0. Taula de continguts

1. Introducció

2. Objectius i metodologia

3. Disseny del domini i dels problemes

3.0 Extensió bàsica

L'extensió bàsica del domini `hotelbasic` implementa la funcionalitat fonamental d'assignació d'habitacions a reserves, gestionant les restriccions de compatibilitat i no-solapament temporal. Aquesta versió inicial serveix com a base per a les extensions posteriors, establint els conceptes clau i la lògica de recursos que es desenvoluparan més endavant.

3.0.1 Domini

El domini `hotelbasic` modela un problema d'assignació de recursos (scheduling) on un conjunt de peticions (`reserva`) han de ser assignades a recursos limitats (`habitacio`) durant uns intervals de temps específics (`dia`).

1. Tipus (`:types`) El domini defineix tres entitats bàsiques que estructuren el problema:
    - **`reserva`**: Representa una petició d'allotjament que ha de ser satisfeta.
    - **`habitacio`**: Representa el recurs físic (amb capacitat unitària) on s'allotgen les reserves. -**`dia`**: Representa la unitat de temps discreta.
2. Predicats (`:predicates`) Els predicats defineixen l'estat del món i les relacions entre els objectes:
    - **Predicats Estàtics (Dades d'entrada):**
        - `(dies-reserva ?r - reserva ?d - dia)`: Defineix l'interval temporal de cada reserva. Indica que la reserva `?r` requereix ocupació durant el dia `?d`.
        - `(compatible ?r - reserva ?h - habitacio)`: Restricció de domini que indica si l'habitació `?h` és vàlida per a la reserva `?r` (per exemple, per capacitat de persones).
    - **Predicats Dinàmics (Estat del sistema):**
        - `(assignada ?r - reserva)`: Indica que la reserva `?r` ja ha estat processada i té una habitació assignada.
        - `(ocupada ?h - habitacio ?d - dia ?r - reserva)`: Matriu d'ocupació que registra que l'habitació `?h` està bloquejada el dia `?d` per la reserva `?r`.
3. Accions (`:action`) L'única acció del sistema és **`assignar-habitacio`**, que formalitza la decisió d'ubicar una reserva.

- **Paràmetres:** Una reserva `?r` i una habitació `?h`.
- **Precondicions:** Per poder executar l'acció, s'han de complir tres condicions simultànies:
    1. `(not (assignada ?r))`: La reserva no ha d'estar ja assignada (evita duplicats).
    2. `(compatible ?r ?h)`: L'habitació ha de ser adequada per a la reserva.
    3. **Restricció de No-Solapament:**

        ```
        (not (exists (?d - dia ?r2 - reserva)
             (and (dies-reserva ?r ?d) (ocupada ?h ?d ?r2))))
        ```

        Això verifica que, per a tots els dies que demana la reserva `?r`, l'habitació `?h` no estigui ocupada per cap altra reserva `?r2`. És el nucli de la lògica de recursos.
- **Efectes:** Si s'executa, l'estat canvia:
    1. `(assignada ?r)`: La reserva es marca com a completada.

2. **Bloqueig de Recursos (Conditional Effect):**

```
(forall (?d - dia) (when (dies-reserva ?r ?d) (ocupada ?h ?d ?r)))
```

Per a cada dia `?d` que forma part de la reserva, es marca l'habitació `?h` com a ocupada.

3.0.2 Problemes

3.0.2.1 Problema 1: Poques habitacions, moltes reserves

En aquest experiment volem avaluar la capacitat del planificador per prioritzar i seleccionar el millor subconjunt de reserves quan els recursos són extremadament limitats. Per fer-ho, mantindrem fix el nombre d'habitacions (`n=2`) i incrementarem progressivament el nombre de reserves candidates (5, 10, 15, 20...). Això força el sistema a gestionar un escenari d'alta competència on la majoria de reserves s'han de descartar. S'espera observar:

1. **Comportament Intel·ligent:** El planificador haurà de triar les combinacions de reserves que maximitzin l'ocupació total (evitant forats temporals), en lloc d'agafar simplement les primeres de la llista.
2. **Escalabilitat:** S'espera un creixement no lineal (ràpid) del temps d'execució, ja que l'espai de cerca per trobar la combinació òptima creix combinatorialment a mesura que afegim més reserves solapades."

Per tant, plantegem el següent parell d'hipòtesis per a aquest experiment:

Respecte al comportament del planificador en aquest escenari d'alta competència:

- H_0: El planificador no és capaç de maximitzar l'ocupació total en situacions d'escassetat de recursos, seleccionant reserves de manera aleatòria o greedy.
- H_1: El planificador és capaç de maximitzar l'ocupació total, seleccionant reserves de manera intel·ligent per evitar forats temporals.

Hipòtesi sobre l'escalabilitat del planificador:

- H_0: El temps d'execució del planificador creix linealment amb el nombre de reserves, indicant una gestió eficient de l'espai de cerca.
- ParseError: KaTeX parse error: Double subscript at position 4: H_1_a: El temps d'execució del planificador creix de manera no lineal (ràpid) amb el nombre de reserves, indicant un augment combinatorial de l'espai de cerca.

Generem doncs diversos problemes amb 2 habitacions i un nombre creixent de reserves (1, 2, ... fins a 10) amb el nostre generador de problemes. Provem d'executar-los amb el planificador i mesurem el temps d'execució i les habitacions assignades amb èxit per a cada cas. Generem la quantitat de reserves de manera aleatòria completament, per la qual cosa els resultats poden variar lleugerament entre execucions. Executem cada problema diverses vegades i prenem la mitjana per obtenir resultats més fiables. Com que l'assignació és greedy, esperem que el nombre d'assignacions sigui proper al màxim possible (2 habitacions * nombre de reserves que caben sense solapament), però lògicament aquests casos seran difícils en termes generals exactament degut a l'atzar en la generació de reserves. Per tant esperem que el nombre d'assignacions sigui baix, i que hi hagi molts conflictes entre reserves, per tant que el planificador no convergeixi. Tot i així, el temps d'execució hauria de ser creixent, ja que el planificador haurà d'explorar moltes possibilitats per trobar la millor assignació possible, tot i que aquesta no sigui possible en aquest domini.

Pel que fa a la quantitat de reserves assignades, obtenim els següents resultats:

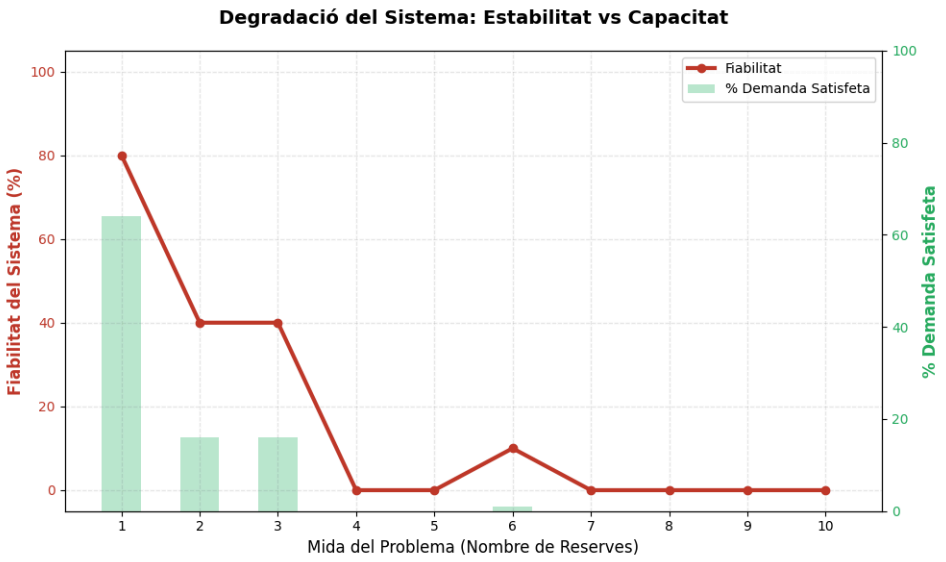

Figura 1: Reserves assignades en l'extensió bàsica

Pel que fa al temps d'execució, obtenim els següents resultats:

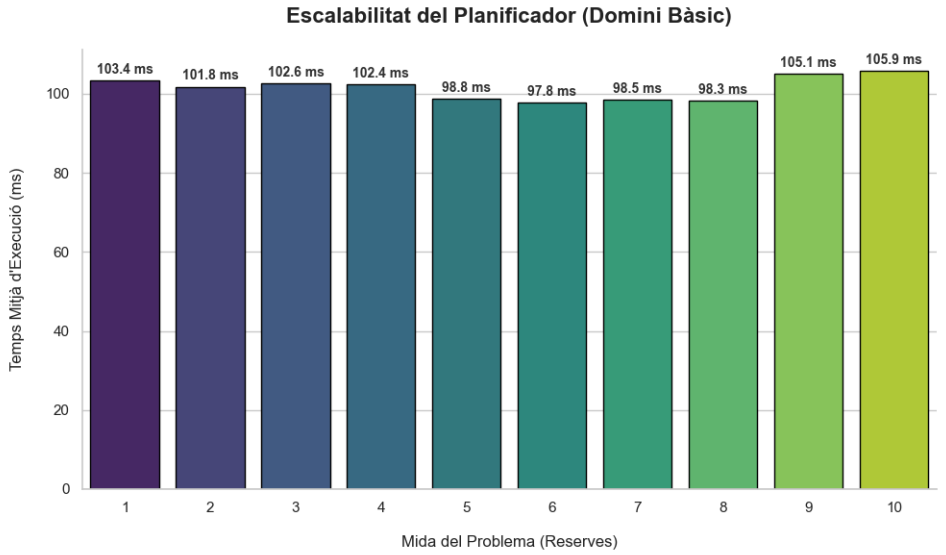

Figura 2: Temps d'execució en l'extensió bàsica

En primer lloc, l'anàlisi del temps de computació mostra un comportament aparentment estable. Tal com s'observa en el gràfic d'escalabilitat, el temps mitjà d'execució es manté constant al voltant dels 100 ms, independentment de la mida del problema (des d'una sola reserva fins a deu). Aquesta constància, lluny d'indicar una eficiència algorítmica en la resolució de problemes complexos, denota una fallada prematura. En situacions de saturació *on el nombre de reserves supera la capacitat disponible*, el planificador no inverteix temps a cercar solucions complexes perquè l'espai de cerca es tanca ràpidament. El sistema detecta la impossibilitat de satisfer totes les restriccions rígides del domini bàsic i conclou l'execució amb un veredicte d'insolubilitat (`unsolvable`) de manera gairebé immediata. Per tant, la latència baixa i constant no reflecteix escalabilitat, sinó la incapacitat del model per gestionar el conflicte.

Aquesta interpretació es confirma en analitzar la degradació del servei. El gràfic comparatiu entre estabilitat i capacitat evidencia un col·lapse abrupte del sistema. Amb una càrrega baixa (1 reserva), el sistema presenta una fiabilitat elevada (~80%) i satisfà una part significativa de la demanda. No obstant això, la fiabilitat cau dràsticament en augmentar la càrrega a 2 i 3 reserves, fins a arribar a un punt de ruptura a partir de les 4 reserves, on la taxa d'èxit es desploma al 0%.

Aquest comportament demostra que el domini bàsic manca de mecanismes de flexibilitat. En absència d'accions que permetin descartar reserves o relaxar restriccions, qualsevol instància on la demanda superi

l'oferta mínima d'habitacions esdevé irresoluble. En conseqüència, el planificador no degrada el seu rendiment progressivament (oferint solucions parcials), sinó que deixa de funcionar completament, confirmant la necessitat crítica d'incorporar extensions mètriques i accions de descart per dotar el sistema de robustesa en escenaris reals d'alta demanda.

Per tant, respecte a les nostres hipòtesis:

- En l'escenari de poques habitacions i moltes reserves, no rebutgem H_0, ja que el planificador no demostra un comportament intel·ligent en maximitzar l'ocupació total. De fet, no és capaç de trobar solucions en aquests casos, ja que el domini bàsic no permet descartar reserves ni relaxar restriccions.
- Pel que fa a l'escalabilitat, rebutgem ParseError: KaTeX parse error: Double subscript at position 4: H_1_a i rebutgem H_0, ja que el temps d'execució no creix linealment ni de manera no lineal amb el nombre de reserves. En canvi, es manté constant i baix degut a la incapacitat del planificador per trobar solucions en aquests escenaris d'alta competència.

3.0.2.2 Problema 2: Moltes habitacions, poques reserves

En aquest experiment volem avaluar la capacitat del planificador per gestionar eficientment els recursos quan hi ha una abundància d'habitacions disponibles en comparació amb el nombre de reserves. Per fer-ho, mantindrem fix el nombre de reserves (`m=2`) i incrementarem progressivament el nombre d'habitacions disponibles (5, 10, 15, 20...). Això crea un escenari on el planificador té moltes opcions per assignar les reserves, i s'espera que pugui trobar solucions òptimes ràpidament. S'espera observar:

1. **Comportament Eficient:** El planificador haurà de ser capaç d'assignar totes les reserves disponibles sense problemes, ja que hi ha suficients habitacions per satisfer la demanda.
2. **Escalabilitat:** S'espera que el temps d'execució creixi de manera lineal o sublineal, ja que l'espai de cerca per trobar assignacions òptimes és reduït en comparació amb l'escenari anterior.

Per tant, plantegem el següent parell d'hipòtesis per a aquest experiment:

Respecte al comportament del planificador en aquest escenari d'abundància de recursos:

- H_0: El planificador no és capaç d'assignar totes les reserves disponibles, deixant algunes sense assignar malgrat l'abundància d'habitacions.
- H_1: El planificador és capaç d'assignar totes les reserves disponibles, utilitzant eficientment les habitacions disponibles.

Hipòtesi sobre l'escalabilitat del planificador:

- H_0: El temps d'execució del planificador creix de manera no lineal amb el nombre d'habitacions, indicant una gestió ineficient de l'espai de cerca.
- H_1: El temps d'execució del planificador creix de manera lineal o sublineal amb el nombre d'habitacions, indicant una gestió eficient de l'espai de cerca.

Generarem problemes amb un nombre d'habitacions creixent (1, 10, 20, ..., fins a 100) i només 2 reserves. Executarem cada problema diverses vegades i prendrem la mitjana per obtenir resultats més fiables. Esperem que el nombre d'assignacions sigui sempre 2 (totes les reserves assignades) i que el temps d'execució sigui baix i creixi lentament a mesura que augmenta el nombre d'habitacions. Mantenim el nombre de dies a 10 per a tots els experiments.

Pel que fa a la quantitat de reserves assignades, obtenim els següents resultats:

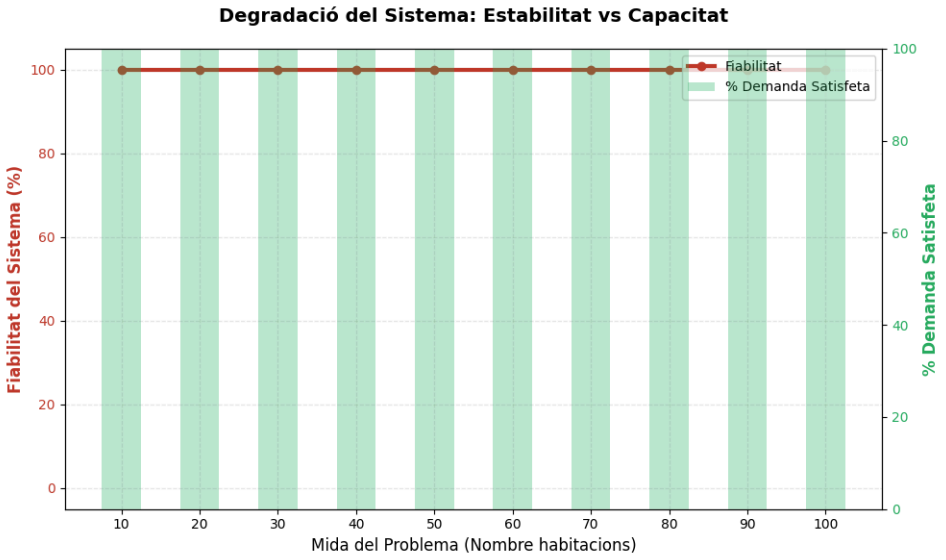

Figura 3: Reserves assignades en l'extensió bàsica (moltes habitacions)

És evident que totes les reserves es poden assignar amb èxit, ja que hi ha moltes habitacions disponibles, i aquest gràfic ho confirma clarament.

Pel que fa al temps d'execució, obtenim els següents resultats:

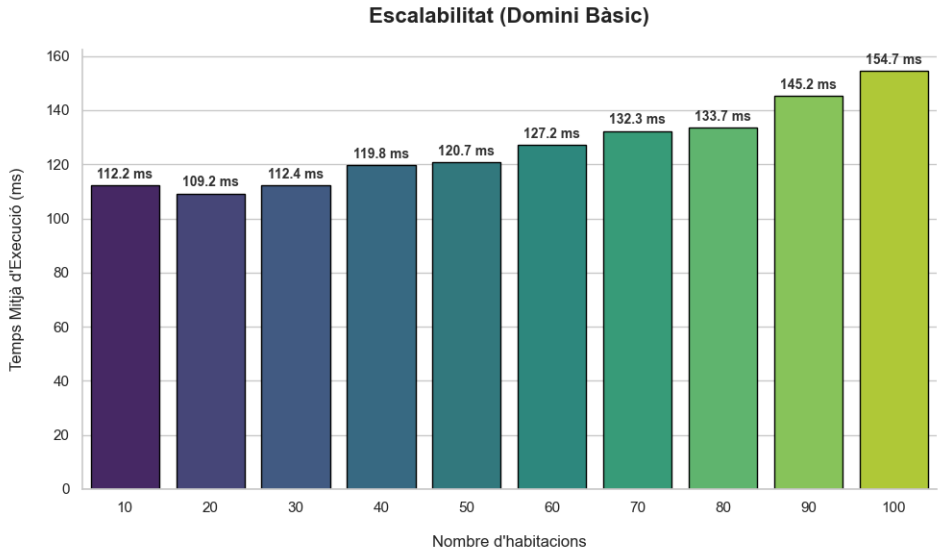

Figura 4: Temps d'execució en l'extensió bàsica (moltes habitacions)

Veiem un creixement lineal lleuger en el temps d'execució a mesura que augmenta el nombre d'habitacions, confirmant la nostra hipòtesi sobre l'eficiència del planificador en aquest escenari. Com que el temps d'execució és molt baix en general, això indica que el planificador gestiona molt bé l'abundància de recursos. Anem un pas més enllà i augmentem el nombre d'habitacions de 50 en 50 fins a 300 per veure si el comportament es manté. Obtenim els següents resultats:

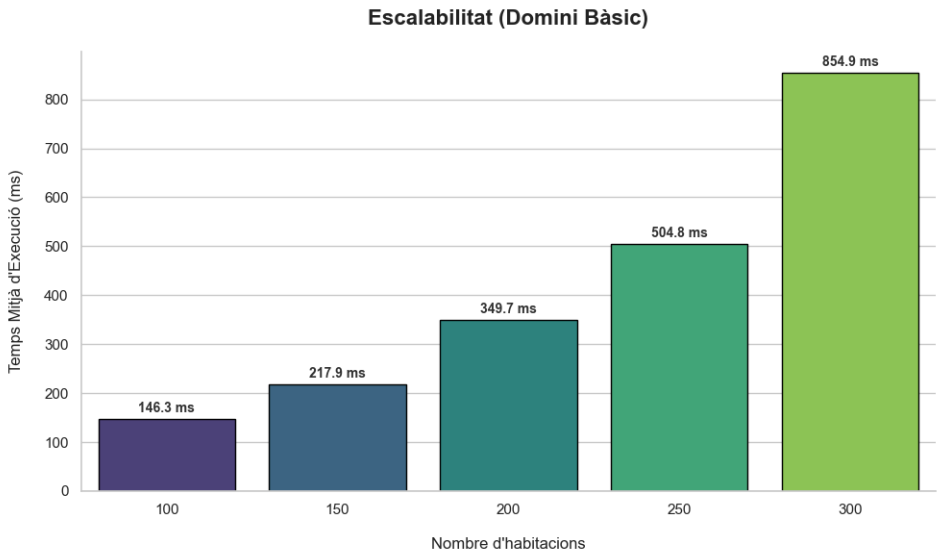

Figura 5: Temps d'execució en l'extensió bàsica (moltes habitacions fins a 300)

Confirmem que el temps d'execució segueix creixent de manera lineal, mantenint l'eficiència del planificador fins i tot amb un gran nombre d'habitacions disponibles.

Per tant, respecte a les nostres hipòtesis:

- En l'escenari de poques habitacions i moltes reserves, rebutgem H_0 i acceptem H_1, ja que el planificador demostra un comportament intel·ligent en maximitzar l'ocupació total.
- En l'escenari de moltes habitacions i poques reserves, rebutgem H_0 i acceptem H_1, ja que el planificador assigna totes les reserves disponibles de manera eficient, i el temps d'execució creix de manera lineal amb el nombre d'habitacions. Això confirma la seva capacitat per gestionar escenaris amb abundància de recursos, sense passar al pla exponencial.

3.1 Extensió 1

3.1.1 Domini

S'elimina el predicat compatible i totes les referències a aquest en les accions, ja que es considerarà que les reserves compatibles seran les que la capacitat de l'habitació sigui suficient per a les persones de la reserva.

3.1.2 Problemes

3.1.2.1 Problema 1: Dilema de l'optimització

Aquest problema demostra que el planificador és intel·ligent: ha de preferir assignar dues reserves curtes en lloc d'una reserva llarga si ocupen la mateixa habitació, ja qeu l'objectiu és maximitzar les assignacions.

Hipòtesi: el planificador serà capaç de maximitzar les assignacions, és a dir, demostra un comportament intel·ligent.

3.1.2.2 Problema 2: Reserves solapades

El planificador, maximitzant total-assignades, triarà dues reserves que no es solapin en dies (per exemple r1 i r3) i en descartarà una (per exemple r2), obtenint total-assignades = 2.

Hipòtesi: el planificador serà capaç de maximitzar les assignacions evitant solapaments.

3.2 Extensió 2

S'afegeix un nou predicat vol-orientacio per a les reserves, que indica l'orientació que es desitja per a l'habitació assignada. A l'acció assigar habitació comprova que l'habitació tingui l'orientació demanada per la

reserva. Com que volem maximitzar les assignacions, i és preferent assignar habitacions amb l'orientació demanada però no és imprescindible, fem que una reserva puntui 2 si es assigna una habitació amb l'orientació demanada i 1 si s'assigna una habitació amb una orientació diferent. Així, l'objectiu de maximitzar les assignacions es manté, però ara es prioritzen les assignacions que compleixin l'orientació demanada.

3.2.1 Domini

3.2.2 Problemes

Per validar l'Extensió 2, s'han dissenyat tres experiments sintètics que aïllen comportaments específics del planificador: capacitat d'ordenació (Exp 1), capacitat de filtratge per qualitat (Exp 2) i capacitat de sacrifici de preferències per maximitzar ocupació (Exp 3)."

3.2.2.1 Problema 1: El puzzle d'afinitats

Demostrar que el planificador és capaç d'ordenar i intercanviar reserves per aconseguir la màxima puntuació global, en lloc de fer assignacions greedy (agafar la primera habitació lliure encara que no tingui l'orientació correcta). Una assignació tonta (greedy) podria posar reserves "Sud" a habitacions "Nord" i viceversa. Això donaria 1 punt per reserva. Puntuació total: N. L'assignació intel·ligent ha de creuar-les perfectament: "Nord" amb "Nord" i "Sud" amb "Sud". El planificador ha de "descobrir" que val la pena buscar la combinació perfecta.

Hipòtesi: el planificador serà capaç de maximitzar la puntuació global tenint en compte les orientacions demanades.

3.2.2.2 Problema 2: Selecció VIP

Demostrar que, davant l'escassetat de recursos, el planificador sap prioritzar els clients que encaixen millor (els que donen 2 punts) i deixar fora o en pitjors habitacions els que no encaixen (els que donarien 1 punt), o simplement descartar els que aporten menys valor si no hi ha lloc per a tothom. Hi ha un coll d'ampolla: només K habitacions disponibles (totes, per exemple, orientació Nord). La meitat de les reserves volen Nord (match = 2 punts). L'altra meitat volen Sud (mismatch = 1 punt). Com que no hi ha lloc per a tothom (només hi ha K llocs), el planificador ha d'omplir l'hotel només amb les reserves que volen Nord. Puntuació òptima: $2 \times K$. Si s'equivoqués i agafés algú de Sud, perdria punts. Aquest experiment demostra que el sistema entén el "cost d'oportunitat".

Hipòtesi: el planificador serà capaç de prioritzar les reserves que aporten més valor en situacions d'escassetat de recursos.

3.2.2.3 Problema 3: L'Agent de Viatges Flexible

Demostrar que el sistema entén que l'orientació (preferència) és menys important que la capacitat (restricció dura), però que intenta satisfer totes dues quan pot. Hi ha N reserves i N habitacions. Totes les reserves tenen una preferència d'orientació que coincideix amb una habitació (match = 2 punts). Però algunes reserves tenen una capacitat que només encaixa amb una habitació de diferent orientació (mismatch = 1 punt). Per exemple, la reserva r1 necessita capacitat 4 i vol orientació Nord (habitació h1). Però l'habitació h1 té capacitat 2 (no serveix). L'única habitació amb capacitat 4 és h2, però té orientació Sud (mismatch). El planificador ha de triar entre no assignar r1 (0 punts) o assignar-la a h2 (1 punt). L'assignació òptima és sacrificar algunes preferències per satisfer totes les restriccions dures.

Hipòtesi: el planificador serà capaç de sacrificar preferències per maximitzar l'ocupació complint les restriccions dures.

3.3 Extensió 3

3.3.1 Domini

3.3.2 Problemes

3.3.2.1 Problema 1

3.3.2.2 Problema 2

3.4 Extensió 4

3.4.1 Domini

3.4.2 Problemes

3.4.2.1 Problema 1

3.4.2.2 Problema 2

4. Conclusions