

Aquest és el problema més gran de l'extensió 3, amb 14 reserves a assignar. L'objectiu és validar l'escalabilitat del sistema i observar com gestiona el matching òptim quan hi ha moltes opcions disponibles.

Per què és no trivial:

Amb 8 habitacions i 14 reserves, l'espai de possibles assignacions és molt ampli. El planificador ha d'explorar moltes combinacions per trobar una que maximitzi les reserves assignades i minimitzi el desperdici global, respectant les restriccions temporals i de capacitat.

Resultats obtinguts:

El planificador ha resolt el problema després d'una cerca substancial:

- Temps de resolució: 11.41 segons
- Estats avaluats: 20,810
- Reserves assignades: 14 (totes)
- Longitud del pla: 14 accions

El temps de resolució de més de 11 segons i l'exploració de més de 20,000 estats demostren la complexitat computacional d'aquest problema. Tot i així, el planificador ha aconseguit assignar totes les 14 reserves, validant la robustesa del modelatge en problemes de mida considerable.

5.6 Extensió 4

L'extensió 4 és la més complexa, combinant tres criteris d'optimització: minimitzar reserves descartades, minimitzar habitacions diferents utilitzades, i minimitzar desperdici de places.

5.6.1 Problema 1

Aquest problema ha estat generat amb:

```
python problem_generator.py -e 4 -H 6 -r 8 -s 900
       -o ext4_problema1.pddl
```

Característiques del problema:

- 6 habitacions disponibles

- 8 reserves amb intervals temporals diversos

Objectiu del test:

Aquest problema té com a objectiu verificar que el planificador minimitza el nombre d'habitacions diferents utilitzades, reutilitzant-les quan els períodes de les reserves no se solapen. Volem observar com el sistema equilibra els tres criteris simultàniament.

Per què és no trivial:

L'extensió 4 utilitza efectes condicionals per incrementar el comptador d'habitacions obertes només la primera vegada que s'utilitza cada habitació. Això fa que l'espai de cerca creixi significativament, ja que el planificador ha de raonejar sobre l'ordre de les assignacions i la reutilització estratègica d'habitacions.

Els tres criteris poden entrar en conflicte: reutilitzar una habitació estalvia cost d'obertura (100) però pot incrementar desperdici de places si el matching de capacitat no és òptim.

Resultats obtinguts:

El planificador ha trobat una solució després d'una cerca intensiva:

- Temps de resolució: 325.47 segons (aproximadament 5.4 minuts)
- Estats avaluats: 69,005
- Reserves assignades: 7
- Reserves descartades: 1 (r1)
- Longitud del pla: 8 accions (7 assignacions + 1 descart)

Aquest és un dels problemes computacionalment més exigents, amb un temps de resolució de més de 5 minuts. L'alt nombre d'estats explorats (69,005) reflecteix la complexitat introduïda pels efectes condicionals i els tres criteris d'optimització. El planificador ha hagut de descartar 1 reserva, indicant que no era possible assignar totes les 8 amb els recursos disponibles.

5.6.2 Problema 2

Aquest problema ha estat generat amb:

```
python problem_generator.py -e 4 -H 6 -r 7 -s 1100
       -o ext4_problema2.pddl
```

Característiques del problema:

- 6 habitacions disponibles
- 7 reserves (una menys que el Problema 1)

Objectiu del test:

Amb una reserva menys que el Problema 1, aquest escenari permet validar que el temps de resolució es redueix significativament quan la mida del problema disminueix lleugerament. També prova la reutilització d'habitacions en un context on hi ha menys pressió sobre els recursos.

Per què és no trivial:

Tot i tenir menys reserves, el problema segueix sent computacionalment complex degut als efectes condicionals i els tres criteris d'optimització. El planificador ha de trobar una configuració que minimitzi el nombre d'habitacions obertes mentre respecta les altres restriccions.

Resultats obtinguts:

El planificador ha resolt el problema molt més ràpidament que el Problema 1:

- Temps de resolució: 1.56 segons
- Estats avaluats: 7,094
- Reserves assignades: 5
- Reserves descartades: 2 (r1, r2)
- Longitud del pla: 7 accions

La diferència dramàtica en temps de resolució (1.56s vs 325s) demostra la sensibilitat de la complexitat computacional a la mida del problema i la distribució específica generada per la llavor aleatòria. Tot i que aquest problema ha requerit descartar 2 reserves (en lloc d'1), el planificador l'ha resolt molt més ràpidament.

Aquest resultat il·lustra que, en problemes d'optimització amb múltiples criteris, la distribució específica de solapaments i capacitats pot tenir un impacte més gran en la complexitat computacional que la mida nominal del problema.

6 Experimentació

Hem realitzat una anàlisi experimental per avaluar l'evolució del temps de planificació en funció de la mida del problema. L'objectiu és minimitzar els costos d'assignació mentre es gestionen les restriccions del problema. Hem dissenyat tres experiments que analitzen diferents dimensions del creixement del problema.

6.1 Experiment 1

Aquest experiment avalua l'impacte del **creixement del nombre d'habitacions** mantenint constant el nombre de reserves.

6.1.1 Configuració

- Nombre de mostres: 9
- Habitacions: de 4 a 12 (creixent)
- Reserves: 4 (constant)

6.1.2 Resultats

Problema	Nombre Estats	Nombre Fets	Nombre Habitacions	Nombre Reserves	Temps (s)
1	53	169	4	4	0.16
2	193	279	5	4	0.16
3	239	308	6	4	0.16
4	339	387	7	4	0.16
5	1795	418	8	4	0.46
6	731	421	9	4	0.31
7	1039	524	10	4	0.31
8	1439	549	11	4	0.46
9	1271	592	12	4	0.62

Taula 1: Resultats Experiment 1

Els resultats mostren un creixement moderat del temps de planificació. Es detecta un increment significatiu en el nombre d'estats explorats, passant de 53 a 1795 estats. Atribuïm el creixement i decreixement del temps d'execució a que els problemes són generats aleatòriament, i per casualitat un problema pot ser més difícil que un altre, per exemple si hi han reserves que ocupen molts dies i són fàcils de descartar

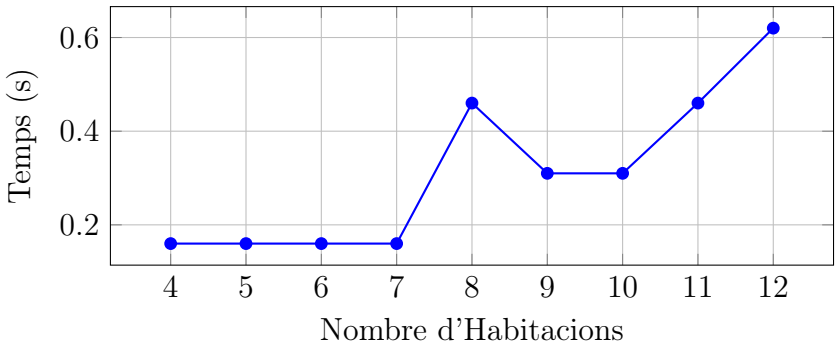

Figura 1: Evolució del temps de planificació amb creixement d'habitacions

6.2 Experiment 2

El segon experiment analitza l'efecte del **creixement del nombre de reserves** mantenint constant el nombre d'habitacions.

6.2.1 Configuració

- Nombre de mostres: 9
- Habitacions: 4 (constant)
- Reserves: de 4 a 12 (creixent)

6.2.2 Resultats

Problema	Nombre Estats	Nombre Fets	Nombre Habitacions	Nombre Reserves	Temps (s)
1	53	169	4	4	0.16
2	346	224	4	5	0.32
3	1692	233	4	6	0.46
4	1259	237	4	7	0.47
5	53313	277	4	8	790.95
6	251587	263	4	9	16313.28
7	55116	260	4	10	431.25
8	–	–	4	11	inf
9	245691	297	4	12	10268.28

Taula 2: Resultats Experiment 2

Els resultats evidencien un creixement exponencial del temps de planificació. Amb 4 reserves el temps és de 0.16s, però amb 9 reserves arriba a aproximadament 4.5 hores. El problema amb 11 reserves no va trobar solució, va petar la memòria abans d'acabar.

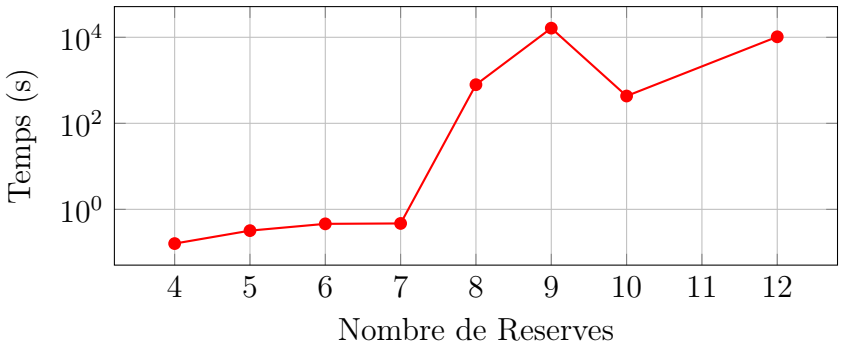

Figura 2: Evolució del temps de planificació amb creixement de reserves (escala logarítmica)

6.3 Comparació entre els experiments 1 i 2

Com és d'esperar, incrementar el nombre d'habitacions com en l'experiment 1, no és un problema per el planificador, el creixement és quasi constant, ara bé, incrementar el nombre de reserves, significa fer moltes mes comprovacions per el mateix nombre d'habitacions, moltes més combinacions possibles que tenen un impacte molt més significatiu en la complexitat computacional. A la figura 3 veiem la comparació entre temps d'execució.

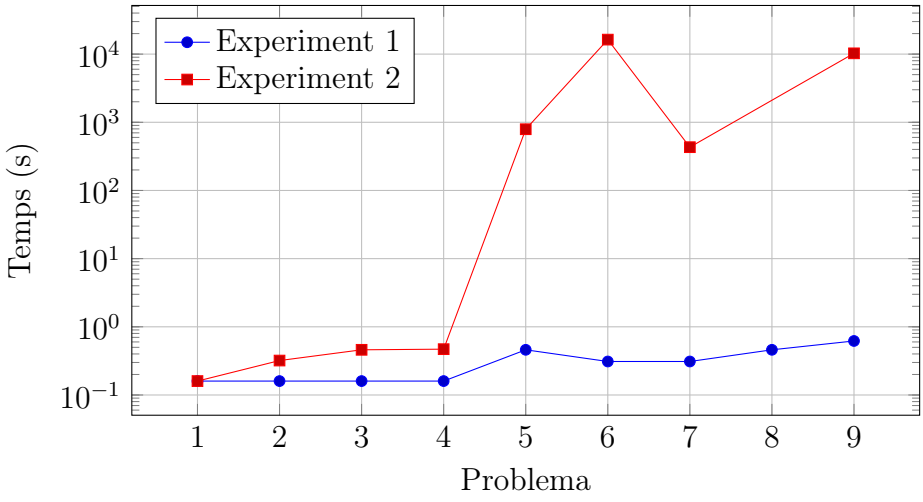

Figura 3: Comparació del temps d'execució entre els experiments 1 i 2 (escala logarítmica)

Pel que fa al nombre d'estats, veiem que el creixement és semblant, amb un creixement important al cinquè problema i després un creixement més moderat

I finalment, el nombre de fets també té un creixement semblant, tot i que la quantitat acaba sent molt menor al nombre d'estats o temps d'execució

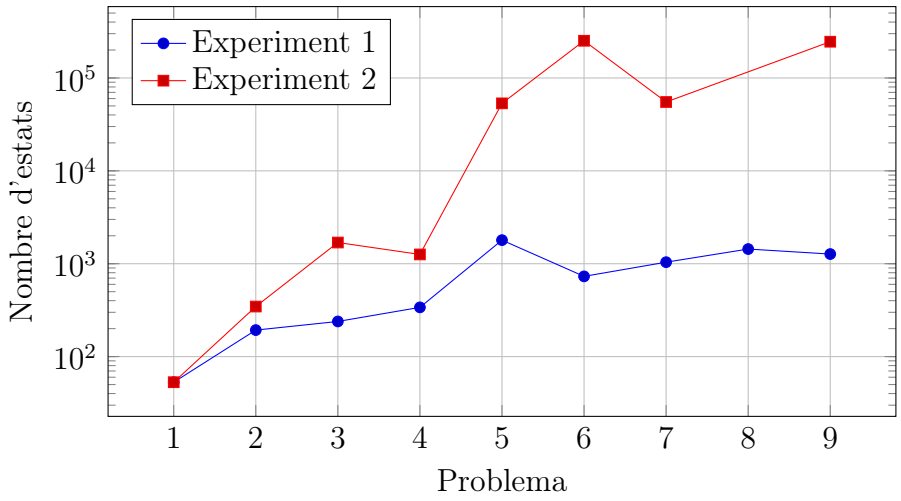

Figura 4: Comparació del nombre d'estats entre Experiment 1 i Experiment 2 (escala logarítmica)

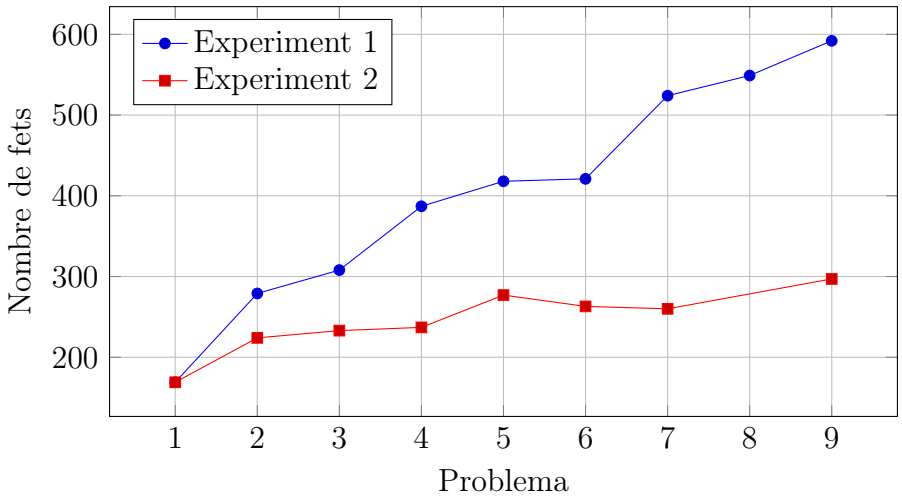

Figura 5: Comparació del nombre de fets entre Experiment 1 i Experiment 2

6.4 Experiment 3

L'últim experiment combina el **creixement simultani d'habitacions i reserves** per analitzar l'efecte conjunt.

6.4.1 Configuració

- Nombre de mostres: 5
- Habitacions: de 3 a 7 (creixent)
- Reserves: de 4 a 8 (creixent)

Problema	Nombre Estats	Nombre Fets	Nombre Habitacions	Nombre Reserves	Temps (s)
1	54	151	3	4	0.00
2	408	222	4	5	0.16
3	614	240	5	6	0.00
4	54880	386	6	7	670.31
5	375683	442	7	8	46820.47

Taula 3: Resultats Experiment 3

6.4.2 Resultats

Els resultats mostren un creixement exponencial molt pronunciat. El temps passa de valors pràcticament instantanis a aproximadament 13 hores. El nombre d'estats explorats creix moltíssim, arribant a 375683 estats en l'últim problema.

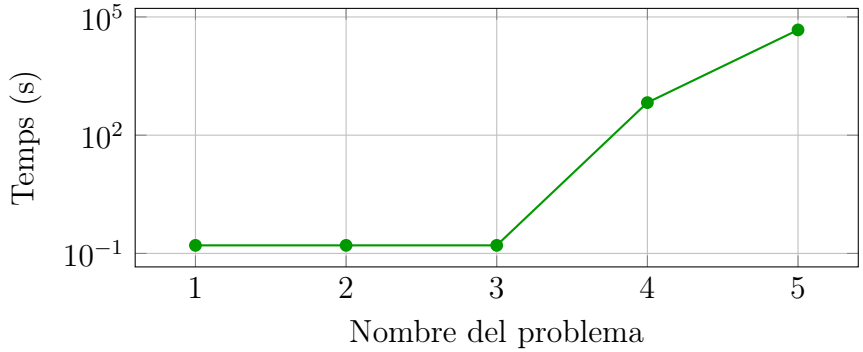

Figura 6: Evolució del temps amb creixement simultani (escala logarítmica)

7 Conclusions

Aquest treball ha demostrat la viabilitat d'aplicar tècniques de planificació automàtica al problema d'assignació de reserves d'hotel, obtenint solucions òptimes respecte a múltiples criteris simultàniament. A continuació presentem les conclusions principals del projecte, organitzades per àrees temàtiques.

Sobre el Modelatge, el modelatge en PDDL del problema d'assignació de reserves ha resultat ser expressiu i eficient.

Les funcions numèriques han demostrat ser l'eina adequada per implementar criteris d'optimització complexos. L'ús de comptadors que s'incrementen segons els efectes de les accions permet expressar funcions objectiu multi-criteri de manera natural. La jerarquització de criteris mitjançant pesos relatius (1000:100:1 a l'extensió 4) ha funcionat correctament, motivant el planificador a prendre decisions coherents amb les prioritats establertes.

Ha resultat molt útil modelar la pràctica iterativament, implementant primer el nivell bàsic i anar avançant per les extensions de mica en mica, provant només una cosa a la vegada.

L'evolució incremental del modelatge a través de les extensions ha permès validar cada afegit de manera independent. Aquesta metodologia facilita la detecció d'errors i permet comprendre l'impacte de cada component en la complexitat global del problema.

Sobre els Jocs de Prova, els jocs de prova han estat fonamentals per validar la correcció del modelatge implementat. Tots els problemes generats han estat resolts satisfactòriament pel planificador FF, demostrant que el sistema és capaç de gestionar escenaris diversos amb diferents nivells de complexitat.

La generació aleatòria de problemes mitjançant el generador en Python ha proporcionat diversitat en els casos de prova, simulant situacions realistes amb solapaments temporals i distribucions de capacitats variades. No obstant això, també ha revelat una sensibilitat significativa als paràmetres de generació: problemes amb mides similars poden presentar complexitats computacionals molt diferents segons la distribució específica de solapaments generada per la llavor aleatòria.

Els resultats dels jocs de prova validen que els pesos assignats a les funcions objectiu implementen correctament les jerarquies de prioritat especificades. Per exemple, el pes 100:1 entre descarts i orientacions incorrectes a l'extensió 2 motiva clarament el planificador a acceptar moltes orientacions subòptimes abans de considerar descartar una reserva.

Sobre l'Experimentació, l'anàlisi experimental ha revelat patrons clars sobre els factors que determinen la complexitat computacional del problema. El creixement del nom-

bre de reserves té un impacte molt més significatiu que el creixement del nombre d'habitacions: incrementar reserves causa creixement exponencial del temps de planificació, mentre que incrementar habitacions produeix un creixement més moderat.

Aquest comportament és coherent amb la naturalesa combinatòria del problema. Cada reserva addicional multiplica les opcions de decisió disponibles (assignar a quina habitació, o descartar), i afegeix noves restriccions temporals que interactuen amb les reserves existents. En canvi, habitacions addicionals simplement augmenten les opcions disponibles sense incrementar significativament l'espai de cerca.

El creixement simultani d'habitacions i reserves (Experiment 3) ha produït els temps de planificació més extrems, arribant a prop de 13 hores en el problema més gran. Això suggereix que per problemes de gran escala seria necessari explorar tècniques alternatives com descomposició temporal del problema, heurístiques específiques del domini, o hibridació amb algoritmes de satisfacció de restriccions.

La variabilitat observada entre problemes de mida similar (per exemple, a l'extensió 2 on el Problema 1 va trigar 31 minuts i el Problema 2 només 0.31 segons) indica que la distribució específica de solapaments té un impacte enorme en la dificultat real del problema. Això implica que, en un entorn productiu, seria valuós identificar característiques estructurals dels problemes que prediguin la seva complexitat abans d'intentar resoldre'ls.