Università della Calabria

Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e Sistemistica

---

Corso di Laurea Magistrale in Ingegneria Informatica

Relazione progetto Intelligenza Artificiale

Docenti:

Prof. Francesco **SCARCELLO**

Ing. Antonio **BONO**

Studenti:

Danilo FORTUGNO,
matr. 256483
Matteo GRECO,
matr. 252238
Vincenzo PRESTA,
matr. 252290

Anno Accademico 2023/2024

Sommario

Introduzione

Il presente elaborato proposto per il corso di Intelligenza Artificiale sviluppa tre punti:

1. la modellazione di un problema di pianificazione classica attraverso il linguaggio PDDL;
2. lo sviluppo e l'implementazione di un algoritmo di ricerca e della relativa euristica per risolvere specifiche istanze del problema;
3. utilizzo di azioni con durata ("durative action") e implementazione del problema in PlanSys2 con azioni simulate ("fake action").

L'elaborato è stato pertanto diviso in tre capitoli, ognuno riguardante uno dei tre punti.

Il contesto applicativo scelto è ispirato a un ambiente industriale, dove il sistema di pianificazione dovrà coordinare le attività di agenti robotici multipli, incaricati di consegnare materiali a varie stazioni di lavoro, ciascuna con requisiti specifici.

Il progetto è presente nella seguente repository di github:

https://github.com/VincenzoPresta/AI_Planning_Assignment.git

La struttura è cosi definita:

- Task1: è presente il file relativo al dominio richiesto nel primo punto del problema.
- Task2: All'interno della prima cartella sono stati inseriti i domini relativi all'istanza1 con il corrispondente problema. Per l'istanza 2 è sono presenti due differenti versioni, una con i numeric fluents l'altra invece priva dei tali.
  All'interno della cartella Planner sono presenti i file relativi, appunto, al Planner; in particolare nella cartella *PDDL* sono inseriti i domini e i problemi utilizzati e in aggiunta anche delle cartelle che permettono l'avvio del problema con l'algoritmo Fast Forward. Nella cartella *Results* sono inseriti tutti i risultati ottenuti tramite il planner con i corrispondenti consumi.
  Infine, nella cartella src è inserito il planner relativo all'istanza 1 e all'istanza 2 (Il planner dell'istanza 1 funziona anche per l'istanza 2, ma nell'istanza 2 è presenta una miglioria adatto per quest'ultima)
- Task3: Sono inserite due sottocartelle: La cartella 3.1 contiene domini e problemi relativi al sopracitato punto. La cartella 3.2 invece presenta i risultati ottenuti (inseriti

nella cartella plans) e i file PDDL nell'omonima cartella.
Nella cartella *Screenshots* è presente il risultato ottenuto tramite l'esecuzione del 3.2 che, a causa delle elevate proporzioni dello screen, non era visualizzabile in maniera leggibile sul PDF.

Capitolo 1: Task 1

1.1 Modellazione del Dominio

La fase iniziale prevede la creazione di un dominio che simula un ambiente di produzione industriale. In questo scenario, diversi tipi di agenti robotici (*agent)* vengono impiegati e coordinati per trasportare e consegnare scatole (*box*) contenenti componenti (*content)* alle varie postazioni di lavoro (*workstation*).

Sono state fatte diverse assunzioni:

- Ogni location può ospitare o meno una o più workstation;
- Se l'agent robotico è carico (*loaded*) non può effettuare operazioni di riempimento o di svuotamento;
- Disponibilità limitata dei contenuti: Se il numero di oggetti disponibili non è sufficiente per soddisfare le richieste delle workstation, il problema risulterà irrisolvibile.

Tipi

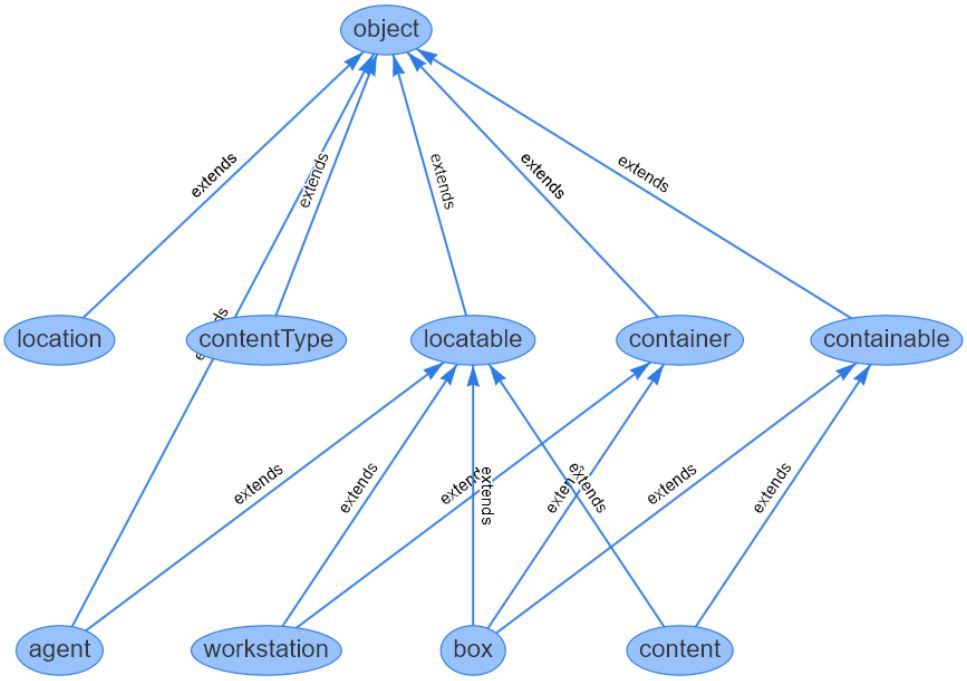
Al fine di modellare il dominio, sono stati definiti i seguenti tipi:

```
(:types

location agent contentType - object
workstation box content agent - locatable
workstation box - container
content box - containable
)
```

- ***location***: Rappresenta le locazioni all'interno dell'ambiente di produzione.
- ***locatable***: Rappresenta tutti gli oggetti che possono essere localizzati all'interno di una location. Gli oggetti di tipo *locatable* possono essere associati a una location per indicare dove si trovano nel sistema.
- ***agent***: Rappresenta un'agent robotico (robot, drone, ecc.) che può eseguire azioni all'interno del dominio. Un agent è anche di tipo *locatable*, quindi può essere localizzato in una specifica *location*.
- ***contentType***: Definisce le diverse categorie di contenuti che possono essere caricati nelle *box* o utilizzati nelle *workstation*. Serve a caratterizzare il tipo di contenuto presente.

- ***workstation***: Rappresenta una stazione di lavoro dove vengono svolte operazioni specifiche. Una workstation è di tipo *locatable*, quindi ha una locazione definita, ed è anche un *container*, poiché può contenere altri oggetti come *content* o *box*.
- ***box***: Rappresenta una box che può contenere oggetti o materiali, identificati come *content*. Una box è di tipo *locatable*, quindi può essere localizzata, e anche di tipo *container*, poiché può contenere *content*.
- ***content***: Rappresenta il contenuto che può essere inserito in una *box* o in una *workstation*. Il *content* è di tipo *containable*, quindi può essere contenuto in un *container* come una box o una workstation.
- ***containable***: Un tipo che rappresenta ciò che può essere contenuto in un *container*, come una *box* in una *workstation* o un *content* in una *box*.
- ***container***: Un tipo generico per gli oggetti che possono contenere altri oggetti. Le *workstation* e le *box* sono esempi di *container*.



Gerarchia dei tipi di oggetti

Predicati

```
(:predicates
        (at ?locatable - locatable ?location - location )
        (contain ?container - container ?containable - containable)
        (connected ?loc1 - location ?loc2 - location)
        (workstation-has-type ?workstation - workstation ?contentType - contentType)
        (is-type ?content - content ?contentType - contentType)
        (is-empty ?box - box)
        (loaded ?agent – agent ?box -box)
        (free ?agent – agent)
    )
```

- **at:** Questo predicato indica la locazione corrente di un oggetto *locatable* all'interno dell'ambiente. Esso afferma che l'oggetto di tipo *locatable* (come un *agent*, una *workstation* o una *box*) si trova in una specifica *location*.
- **contain:** Esprime la relazione di contenimento tra un *container* (come una *workstation* o una *box*) e un oggetto *containable* (come un *content*). Esso afferma che il *container* contiene un certo oggetto *containable*.
- **connected:** Rappresenta una connessione tra due *location* all'interno dell'ambiente. Esso afferma che le due posizioni specificate (*loc1* e *loc2*) sono collegate tra loro, garantendo la possibilità di muoversi "direttamente" dall'una all'altra.
- **workstation-has-type** Associa una *workstation* a un determinato *contentType*, indicando che la workstation necessita di un tipo specifico di contenuto.
- **is-type:** Identifica il tipo specifico di un certo *content*, associandolo a un *contentType*. Esso afferma che un certo *content* è di un particolare tipo.
- **is-empty:** Questo predicato si applica a una *box* e verifica se essa è vuota.

Azioni

L'ultimo passaggio nella definizione del dominio consiste nel modellare le azioni, che rappresentano le operazioni eseguibili dagli agenti robotici per raggiungere gli obiettivi prefissati. Un'azione in PDDL è caratterizzata da tre elementi fondamentali:

- **Parametri**: Sono le variabili che l'azione utilizza e rappresentano gli oggetti coinvolti nell'azione stessa. Questi parametri permettono di specificare su quali entità l'azione verrà applicata.
- **Precondizioni**: Sono le condizioni che devono essere vere nel dominio affinché l'azione possa essere eseguita. Le precondizioni definiscono lo stato richiesto dell'ambiente prima che l'azione possa aver luogo.

- **Effetti**: Rappresentano le modifiche che l'azione provoca nell'ambiente una volta eseguita. Gli effetti descrivono il cambiamento dello stato dell'ambiente come conseguenza dell'azione.

Per il presente dominio, sono state definite le seguenti azioni:

- **fill-box-from-location**

```
(:action fill-box-from-location
      :parameters (?agent - agent ?box - box ?content - content ?location - location )
      :precondition (and
                      (at ?agent ?location)
                      (at ?box ?location)
                      (at ?content ?location)
                      (is-empty ?box)
      )
      :effect (and
                  (not (is-empty ?box))
                  (not (at ?content ?location))
                  (contain ?box ?content)
      )
  )
```

L'azione "*fill-box-from-location*", descrive l'operazione di riempimento di una *box* con un *content* in una *location*.

- Gli oggetti coinvolti nell'azione (*parameters*):
  - *Agent*: l'agent che esegue l'azione;
  - *Box*: la box che viene riempita;
  - *Content*: il contenuto che verrà inserito nella *box;*
  - *Location*: La locazione da cui il contenuto sarà prelevato e dove *agent*, *box* e *content* sono attualmente situati;

- Condizioni che devono essere soddisfatte affinché l'azione possa essere eseguita (*preconditions*):

  - L'*agent* deve trovarsi nella stessa posizione del *content* e della *box*;
  - La *box* deve essere nella stessa posizione del *content* e dell'*agent*;
  - Il *content* deve trovarsi nella stessa posizione della *box* e dell'*agent*;
  - La *box* deve essere vuota.

- Modifiche che l'azione provoca (*effects*):

  - La *box* non sarà più vuota;
  - Il *content* non sarà più nella posizione iniziale;

- La *box* conterrà ora il *content*.

In sintesi, L'azione '*fill-box-from-location*' consente a un *agent* di riempire una *box* con un *content*.
Per eseguire questa azione, è necessario che l'agent, la box e il contenuto si trovino tutti nella stessa posizione e che la box sia vuota. Dopo che l'azione è stata completata, la *box* non sarà più vuota e conterrà il *content* che era precedentemente nella *location*.

- **fill-box-from-workstation**

```
 (:action fill-box-from-workstation
        :parameters (?agent - agent ?box - box ?content - content ?workstation -
workstation ?contentType - contentType ?location - location)
        :precondition (and
                        (at ?agent ?location)
                        (at ?workstation ?location)
                        (contain ?workstation ?box)
                        (contain ?workstation ?content)
                        (is-type ?content ?contentType)
                        (is-empty ?box)
        )
        :effect (and
                (not (is-empty ?box))
                (not (contain ?workstation ?content))
                (not (workstation-has-type ?workstation ?contentType))
                (contain ?box ?content)
        )
    )
```

L'azione "*fill-box-from-location*", descrive l'operazione di riempimento di una *box* con un *content* in una workstation.

- Gli oggetti coinvolti nell'azione (*parameters*):
  - *agent*: l'agent che esegue l'azione;
  - *box*: la box che viene riempita;
  - *content*: il contenuto che verrà inserito nella *box;*
  - *workstation*: La workstation da cui il contenuto sarà prelevato;
  - *contentType*: Il tipo di *content* che deve essere caricato;
  - *location*: La posizione in cui si trovano *l'agent* e la *workstation*.

- Condizioni che devono essere soddisfatte affinché l'azione possa essere eseguita (*preconditions*):
    - L'*agent* deve trovarsi nella stessa posizione della *workstation*;
    - La *workstation* deve essere nella stessa posizione dell'*agent*;
    - La *workstation* deve contenere la *box*;
    - La *workstation* deve contenere il *content* che deve essere caricato nella *box*;
    - Il *content* deve essere del tipo specificato;
    - La *box* deve essere vuota prima di poter essere riempita.

- Modifiche che l'azione provoca (*effects*):
    - Dopo l'azione, la box non sarà più vuota;
    - Il contenuto non sarà più nella workstation;
    - La workstation non avrà più il tipo di contenuto specificato;
    - La box conterrà ora il contenuto.

In sintesi, L'azione *fill-box-from-workstation* consente a un *agent* di trasferire un *content* da una *workstation* a una *box*.

Per eseguire questa azione, è necessario che l'*agent* e la *workstation* si trovino nella stessa posizione e che la *workstation* contenga sia la *box* che il *content*. Inoltre, il *content* deve essere del tipo specificato e la *box* deve essere vuota. Dopo che l'azione è stata completata, la *box* non sarà più vuota e conterrà il *content*, mentre la *workstation* non avrà più quel *content*.

- **empty-box-workstation**

```
(:action empty-box-workstation
        :parameters (?agent - agent ?box - box ?content - content ?contentType -
contentType ?workstation - workstation ?location - location)
        :precondition (and
                            (at ?agent ?location)
                            (contain ?workstation ?box)
                            (at ?workstation ?location)
                            (contain ?box ?content)
                            (is-type ?content ?contentType)
        )
        :effect (and
                    (not (contain ?box ?content))
                    (is-empty ?box)
                    (contain ?workstation ?content)
                    (workstation-has-type ?workstation ?contentType)
        )
    )
```

L'azione riguarda lo svuotamento di una *box* e il trasferimento del *content* alla *workstation*.

- Gli oggetti coinvolti nell'azione (*parameters*):
  - ***agent***: l'*agent* che esegue l'azione;
  - ***box***: la *box* da svuotare;
  - ***content***: Il *content* da trasferire alla *workstation*;
  - ***contentType***: Il tipo di *content* che viene trasferito alla *workstation*;
  - ***workstation***: La *workstation* in cui il *content* sarà trasferito;
  - ***location***: La posizione in cui si trovano l'*agent*, la *box* e la *workstation*.
- Condizioni che devono essere soddisfatte affinché l'azione possa essere eseguita (*preconditions*):
  - L'*agent* deve trovarsi nella stessa posizione della *workstation*.
  - La *box* deve essere contenuta nella *workstation*.
  - La workstation deve essere nella stessa posizione dell'*agent*.
  - La *box* deve contenere il *content* che deve essere trasferito.
  - Il *content* deve essere del tipo specificato (*contentType*).
- Modifiche che l'azione provoca (*effects*):
  - Dopo l'azione, la *box* non conterrà più il *content*.
  - La *box* sarà vuota dopo l'azione.
  - Il *content* sarà trasferito alla *workstation*.
  - La *workstation* avrà il tipo di *content* specificato.

In sintesi, L'azione *empty-box-workstation* consente a un *agent* di trasferire il *content* da una *box* a una *workstation*.

Per eseguire questa azione, è necessario che l'*agent*, la *box* e la *workstation* si trovino nella stessa posizione o che la *box* sia contenuta nella *workstation*. Inoltre, il *content* deve essere presente nella *box* e deve essere del tipo specificato. Dopo che l'azione è completata, la *box* sarà vuota e il *content* sarà trasferito alla *workstation*, che avrà ora associato il tipo di contenuto specificato.

- **empty-box-location**

```
    (:action empty-box-location
        :parameters (?agent - agent ?box - box ?content - content ?location -
location)
        :precondition (and
                            (at ?agent ?location)
                            (at ?box ?location)
                            (contain ?box ?content)
        )
        :effect (and
                    (not (contain ?box ?content))
                    (is-empty ?box)
                    (at ?content ?location)
        )
    )
```

L'azione riguarda lo svuotamento di una *box* e il trasferimento dello stesso alla *location*.

- Gli oggetti coinvolti nell'azione (*parameters*):
  - **agent**: l'*agent* che esegue l'azione;
  - **box**: la *box* da svuotare;
  - **content**: Il *content* da trasferire nella *location*;
  - **location**: la location in cui si trovano l'*agent* e la *box*.

- Condizioni che devono essere soddisfatte affinché l'azione possa essere eseguita (*preconditions*):
  - L'agent deve trovarsi nella stessa posizione della box e del contenuto;
  - La box deve essere nella stessa posizione dell'agent e del contenuto;
  - La box deve contenere il contenuto che deve essere trasferito nella location.

- Modifiche che l'azione provoca (*effects*):
  - Dopo l'azione, la *box* non conterrà più il *content*;
  - La *box* sarà vuota dopo l'azione;
  - Il *content* sarà trasferito alla *workstation*;
  - Il *content* sarà posizionato nella *location* specificata.

L'azione *empty-box-location* consente a un *agent* di trasferire il *content* da una *box* a una *location*.
Per eseguire questa azione, è necessario che l'*agent*, la *box* e il *content* si trovino tutti nella stessa posizione. Inoltre, la *box* deve contenere il *content* che deve essere trasferito. Dopo che l'azione è completata, la *box* sarà vuota e il *content* sarà posizionato nella *location* specificata.

- **pick-up-from-workstation**

```
   (:action pick-up-from-workstation
        :parameters (?agent - agent ?box - box ?workstation - workstation
?location - location)
        :precondition (and
                            (free ?agent)
                            (at ?agent ?location)
                            (at ?workstation ?location)
                            (contain ?workstation ?box)
        )
        :effect (and
                    (not (free ?agent))
                    (not (contain ?workstation ?box))
                    (loaded ?agent ?box)
        )
    )
```

L'azione riguarda il prelievo di una *box* da una *workstation* da parte di un *agent*.

- Gli oggetti coinvolti nell'azione (*parameters*):
  - **agent**: l'*agent* che esegue l'azione.
  - **box**: la *box* da svuotare.
  - **workstation**: La *workstation* da cui la *box* sarà prelevata
  - **location**: la *location* in cui si trovano l'*agent*, la *box* e la *workstation*.
- Condizioni che devono essere soddisfatte affinché l'azione possa essere eseguita (*preconditions*):
  - L'*agent* deve essere libero, cioè non deve essere impegnato in altre attività.
  - L'*agent* deve trovarsi nella stessa posizione della *workstation* e della *box*.
  - La *workstation* deve essere nella stessa posizione dell'*agent* e della *box*.
  - La *workstation* deve contenere la *box* che deve essere prelevata.

- Modifiche che l'azione provoca (*effects*):
  - Dopo l'azione, l'agent non sarà più libero, in quanto avrà preso la box e quindi sarà considerato occupato.
  - La workstation non conterrà più la box
  - L'agent sarà considerato carico della box, indicando che la box è ora in possesso dell'agent.

L'azione *pick-up-from-workstation* permette a un agent di prelevare una *box* da una *workstation*.

Per eseguire questa azione, è necessario che l'*agent* e la *workstation* si trovino nella stessa posizione e che la *workstation* contenga la *box* da prelevare. Dopo che l'azione è completata, la *box* non sarà più presente nella *workstation* e l'*agent* risulterà carico.

- **move**

```
    (:action move
        :parameters (?agent - agent ?from - location ?to - location)
        :precondition (and
                        (connected ?from ?to)
                        (at ?agent ?from)
        )
        :effect (and
                    (not (at ?agent ?from))
                    (at ?agent ?to)
        )
    )
```

L'azione riguarda il movimento di un *agent* da una posizione a un'altra.

- Gli oggetti coinvolti nell'azione (*parameters*):
  - **agent**: l'*agent* che esegue l'azione.
  - **from**: La *location* di partenza da cui l'*agent* deve muoversi.
  - **to:** La *location* di destinazione verso cui l'*agent* deve muoversi.
- Condizioni che devono essere soddisfatte affinché l'azione possa essere eseguita (*preconditions*):
  - Le due *location* devono essere collegate, ovvero esiste una connessione tra la *location* di partenza e quella di destinazione
  - L'*agent* deve trovarsi attualmente nella *location* di partenza.
- Modifiche che l'azione provoca (effects):
  - Dopo l'azione, l'*agent* non sarà più nella *location* di partenza.
  - Dopo l'azione, l'*agent* sarà nella *location* di destinazione.

In sintesi, l'azione *move* consente a un *agent* di spostarsi da una *location* a un'altra. Per eseguire questa azione, è necessario che le due *location* siano collegate e che l'*agent* si trovi nella posizione di partenza. Dopo che l'azione è completata, l'*agent* non sarà più nella *location* di partenza, ma si troverà nella *location* di destinazione.

- **deliver-to-location**

```
(:action deliver-to-location
       :parameters (?agent - agent ?box - box ?location - location)
       :precondition (and
                           (at ?agent ?location)
                           (loaded ?agent ?box)
       )
       :effect (and
                   (not (loaded ?agent ?box))
                   (free ?agent)
                   (at ?box ?location)
       )
```

L'azione riguarda la consegna di una *box* a una *location* specifica da parte di un *agent*.

- Gli oggetti coinvolti nell'azione (*parameters*):
  - **agent**: l'*agent* che esegue l'azione.
  - **box**: la *box* che deve essere consegnata.
  - **location**: la *location* in cui la *box* deve essere consegnata.
- Condizioni che devono essere soddisfatte affinché l'azione possa essere eseguita (*preconditions*):
  - *L'agent* deve trovarsi nella *location* di destinazione.
  - *L'agent* deve essere attualmente carico della *box*, cioè deve avere la *box* con sé.
- Modifiche che l'azione provoca (*effects*):
  - Dopo l'azione, *l'agent* non sarà più carico della *box*.
  - Dopo l'azione, *l'agent* sarà considerato *free*, cioè non sarà più occupato con la box.
  - Dopo l'azione, la *box* sarà posizionata nella *location* specificata.

L'azione *deliver-to-location* consente a un *agent* di consegnare una *box* a una determinata *location*. Per eseguire questa azione, è necessario che *l'agent* sia nella *location* di destinazione e che abbia la *box* con sé. Dopo che l'azione è completata, *l'agent* non sarà più carico della box e sarà considerato *free*, mentre la *box* sarà posizionata nella *location* specificata.

- **deliver-to-workstation**

```
    (:action deliver-to-workstation
        :parameters (?agent - agent ?workstation - workstation
?location - location ?box - box)
        :precondition (and
                            (at ?agent ?location)
                            (at ?workstation ?location)
                            (loaded ?agent ?box)
        )
        :effect (and
                    (not (loaded ?agent ?box))
                    (free ?agent)
                    (contain ?workstation ?box)
        )
    )
```

L'azione riguarda la consegna di una *box* a una *workstation* da parte di un *agent*.

- Gli oggetti coinvolti nell'azione (parameters):
  - **agent**: *L'agent* che esegue l'azione di consegna.
  - **workstation**: La *workstation* a cui la *box* deve essere *consegnata*.
  - **location**: La locazione in cui si trovano *l'agent* e la *workstation*.
  - **box**: La *box* che deve essere consegnata.
- Condizioni che devono essere soddisfatte affinché l'azione possa essere eseguita (*preconditions*):
  - *L'agent* deve trovarsi nella stessa posizione della *workstation*.
  - La *workstation* deve essere nella stessa posizione *dell'agent*.
  - *L'agent* deve essere attualmente carico della *box*, cioè deve avere la *box* con sé.
- Modifiche che l'azione provoca (*effects*):
  - Dopo l'azione, *l'agent* non sarà più carico della *box*.
  - Dopo l'azione, *l'agent* sarà considerato *free*, cioè non sarà più occupato con la *box*.
  - Dopo l'azione, la *box* sarà posizionata presso la *workstation*, indicando che la *workstation* ora contiene la *box*.

L'azione *'deliver to location'* consente a un *agent* di consegnare una *box* a una *workstation*. Per eseguire questa azione, è necessario che *l'agent* e la *workstation* si trovino nella stessa posizione e che *l'agent* sia carico della *box* da consegnare. Dopo che l'azione è completata, *l'agent* non sarà più carico della box e sarà considerato *free*, mentre la *box* sarà ora contenuta nella *workstation*.

Capitolo 2: Task 2

In questa sezione, si procederà con lo sviluppo di un planner personalizzato, finalizzato alla risoluzione delle due istanze fornite.

2.1 Istanza 1

2.1.1 Dominio

Per la prima istanza, si fa riferimento al dominio descritto in precedenza. Lo stato iniziale è così descritto:

- Posizionamento delle *box*: tutte le *box* sono inizialmente collocate in un'unica *location* chiamata "*warehouse*".
- Posizionamento dei contenuti: tutti i *content* si trovano nella "*warehouse*".
- Nella "warehouse" non sono presenti *workstation*.
- Posizionamento degli *agent*: un singolo *agent* è posizionato nella "*warehouse*".
- Obiettivo (goal): l'obiettivo è consegnare le *box* alle *workstation* secondo le loro richieste.

2.1.2 Problema

Per quanto riguarda l'istanza del problema, è stato scelto di utilizzare:

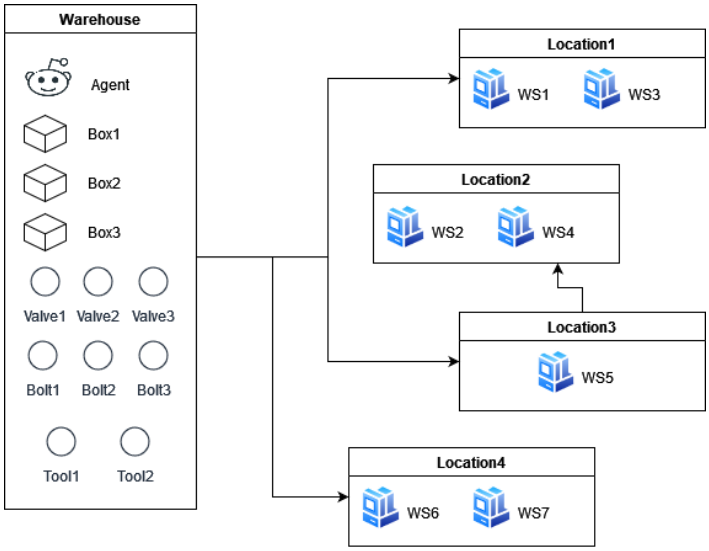


Diagramma descrittivo del problema dell'istanza 1

- 1 *agent*, localizzato nella *warehouse*;
- 3 *box*, localizzate nella *warehouse*
- 3 tipi di *content*: *bolt*, *valve*, *tool*
- 3 oggetti di tipo *bolt*
- 3 oggetti di tipo *valve*
- 2 oggetti di tipo *tool*
- 5 *location*: *warehouse, loc1, loc2, loc3, loc4*
- 7 *workstation* così localizzate:
    - *ws1* e *ws3* in *loc1*
    - *ws2* e *ws4* in *loc2*
    - ws5 in loc3
    - *ws6* e *ws7* in *loc4*
- Connessioni tra location:
    - *warehouse* connessa con *loc1*, *loc3* e *loc4*
    - *loc2* connessa con *loc3*

Il **goal** consiste nell'avere in *ws1* oggetti di tipo *bolt*, in *ws2* oggetti di tipo *valve* e in *ws3* oggetti di tipo *valve* e oggetti di tipo *tool*.

2.2 Instanza 2

2.2.1 Dominio

Nel caso della seconda istanza, sono state introdotte nel dominio le seguenti estensioni rispetto al caso precedente:

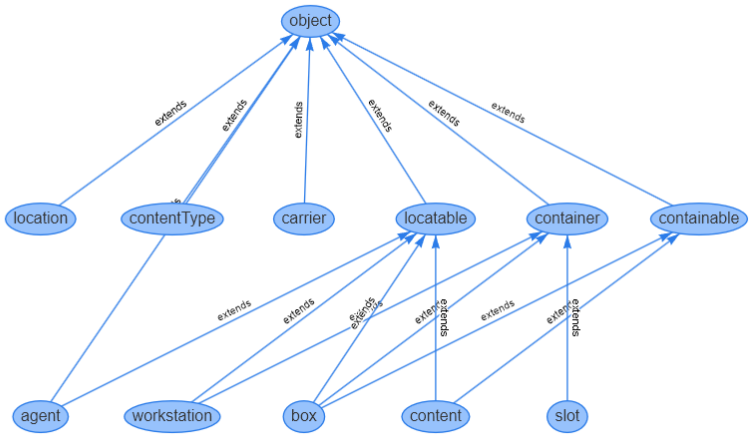
- ***Carrier* per ogni *agent***: Ogni *agent* è dotato di un unico *carrier* con una capacità di carico massima che può variare. Gli *agent* possono quindi caricare più *box* negli *slot* del loro *carrier* fino a raggiungere la capacità massima;
- ***Slot* per ogni *carrier***: ogni *carrier* è dotato di uno o più *slot* che le *box* possono occupare durante il trasporto;
- **Tracciamento del carico**: Per ciascun agent, viene monitorato:
    - Quali *box* sono state caricate sul *carrier*.
    - Il numero totale di *box* presenti su ogni *carrier*.

A tal fine, sono stati modificati tipi e predicati rispetto all'istanza 1, inserendo i tipi "*carrier*" e "*slot*":

Tipi

```
(:types

  location agent contentType carrier - object
  workstation box content agent - locatable
  workstation box slot - container
  content box - containable
)
```



Gerarchia dei tipi nel dominio dell'istanza 1

Predicati

```
   (:predicates
        (at ?locatable - locatable ?location - location )
        (contain ?container - container ?containable -
containable)
        (connected ?loc1 - location ?loc2 - location)
        (workstation-has-type ?workstation - workstation
?contentType - contentType)
        (is-type ?content - content ?contentType -
contentType)
        (is-empty ?container - container)
        ;Instance no. 2
        (agent-has-carrier ?agent - agent ?carrier -
carrier)
        (carrier-has-slot ?carrier - carrier ?slot - slot)
    )
```

I predicati aggiunti sono:

- **agent-has-carrier:** true se l'*agent* ha a disposizione un *carrier*;
- **carrier-has-slot:** *true* se il *carrier* contiene lo *slot*;

I predicati eliminati sono:

- **free**
- **loaded**

Visto l'utilizzo di un *carrier* cade l'assunzione sull'impossibilità di effettuare *fill* o *empty* qualora *l'agent* stia trasportando una *box*, dunque viene meno l'utilità dei predicati "*free*" e "*loaded*" in quanto si presuppone che a ogni *agent* venga associato un unico *carrier*.

Azioni

Al fine di adattare il nuovo dominio, sono stati effettuati diversi cambiamenti nelle azioni. Di seguito sono riportate e descritte:

- **pick-up-from-location**

```
 (:action pick-up-from-location
        :parameters (?agent - agent ?box - box ?location - location
?carrier - carrier ?slot - slot)
        :precondition (and
                              (at ?agent ?location)
                              (at ?box ?location)
                              (agent-has-carrier ?agent ?carrier)
                              (carrier-has-slot ?carrier ?slot)
                              (is-empty ?slot)
        )
        :effect (and
                    (not (at ?box ?location))
                    (not (is-empty ?slot))
                    (contain ?slot ?box)
        )
    )
```

L'azione permette a un *agent* di prelevare una *box* da una *location* specifica e caricarla su un *carrier*.

- Gli oggetti coinvolti nell'azione (*parameters*):
    - **agent**: *L'agent* che esegue l'azione di prelievo.
    - **location**: La posizione in cui si trovano *l'agent* e la *box*.
    - **box**: La *box* che deve essere prelevata.
    - **carrier**: Il *carrier* su cui verrà caricata la *box*.
    - **slot**: Lo *slot* specifico del *carrier* in cui la *box* verrà posizionata.
- Condizioni che devono essere soddisfatte affinché l'azione possa essere eseguita (*preconditions*):
    - *L'agent* deve trovarsi nella stessa posizione della *box*.
    - La *box* deve trovarsi nella stessa posizione *dell'agent*.
    - *L'agent* deve essere associato a un *carrier*.
    - Il *carrier* deve avere uno *slot* disponibile in cui la *box* può essere caricata.

    - Lo *slot* del *carrier* deve essere vuoto.
- Modifiche che l'azione provoca (*effects*):
    - Dopo l'azione, la *box* non si troverà più nella *location*.
    - Dopo l'azione, lo *slot* nel *carrier* non sarà più vuoto.
    - La *box* sarà ora contenuta nello *slot* del *carrier* specificato.

L'azione *pick-up-from-location* consente a un *agent* di prelevare una *box* da una *location* specifica e di caricarla in un slot vuoto del suo *carrier*.
Per eseguire questa azione, è necessario che *l'agent* e la *box* si trovino nella stessa *location*, che *l'agent* disponga di un *carrier* con uno *slot* vuoto, e che lo *slot* sia disponibile. Una volta completata l'azione, la *box* sarà caricata nel *carrier*, e non sarà più presente nella *location*

- **pick-up-from-workstation**

```
   (:action pick-up-from-workstation
        :parameters (?agent - agent ?workstation - workstation
?location - location ?box - box  ?carrier - carrier ?slot - slot)
        :precondition (and
                    (at ?agent ?location)
                    (at ?workstation ?location)
                    (contain ?workstation ?box)
                    (agent-has-carrier ?agent ?carrier)
                    (carrier-has-slot ?carrier ?slot)
                    (is-empty ?slot)
        )
        :effect (and
            (not (contain ?workstation ?box))
            (not (is-empty ?slot))
            (contain ?slot ?box)
        )
    )
    )
```

L'azione permette a un *agent* di prelevare una *box* da una *workstation* e caricarla su un *carrier*.

- Gli oggetti coinvolti nell'azione (*parameters*):
  - **agent**: *L'agent* che esegue l'azione di prelievo.
  - **workstation**: La *workstation* da cui la *box* viene prelevata.
  - **location**: La *location* in cui si trovano *l'agent* e la *workstation*.
  - **box**: La *box* che deve essere prelevata.
  - **carrier**: Il *carrier* su cui verrà caricata la *box*.
  - **slot**: Lo *slot* specifico del *carrier* in cui la *box* verrà posizionata.
- Condizioni che devono essere soddisfatte affinché l'azione possa essere eseguita (*preconditions*):
  - *L'agent* deve trovarsi nella stessa *location* della workstation.
  - La *workstation* deve contenere la *box* da prelevare.
  - *L'agent* deve essere associato a un *carrier***.**
  - Il *carrier* deve avere uno *slot* disponibile in cui la *box* può essere caricata.
  - Lo slot del *carrier* deve essere vuoto.

- Modifiche che l'azione provoca (*effects*):
    - Dopo l'azione, la *workstation* non conterrà più la *box*.
    - Dopo l'azione, lo *slot* nel *carrier* non sarà più vuoto.
    - La *box* sarà ora contenuta nello *slot* del *carrier* specificato.

L'azione *pick-up-from-workstation* permette a un *agent* di prelevare una *box* da una *workstation* e di caricarla in uno *slot* vuoto del suo *carrier*. Per poter eseguire questa azione, *l'agent* e la *workstation* devono trovarsi nella stessa posizione, la *workstation* deve contenere la *box*, e il *carrier dell'agent* deve avere uno *slot* vuoto disponibile. Una volta eseguita l'azione, la *box* sarà trasferita dalla *workstation* al *carrier*, liberando lo *slot* della *workstation* e occupando quello del *carrier*.

- **Deliver-to-location**

```
  (:action deliver-to-location
        :parameters (?agent - agent ?location - location ?box - box
?carrier - carrier ?slot - slot)
        :precondition (and
                            (at ?agent ?location)
                            (agent-has-carrier ?agent ?carrier)
                            (carrier-has-slot ?carrier ?slot)
                            (contain ?slot ?box)
        )
        :effect (and
                    (not (contain ?slot ?box))
                    (is-empty ?slot)
                    (at ?box ?location)
        )
    )
```

L'azione permette a un *agent* di consegnare una *box* in una specifica *location*, scaricandola dal suo *carrier*.

- Gli oggetti coinvolti nell'azione (*parameters*):
    - **agent**: *L'agent* che esegue l'azione di consegna.
    - **location**: La *location* in cui la *box* deve essere consegnata.
    - **box**: La *box* che deve essere consegnata.

- **carrier**: Il *carrier dell'agent* in cui è posizionata la *box*.
- **slot**: Lo *slot* specifico del *carrier* in cui la *box* è posizionata.

- Condizioni che devono essere soddisfatte affinché l'azione possa essere eseguita (*preconditions*):
  - *L'agent* deve trovarsi nella posizione in cui la box verrà consegnata
  - *L'agent* deve essere associato a un *carrier***.**
  - Il *carrier* deve avere uno slot in cui è posizionata la box.
  - Lo slot specificato deve contenere la box da consegnare.
- Modifiche che l'azione provoca (*effects*):
  - Dopo l'azione, lo *slot* nel *carrier* non conterrà più la *box*.
  - Lo *slot* nel *carrier* sarà vuoto dopo la consegna.
  - La *box* sarà ora posizionata nella *location* specificata.

L'azione *deliver-to-location* permette a un *agent* di scaricare una *box* da uno *slot* del suo *carrier* e di posizionarla in una specifica *location*.
Affinché questa azione possa essere eseguita, *l'agent* deve trovarsi nella *location* corretta, deve avere un *carrier* con uno *slot* contenente la *box* da consegnare. Dopo l'esecuzione dell'azione, la *box* sarà posizionata nella *location* e lo *slot* del *carrier* sarà vuoto.

- ## Deliver-to-workstation

```
   (:action deliver-to-workstation
        :parameters (?agent - agent ?workstation - workstation
?location - location ?box - box ?carrier - carrier ?slot - slot)
        :precondition (and
                            (at ?agent ?location)
                            (at ?workstation ?location)
                            (agent-has-carrier ?agent ?carrier)
                            (carrier-has-slot ?carrier ?slot)
                            (contain ?slot ?box)
        )
        :effect (and
                    (not (contain ?slot ?box))
                    (is-empty ?slot)
                    (contain ?workstation ?box)
        )
    )

)
```

L'azione permette a un *agent* di consegnare una *box* in una specifica *workstation*, scaricandola dal suo carrier.

- Gli oggetti coinvolti nell'azione (*parameters*):
    - **agent**: *l'agent* che esegue l'azione di consegna.
    - **workstation**: la *workstation* a cui la *box* deve essere consegnata.
    - **location:** la posizione in cui si trovano sia *l'agent* che la *workstation*.
    - **box**: La *box* che deve essere consegnata.
    - **carrier**: Il *carrier* dell'agent in cui è posizionata la *box*.
    - **slot**: Lo *slot* specifico del *carrier* in cui la box è posizionata.
- Condizioni che devono essere soddisfatte affinché l'azione possa essere eseguita (*preconditions*):
    - *L'agent* deve trovarsi nella stessa posizione della *workstation*.
    - La *workstation* deve trovarsi nella stessa posizione *dell'agent*.
    - *L'agent* deve essere associato a un *carrier*.
    - Il *carrier* deve avere uno *slot* in cui è posizionata la *box*.
    - Lo *slot* specificato deve contenere la *box* da consegnare.
- Modifiche che l'azione provoca (*effects*):
    - Dopo l'azione, lo *slot* nel *carrier* non conterrà più la *box*.
    - Lo slot nel *carrier* sarà vuoto dopo la consegna.
    - La *box* sarà ora posizionata nella *location* specificata.

L'azione *deliver-to-workstation* permette a un *agent* di scaricare una *box* da uno *slot* del suo carrier e di posizionarla in una *workstation* specifica.
Per eseguire questa azione, *l'agent* e la *workstation* devono trovarsi nella stessa *location* e *l'agent* deve avere un *carrier* con uno slot contenente la *box* da consegnare.
Dopo l'esecuzione dell'azione, la box sarà trasferita dallo *slot* del *carrier* alla *workstation*, lasciando lo *slot* del *carrier* vuoto.

2.2.2 Problema

Per il problema relativo all'istanza 2, sono state definite due istanze, che differiscono tra loro per la complessità. Di seguito verranno indicate come "istanza facile" e "istanza difficile".

Istanza facile

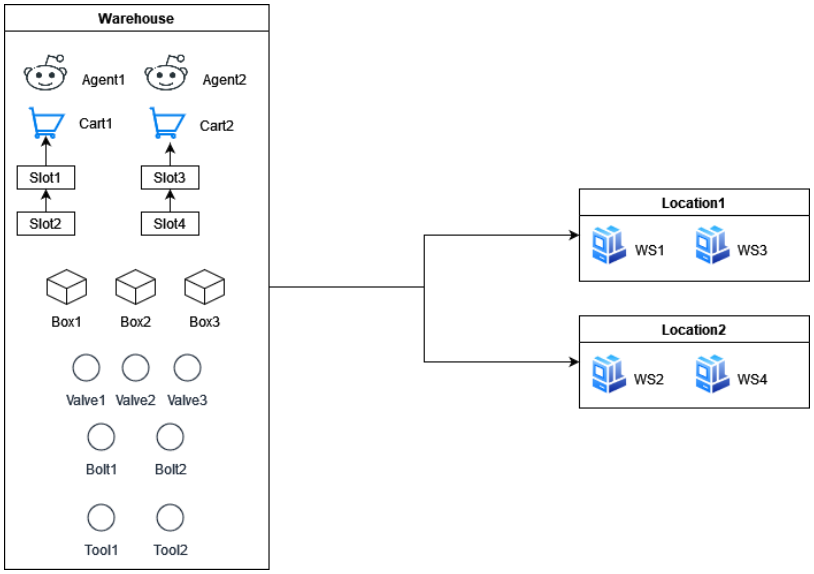


Diagramma descrittivo del problema "facile" dell'istanza 2

- 2 *agent*, localizzati nella *warehouse*
- 2 *cart*, *cart1* per *agent1* e *cart2* per *agent2*
- Ogni *cart* contiene 2 *slot*, *slot1* e *slot2* per il *cart1*, *slot3* e *slot4* per il cart2
- 3 *box*, localizzate nella *warehouse*
- 3 tipi di contenuto: *bolt, valve, tool*
- 2 oggetti di tipo *bolt*
- 3 oggetti di tipo *valve*
- 2 oggetti di tipo *tool*
- 3 location: warehouse, *loc1*, *loc2*
- 4 *workstation* così localizzate:
    - *ws1* e *ws3* in *loc1*
    - *ws2* e *ws4* in *loc2*
- Connessioni tra *location*:
    - warehouse connessa con *loc1, loc2 e loc4*

Il **goal** consiste nell'avere in *ws1* oggetti di tipo *bolt*, in *ws2* oggetti di tipo *bolt* e in *ws3* oggetti di tipo *valve* e in *ws4* oggetti di tipo *tool*.

Istanza difficile

Per quanto riguarda l'istanza difficile, è stato scelto di utilizzare:

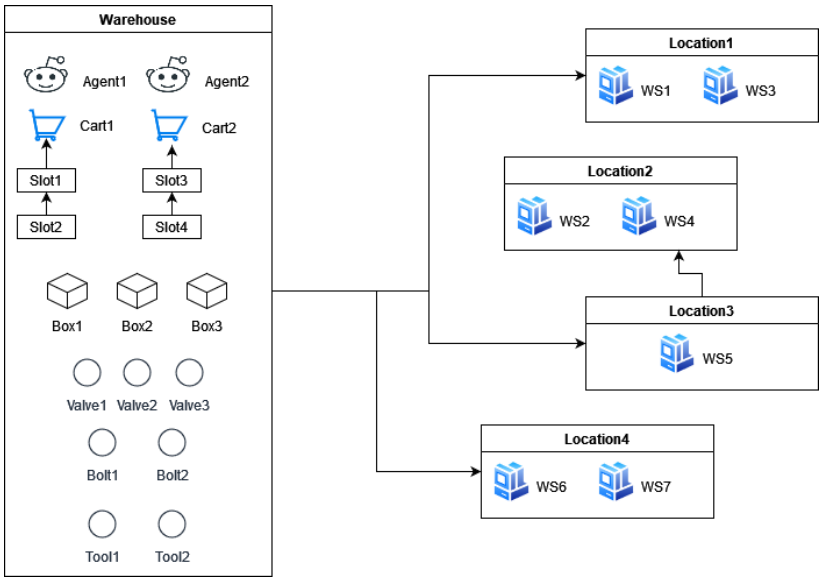


Diagramma relativo al problema "difficile" dell'istanza 2

- 2 *agent*, localizzati nella *warehouse*
- 2 *cart*, *cart1* per *agent1* e *cart2* per *agent2*
- Ogni *cart* contiene 2 *slot*, *slot1* e *slot2* per il *cart1*, *slot3* e *slot4* per il *cart2*
- 3 *box*, localizzate nella *warehouse*
- 3 tipi di contenuto: *bolt, valve, tool*
- 2 oggetti di tipo *bolt*
- 3 oggetti di tipo *valve*
- 2 oggetti di tipo *tool*
- 5 location: *warehouse, loc1, loc2, loc3, loc4*
- 7 *workstation* così localizzate:
    - *ws1* e *ws3* in *loc1*
    - *ws2* e *ws4* in *loc2*
    - *ws5* in *loc3*
    - *ws6* e *ws7* in *loc4*
- Connessioni tra location:
    - *warehouse* connessa con *loc1, loc3* e *loc4*
    - *loc2* connessa con *loc3*

Il **goal** consiste nell'avere in *ws1* oggetti di tipo *bolt*, in *ws2* oggetti di tipo *valve*, in *ws3* oggetti di tipo *valve* e oggetti di tipo *tool* e in *ws6* oggetti di tipo *tool*.

2.3 Instance 2 - numeric fluents

2.3.1 Dominio

Nella definizione del dominio per l'istanza 2.2 è stata definita una versione dello stesso utilizzando i *numeric-fluents*. Quest'ultima non è supportata dal planner custom e di conseguenza per la sua successiva risoluzione è stato utilizzato un altro planner.

Tipi

I tipi definiti nella modellazione del dominio sono rimasti invariati, ad eccezione di "*slot*", che è stato eliminato:

```
(:types
    location agent contentType - object
    workstation box content agent - locatable
    workstation box - container
    content box - containable
  )
```

Predicati

```
(:predicates
    (at ?locatable - locatable ?location - location )
    (contain ?container - container ?containable - containable)
    (connected ?loc1 - location ?loc2 - location)
    (workstation-has-type ?workstation - workstation ?contentType - contentType)
    (is-type ?content - content ?contentType - contentType)
    (is-empty ?container - container)
    ;Instance no. 2 with numeric fluents
    ;No need slots, handling of capacity is done with functions
    (agent-has-carrier ?agent - agent ?carrier - carrier)
    (carrier-has-box ?carrier - carrier ?box - box)
)
```

I predicati eliminati sono:

- **carrier-has-slot**

  Visto l'utilizzo di funzioni e *numeric-fluents*, la capacità del carrier viene modellata tramite questi ultimi; dunque, non si ha più bisogno del concetto di *"slot"*.

Funzioni

A questo punto arrivati, sono state definite le seguenti funzioni:

```
    (:functions

        (curr-carrier-load ?carrier - carrier);current load of the carrier:
number of boxes in it
        (capacity ?carrier - carrier )              ;capacity of the carrier
    )
```

Le funzioni definite si occupano di gestire l'uso efficiente dei *carrier* da parte degli agenti.

- *curr-carrier-load* indica quante scatole sono già caricate in un *carrier*, mentre *capacity* definisce quante *box* il *carrier* può contenere in totale. Utilizzando queste due funzioni, il sistema può decidere se un *carrier* può ricevere ulteriori *box* o se è già pieno.

Più specificatamente:

1. curr-carrier-load:

- **Descrizione**: Questa funzione rappresenta il carico attuale di un *carrier*, misurato come il numero di *box* attualmente contenute al suo interno;
- **Tipo di Ritorno**: Un valore numerico che indica quante *box* sono presenti nel *carrier*;
- **Utilizzo**: Questa funzione viene utilizzata per monitorare e gestire la capacità di carico di un *carrier*, verificando se è possibile aggiungere ulteriori *box* o se il *carrier* ha raggiunto la sua capacità massima;
- **Esempio**: Se un *carrier carrier1* contiene attualmente 3 *box*, la funzione *(curr-carrier-load carrier1)* restituirà il valore 3.

2. capacity:

- **Descrizione**: Questa funzione rappresenta la capacità massima di un *carrier*, ovvero il numero massimo di scatole che il carrier può contenere;
- **Tipo di Ritorno**: Un valore numerico che indica la capacità totale del carrier;
- **Utilizzo**: Viene utilizzata per confrontare il carico attuale del *carrier* con la sua capacità massima, al fine di determinare se è possibile aggiungere altre *box* nel *carrier* senza superare il limite;
- **Esempio**: Se un *carrier carrier1* ha una capacità massima di 5 *box*, la funzione *(capacity carrier1)* restituirà il valore 5.

Azioni

Infine, sono state adattate le azioni alla definizione del dominio con funzioni e *numeric-fluents*. Logicamente, le azioni modificate sono quelle che coinvolgono l'azione di "caricamento" e di "scaricamento" delle box.

Dunque:

- **pick-up-from-location**

```
    (:action pick-up-from-location
        :parameters (?agent - agent ?box - box ?location - location
?carrier - carrier)
        :precondition (and
                            (at ?agent ?location)
                            (at ?box ?location)
                            (agent-has-carrier ?agent ?carrier)
                            (< (curr-carrier-load ?carrier) (capacity
?carrier))
        )
        :effect (and
                    (not (at ?box ?location))
                    (carrier-has-box ?carrier ?box)
                    (increase (curr-carrier-load ?carrier) 1)
        )
    )
```

L'azione consente a un *agent* di prelevare una *box* da una *location* specifica e caricarla su un *carrier*, a condizione che ci sia spazio disponibile nel *carrier*. Una volta eseguita, la *box* viene rimossa dalla posizione iniziale e aggiunta al *carrier*.

Negli effetti si può notare l'aggiornamento del carico.

- **pick-up-from-workstation**

```
 (:action pick-up-from-workstation
        :parameters (?agent - agent ?workstation - workstation
?location - location ?box - box  ?carrier - carrier)
        :precondition (and
                    (at ?agent ?location)
                    (at ?workstation ?location)
                    (contain ?workstation ?box)
                    (agent-has-carrier ?agent ?carrier)
                    (< (curr-carrier-load ?carrier) (capacity
?carrier))
        )
        :effect (and
            (not (contain ?workstation ?box))
            (carrier-has-box ?carrier ?box)
            (increase (curr-carrier-load ?carrier) 1)
        )
    )
```

L'azione *pick-up-from-workstation* permette a un *agent* di prelevare una *box* da una workstation e di caricarla su un *carrier*, a condizione che il *carrier* non abbia già raggiunto la sua capacità massima. Una volta eseguita l'azione, la *box* viene rimossa dalla *workstation* e aggiunta al carico del *carrier*.

Negli effetti si può notare l'aggiornamento del carico.

- **Deliver-to-location**

```
    (:action deliver-to-location
        :parameters (?agent - agent ?location - location ?box - box
?carrier - carrier)
        :precondition (and
                            (at ?agent ?location)
                            (agent-has-carrier ?agent ?carrier)
                            (carrier-has-box ?carrier ?box)
        )
        :effect (and
                    (not (carrier-has-box ?carrier ?box))
                    (at ?box ?location)
                    (decrease (curr-carrier-load ?carrier) 1)
        )
    )
```

L'azione permette a un *agent* di consegnare una *box* in una specifica *location*, scaricandola dal suo carrier.

Negli effetti, si può notare come il carico attuale del *carrier* venga decrementato di 1, riflettendo che una *box* è stata rimossa dal *carrier*.

- **Deliver-to-workstation**

```
    (:action deliver-to-workstation
        :parameters (?agent - agent ?workstation - workstation
?location - location ?box - box ?carrier - carrier)
        :precondition (and
                            (at ?agent ?location)
                            (at ?workstation ?location)
                            (agent-has-carrier ?agent ?carrier)
                            (carrier-has-box ?carrier ?box)
        )
        :effect (and
                    (not (carrier-has-box ?carrier ?box))
                    (contain ?workstation ?box)
                    (decrease (curr-carrier-load ?carrier) 1)
        )
    )

)
```

L'azione rappresenta la consegna di una *box* da parte di un *agent* a una *workstation* specifica. Alla fine dell'azione, la *box* viene trasferita dal *carrier* alla *workstation*, e il carico del *carrier* viene aggiornato per riflettere la rimozione della *box*.

2.3.2Problema

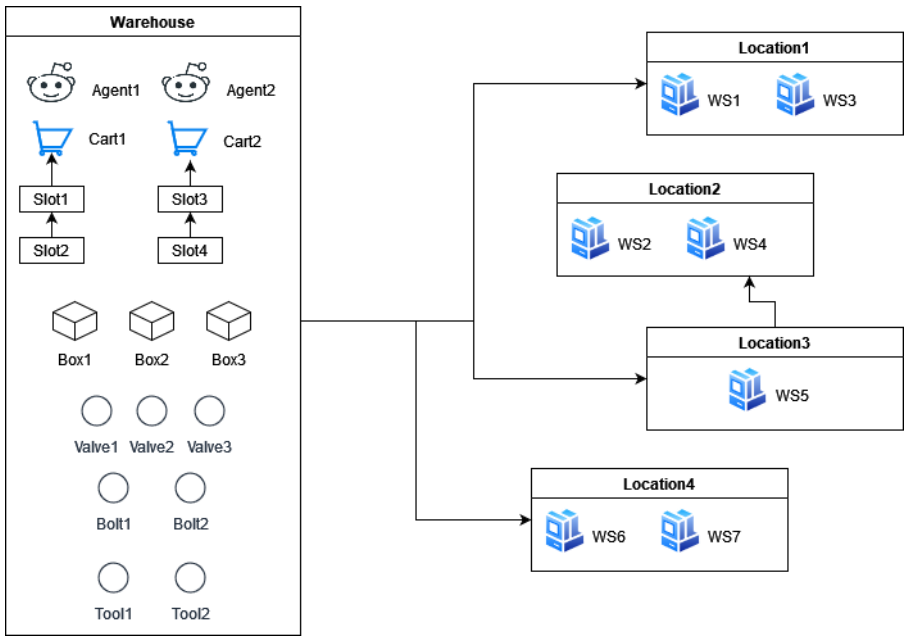


Diagramma relativo al problema per i numeric fluents

L'istanza del problema è uguale all'istanza difficile definita in precedenza ma con le seguenti considerazioni:

- Non è necessario utilizzare oggetti di tipo "*slot*".
- Non è necessario utilizzare il predicato "*carrier-has-slot*".

Per esprimere che i due *carrier* hanno capacità pari a 2 e che inizialmente sono vuoti, nel file del problema vengono inseriti:

```
- (= (capacity cart1) 2)
- (= (capacity cart2) 2)
- (= (curr-carrier-load cart1) 0)
- (= (curr-carrier-load cart2) 0)
```

2.4 Classical Planning

Il Planner sviluppato sfrutta l'algoritmo di ricerca A^*.

Al fine di calcolare i consumi relativi di memoria e tempo sono state utilizzate rispettivamente le librerie fornite da java: *Java Lang Memory Management* e *Java Io.*

```java
@Override
public Plan solve(final Problem problem) {

    LOGGER.info("* Starting A* search \n");
    // Search a solution
    Plan plan = null;
    //Per poter tenere traccia della memoria usata dall'algoritmo
    MemoryMXBean memoryBean = ManagementFactory.getMemoryMXBean();
    MemoryUsage beforeHeapMemoryUsage = memoryBean.getHeapMemoryUsage();
    Long beforeUsedMemory = beforeHeapMemoryUsage.getUsed();

    //Per tenere traccia del tempo usato dall'algoritmo
    final long begin = System.currentTimeMillis();

    try {
        plan = this.astar(problem);
    } catch (ProblemNotSupportedException e) {
        LOGGER.error("Problem not supported: " + e.getMessage());
    }

    final long end = System.currentTimeMillis();
    MemoryUsage afterHeapMemoryUsage = memoryBean.getHeapMemoryUsage();
    long afterUsedMemory = afterHeapMemoryUsage.getUsed();

    long memoryUsedByCustomAstar = afterUsedMemory - beforeUsedMemory;
    // If a plan is found update the statistics of the planner and log search information
    if (plan != null) {
        LOGGER.info("* A* search succeeded\n");
        this.getStatistics().setTimeToSearch(end - begin);
        this.getStatistics().setMemoryUsedToSearch(memoryUsedByCustomAstar);
    } else {
        LOGGER.info("* A* search failed\n");
    }
    // Return the plan found or null if the search fails.
    return plan;
}
```

Solver

Per garantire un'elevata efficienza sono state adottate due tecniche: *Pruning* ed *Eliminazione di stati ridondanti*.

In supporto delle tecniche sopracitate è stata creata una classe aggiuntiva: *CheckerBoxContent*.

2.4.1 CheckerBoxContent

La classe CheckerBoxContent è utilizzata per mantenere traccia di diversi aspetti fondamentali:

1. Contenuti delle box.
2. Locazione Delle box.
3. Locazione dei contenuti.
4. Contenuti e Box processati (utile al pruning)

Per poter gestire queste informazioni sono state utilizzate delle mappe, la classe appare così strutturata:

```
public class CheckerBoxContent {
    public HashMap<String, List<String>> boxLocation = new HashMap<>();   // Mappa delle locazioni alle liste di scatole
    public HashMap<String, List<String>> contentLocation = new HashMap<>(); // Mappa delle locazioni alle liste di contenuti
    public HashMap<String,String> boxContent = new HashMap<>(); //Mappa delle posizioni dei contenuti nei box
    public Set<String> processedContent = new HashSet<>(); //Set dei contenuti processati
    public Set<String> processedBox = new HashSet<>(); //Set delle box processate

    public HashMap<String, String> contentType;  // Mappa che associa ogni contenuto al suo tipo
    public Set<String> locations = new HashSet<>();  // Lista di tutte le locazioni conosciuteì
```

Le strutture dati di box e contenuti processati è un set per avere sempre ordinati i contenuti e non avere strutture diverse dettata da liste disordinate.
Le due liste *processedContent* e *processedBox* sono utilizzate per evitare l'analisi di Stati Ridondanti durante l'esplorazione dell'albero.
Dato che un contenuto può essere inserito in diverse box e la box può essere caricata su diversi slot; si sviluppa un numero enorme di stati che risultano essere però uguali tra loro poiché è importante l'inserimento del contenuto in una box, ma non è rilevante quale sia la box.

Oltre al tipico costruttore è presente un costruttore che genera una copia profonda dell'intera struttura.

Le tre funzioni principali sono: *Insert*, *UpdateContentLocation*, *UpdateBoxLocation*.

```java
// Funzione per aggiungere un oggetto a una locazione specifica
public void insert(String object, String location, String type) {
    switch (type) {
        case "BOX":
            addObjectToLocation(boxLocation, object, location);
            break;
        case "CONTENT":
            addObjectToLocation(contentLocation, object, location);
            break;
        default:
            System.err.println("Tipo non riconosciuto: " + type);
            break;
    }
    addLocationIfAbsent(location);
    updateStateHash(); // Ricomputare l'hash dopo ogni inserimento
}
```

Insert

```java
public void updateBoxLocation(String box, String location, boolean deploy){
    List<String> boxes = boxLocation.get(location);

    if(deploy){ //Caso in cui l'agente lascia la scatola
        if(boxes==null){
            List<String> boxesList = new LinkedList<>();
            boxLocation.put(location,boxesList);
        }
        boxLocation.get(location).add(box);
    }
    else{ //Caso in cui l'agente fa la pickup della scatola
        boxLocation.get(location).remove(box);
        processedBox.add(box);
    }

}
```

UpdateBoxLocation

```java
public void updateContentLocation(String box, String location, String content, boolean empty, boolean isWorkstation){
    List<String> contents = contentLocation.get(location);
    //System.out.println(box+" . "+location+" . "+content);

    if(empty){ //Caso della empty del contenuto nella workstation
        processedBox.remove(box);
        if (boxContent.containsKey(box)) {
            boxContent.remove(box);
        }
        //Nel caso della delivery nella workstation il contenuto, dato il pruning,
        //soddisferà sicuramente il goal quindi non lo aggiungiamo alla location
        //In caso di empty in una location allora deve essere inserito
        if(!isWorkstation){
            if(contentLocation.get(location)==null){
                List<String> contentsList = new LinkedList<>();
                contentLocation.put(location,contentsList);
            }
            contentLocation.get(location).add(content);
        }
    }
    else{ //Caso di fill
        if(contentLocation.containsKey(location)){
            contentLocation.get(location).remove(content);
            processedContent.add(content);
            //System.out.println("processedContent: "+processedContent);
            boxContent.put(box,content);
        }
    }
}
```

UpdateContentLocation

Il confronto degli stati ridondanti è sviluppato tramite gli Hash delle due liste processate, dunque, sono presenti anche i seguenti due metodi:

```
public int getProcessedContentHash() {
    return processedContent.hashCode();
}

public int getProcessedBoxHash(){
    return processedBox.hashCode();
}
```

2.4.2 Algoritmo A^*

A supporto dell'algoritmo di ricerca si utilizzano le seguenti strutture dati:

```
HashMap<String,String> object_type = new HashMap<>(); //<Object, Type>
HashMap<String, String> Types = new HashMap<>(); //<Content, ContentType>
HashMap<String, List<String>> TypesLoc = new HashMap<>(); //<Type, Location>
HashMap<String, Boolean> goalType = new HashMap<>(); //<Type, IsNeededForGoal>
Set<String> goalWS = new HashSet<>(); //<Workstations needed for the goal>
Set<Integer> rendundantContentState = new HashSet<>(); //<States with redundant content>
Set<Integer> rendundantBoxState = new HashSet<>(); //<States with redundant boxes>
HashMap<String, String> locationWS = new HashMap<>(); //<Location, Workstation>
HashMap<Integer, String> Predicates = new HashMap<>(); //<PredicateValue, Predicate>
HashMap<Integer, String> Constants = new HashMap<>();  //<ConstantValue, Constant>
HashMap<String,LinkedList<String>> goal = new HashMap<>(); //<Workstation, ContentsNeededForTheGoal>
Map<Node, CheckerBoxContent> nodeCheckerBoxMap = new HashMap<>(); //Saving CheckerBox content
```

Mappe utilizzate per l'algoritmo A^*

Ognuna di queste è necessaria per il raggiungimento del goal tramite il percorso più veloce:

- Object_type: è una mappa che ha come chiave le costanti presenti nel problema e come valore associato il corrispondente tipo.
- Types: Contiene la mappatura tra *Content* e *ContentType.* (Es. <Content1,valve>)
- GoalType: contiene le informazioni relative ai tipi necessari al goal.
- GoalWs: contiene le informazioni relative alle workstation che hanno richieste nel goal.
- rendundantContentState e rendundantBoxState: Contengono il valore hash delle due strutture precedentemente analizzate.
- LocationWS: Detiene le informazioni su quali location sono presenti le workstation.
- Predicates: Mappatura dei predicati.
- Constants: Mappatura delle costanti.
- nodeCheckerBoxMap: Mappatura della corrispondente checkerBox al corrispondente nodo.

Le mappe appena analizzate si riempiono durante il primo passo della ricerca di A^*.
L'uso di mappe è necessario in quanti PDDL4J sfrutta una mappatura interna alla quale associa i Predicati e le Costanti a dei valori, per poterli utilizzare all'interno della ricerca era necessario mapparli.

```
//Constants Mapping
int index = 0;
for (String s : problem.getConstantSymbols()) {
    Constants.put(index, s);
    index++;
}

//Predicates Mapping
index = 0;
for (String s : problem.getPredicateSymbols()) {
    Predicates.put(index, s);
    index++;
}
```

Mapping di Predicati e Costanti

Una volta ottenuti i valori necessari alla decodifica si riempiono le restanti mappe e la checkerBoxContent del nodo radice:

```
// Problem Fluents full List
List<Fluent> fluents = problem.getFluents();

//Bit Vector of positive Fluents
BitVector initialStateFluents = problem.getInitialState().getPositiveFluents();

//Bit Vector of Goal Fluents
BitVector goalFluents = problem.getGoal().getPositiveFluents();

int initialStateBitVectorIndex = initialStateFluents.nextSetBit(0);
int goalBitVectorIndex = goalFluents.nextSetBit(0);
int indexFluent = 0;

//Creation of checkerBox, ready to fill up
CheckerBoxContent checkerBoxContent = new CheckerBoxContent();

//Temporary Map to check if the problem is solvable
Map<String,Integer> numberOfType = new HashMap<>();
```

Inizializzazione dei Fluents

```
for (Fluent f : fluents) {
    boolean isInInitialState = (initialStateBitVectorIndex == indexFluent);
    boolean isInGoalState = (goalBitVectorIndex == indexFluent);
    int simboloAzione = f.getSymbol();

    // Gestisci il fluente se è presente nello stato iniziale
    if (isInInitialState) {
        if (Predicates.get(simboloAzione).equals("at")) {
            int objectValue = f.getArguments()[0];
            int locationValue = f.getArguments()[1];

            String object = Constants.get(objectValue);
            String location = Constants.get(locationValue);

            if (object_type.get(object).equals("BOX")) {
                checkerBoxContent.insert(object, location, "BOX");
            } else if (object_type.get(object).equals("CONTENT")) {
                checkerBoxContent.insert(object, location, "CONTENT");
            }
            else if (object_type.get(object).equals("WORKSTATION"))
                locationWS.put(object,location);
        }
        if (Predicates.get(simboloAzione).equals("is-type")) {
            int objectValue = f.getArguments()[0];
            int typeValue = f.getArguments()[1];

            String object = Constants.get(objectValue);
            String type = Constants.get(typeValue);

            if(numberOfType.containsKey(type)){
                int pre = numberOfType.get(type);
                pre++;
                numberOfType.put(type,pre);
            }
            else numberOfType.put(type,1);

            Types.put(object, type);
        }
        initialStateBitVectorIndex = initialStateFluents.nextSetBit(initialStateBitVectorIndex + 1);
    }
```

Gestione Fluents Positivi del Problem

Si effettuano le iterazioni su tutti i Fluents presentanti nel problema, e si verificano quali sono veri e quali non lo sono.

In particolare, tramite la lista temporanea *numbersOfType*, si verifica se i contenuti presenti nel problema sono sufficienti a soddisfare il goal; in caso negativo la ricerca si arresta restituendo un messaggio d'errore.

```
// Managament of the fluents if its in the goal
if (isInGoalState) {
    if (Predicates.get(simboloAzione).equals("workstation-has-type")) {
        String contentType = Constants.get(f.getArguments()[1]);
        String ws = Constants.get(f.getArguments()[0]);
        if(!goal.containsKey(ws)){
            goal.put(ws,new LinkedList<>());
        }

        if(numberOfType.containsKey(contentType)) {
            int pre = numberOfType.get(contentType);
            pre--;
            numberOfType.put(contentType,pre);
        }
        else
            numberOfType.put(contentType,-1);
        goal.get(ws).add(contentType);
        goalType.put(contentType, true);
        goalWS.add(ws);
    }
    goalBitVectorIndex = goalFluents.nextSetBit(goalBitVectorIndex + 1);
}
```

Gestione Fluents Positivi del Goal

Tramite la seconda condizione di *if*, si gestiscono i goal, in particolar modo si analizzano quali contenuti e workstation sono interessate per il goal.

```
//Creation of the first node (root) and its CheckeBoxMap which represent the initial state of box and content;
final Node root = new Node(init, null, -1, 0, heuristic.estimate(init, problem.getGoal()));
nodeCheckerBoxMap.put(root, checkerBoxContent);
open.add(root);
```

Nei passaggi precedenti è stata sviluppata la CheckerBoxMap iniziale; in questa sono salvate tutte le locazioni di box e contenuti nel ***primo stato iniziale*** dettato dall'init del problema.

A questo punto è possibile l'inizio della ricerca:

```
// Start the search
while (!open.isEmpty() && plan == null && time < timeout) {
    final Node current = open.poll();
    close.add(current);

    CheckerBoxContent currentCheckerBox = nodeCheckerBoxMap.get(current);

    if (current.satisfy(problem.getGoal())) {
        return this.extractPlan(current, problem);
    } else {
        for (int i = 0; i < problem.getActions().size(); i++) {
            Action a = problem.getActions().get(i);
```

Ad ogni iterazione del ciclo si toglie dalla coda il Nodo corrente che deve essere analizzato.

Dato che, ad ogni iterazione si generano dei nodi dettati dalle azioni possibili, è sicuramente presente, all'interno della mappa, il corrispondente stato degli elementi del nodo preso in considerazione.

```
if (a.isApplicable(current)) {
    boolean modifiedState = false;
    CheckerBoxContent newCheckerBoxContent;
    if(a.getName().contains("pick-up") ||  a.getName().contains("deliver") || a.getName().contains("fill") || a.getName().contains("empty")){
        newCheckerBoxContent = new CheckerBoxContent(currentCheckerBox);
        modifiedState = true;
    }
    else {
        newCheckerBoxContent = currentCheckerBox;
    }
```

Creazione opportuna delle CheckerBox

Al fine di evitare una generazione esorbitante di stati che porterebbe il sistema a sforare l'heap, si generano nuove checkerBoxContent solo e soltanto quando si prendono in considerazione delle mosse che portano ad una modifica dello stato attuale di box o contenuti; in caso contrario si utilizza lo stato precedente.

```
//Check if the action selected is promising or if it will generate a sub-optimal state
if(PromisingMove(a,newCheckerBoxContent)){
    Node next = new Node(current);

    final List<ConditionalEffect> effects = a.getConditionalEffects();
    for (ConditionalEffect ce : effects) {
        if (current.satisfy(ce.getCondition())) {
            next.apply(ce.getEffect());
        }
    }
```

Prima Fase del Pruning

Per la gestione del Pruning si utilizza il metodo apposito *Promising Move*, al quale si passano come argomenti l'action ammissibile e lo stato attuale di box e contenuti.

Il Pruning rimuove tutte le mosse che sono valide ma che porterebbero ad uno stato sub-ottimale del plan; nel dettaglio:

```
//If a box is filled with a content which isn't requested in the goal -> remove the move
if(a.getName().contains("fill-box-from-location")){
    String object = Constants.get(a.getValueOfParameter(2));
    String type = Types.get(object);
    return goalType.containsKey(type);
}

else if(a.getName().contains("fill-box-from-workstation")){
    String type = Constants.get(a.getValueOfParameter(4));
    return goalType.containsKey(type);
}
```

Operazione Fill Box

Nel caso in cui si effettui l'operazione di riempimento (fill) di una box, ma il contenuto preso in considerazione NON fa parte di quelli del goal allora la mossa non è ottimale e si rimuove l'albero corrispondente generato.

```
//emptying a box in a workstation that DOESN'T need that content type is a wrong move
else if(a.getName().contains("empty-box-workstation")){
    String workstation = Constants.get(a.getValueOfParameter(4));
    String leavingContent = Constants.get(a.getValueOfParameter(2));
    //If its leaving a goal content in the wrong workstation the move is not logical
    boolean isAGoalContent = goalType.get(Types.get(leavingContent));
    if(goal.get(workstation).contains(Types.get(leavingContent))) return true;
    return false;
}
```

Operazione Empty-box-Workstation

Per quanto riguarda le operazioni di empty esistono due casi. Nel caso in cui l'agent svuota la box in una workstation allora necessariamente deve aver inserito il content del tipo giusto nella workstation che lo richiede, altrimenti la mossa porterebbe sicuramente ad uno stato non ottimale.

```
//emptying a box which has a contentype needed in goal is a wrong move move
//emptying a box in a location that doesn't have a goal content is a wrong move
else if(a.getName().contains("empty-box-location") ){
    String location = Constants.get(a.getValueOfParameter(3));
    String leavingContent = Constants.get(a.getValueOfParameter(2));
    boolean isAGoalContent = goalType.containsKey(Types.get(leavingContent)); //If its removing a goal content the move is not logical
    if(!isAGoalContent){
        //If the content removed from the box is not a goal content,
        //there must be at least one other content at the location that has a goal content type.
        if(checkerBoxContent.contentLocation.containsKey(location)){
            List<String> contentInLocation = checkerBoxContent.contentLocation.get(location);
            for(String content : contentInLocation){ //For all elements in that location
                String typeOfContent = Types.get(content);
                if(goalType.containsKey(typeOfContent)){
                    return true;
                }
            }
        }
    }
    return false;
}
```

Operazione Empty-box-Location

Il secondo caso risulta essere la *empty* in una location. In questo caso si hanno due controlli da effettuare:

1. Se si effettua la empty di un contenuto che è utile al goal allora la mossa è da scartare in quanto porterebbe ad un maggiore numero di mosse.
2. Se si effettua la empty in una location di un contenuto NON utile al goal, allora in quella stessa location deve essere presente un contenuto UTILE al goal, altrimenti la mossa non porterebbe ad un avanzamento della soluzione.

```
//if there is a goal content in that location then the move makes sense
else if(a.getName().contains("deliver-to-location")){
    String location = Constants.get(a.getValueOfParameter(1));
    String box = Constants.get(a.getValueOfParameter(2));
    String contentInBox = checkerBoxContent.boxContent.get(box);

    //if the box has a content and its a goal content then the deliver to location move is wrong
    if(contentInBox!=null){
        boolean isAGoalContent = goalType.containsKey(Types.get(contentInBox));
        if(isAGoalContent) return false;
    }
    if(checkerBoxContent.contentLocation.containsKey(location)){
            List<String> contentInLocation = checkerBoxContent.contentLocation.get(location);
            for(String content : contentInLocation){ //For all elements in that location
                String typeOfContent = Types.get(content);
                if(goalType.containsKey(typeOfContent)){
                    return true;
                }
            }
        }
    return false;
}
```

Operazione Deliver to Location

Nel caso dell'azione di *deliver to location* si effettua un controllo sui contenuti presenti nella location considerata nell'azione; se un contenuto del tipo richiesto nel goal è presente nella location allora ha senso svuotare la box, ma se la box contiene già un elemento utile al raggiungimento del goal non ha senso rimuoverlo per un altro.

```
//if the box is empty or the content is not the one requested than the action is wrong
else if(a.getName().contains("deliver-to-workstation")){
    String workstation = Constants.get(a.getValueOfParameter(1));
    String box = Constants.get(a.getValueOfParameter(3));
    String contentInsideBox = checkerBoxContent.boxContent.get(box);
    if(contentInsideBox==null) return false;

    if(goalWS.contains(workstation))
     if(goal.get(workstation).contains(Types.get(contentInsideBox)))
        return true;
    return false;
}
```

Operazione Deliver to Workstation

L'azione *deliver to workstation* è sensata se e soltanto se si rilascia ad una workstation presente nel goal una box che ha il contenuto che serve; altrimenti è da scartare.

```
//If the result is true, that means there is a box with a goal content in another place
else if(a.getName().contains("pick-up-from-workstation")){
    return !checkerBoxContent.checkPresence(goalType.keySet());
}

return true;
```

Operazione Pick-up-from-workstation

Tramite il seguente metodo si verifica se esiste un contenuto utile al goal nella stessa location, nel caso in cui questa condizione sia vera è insensato caricare sul robot la box vuota in quanto si effettuerebbero azioni extra.

```
else if(a.getName().contains("pick-up-from-location")){
    String box = Constants.get(a.getValueOfParameter(1));
    String loc = Constants.get(a.getValueOfParameter(2));
    String contentInsideBox = checkerBoxContent.boxContent.get(box);
    if(contentInsideBox!=null){ //if the box is full and not contain a goal content than the move is wrong
        boolean isAGoalContent = goalType.containsKey(Types.get(contentInsideBox));
        if(!isAGoalContent)
            return false;
    }
    else{ //if the box is empty and there is a goal content in that loction the pickup is wrong
        if(checkerBoxContent.contentLocation.containsKey(loc)){
            List<String> contentInLocation = checkerBoxContent.contentLocation.get(loc);
            for(String content : contentInLocation)
                if(goalType.containsKey(Types.get(content)))
                    return false;
        }
    }
}
```

Operazione Pick-up-from location

L'ultima azione considerata è la *pickup-from-location*, in questo caso si presentano due possibili eventi:

1. La box è piena: in questo caso si verifica se il contenuto è utile ai fini del goal, in caso negativo la mossa viene rimossa dall'albero.
2. La box è vuota: in questo caso, se un contenuto utile al goal è presente nella location, allora l'azione non ha senso e viene rimossa.

L'ultimo aspetto è relativo all'eliminazione degli stati ridondanti, per evitarli si usufruisce delle mappe analizzate precedentemente all'inizio del paragrafo: *rendundantContentState* e *rendundantBoxState.*

Uno stato si definisce ridondante quando non porta effettivi progressi al raggiungimento del goal.
Ad esempio, una stessa box può essere caricata da più agenti e/o su diversi slot, ma i vari stati che si creano sono ridondanti in quanto per il raggiungimento di un planner non interessa quale box si utilizza ma che venga usata almeno una box.
Dunque, moltissimi stati che si creano sono inutili al fine del raggiungimento del plan finale poiché una mossa è equivalente all'altra.

In prima fase si verifica se l'azione modifica lo stato attuale di box e content:

```
if (a.getName().equals("pick-up-from-workstation")) {
    String obj = Constants.get(a.getValueOfParameter(3));
    String loc = Constants.get(a.getValueOfParameter(2));

    //Update the content and box state
    newCheckerBoxContent.updateBoxLocation(obj, loc, false);

    int hash = newCheckerBoxContent.getProcessedBoxHash();
    if(rendundantBoxState.contains(hash)) continue;
    else rendundantBoxState.add(hash);
} else if (a.getName().equals("pick-up-from-location")) {
    String obj = Constants.get(a.getValueOfParameter(1));
    String loc = Constants.get(a.getValueOfParameter(2));

    //Update the content and box state
    newCheckerBoxContent.updateBoxLocation(obj, loc, false);

    //Check if the state is rendundant
    int hash = newCheckerBoxContent.getProcessedBoxHash();
    if(rendundantBoxState.contains(hash)) continue;
    else rendundantBoxState.add(hash);
}
```

Aggiornamento con operazione di pick-up

Nel caso dell'azione di *pickup* si modifica lo stato delle box, dunque si verifica se quella presa in considerazione non era già stata presa in considerazione precedentemente, in caso positivo si esclude l'azione, altrimenti si procede.

```
else if (a.getName().equals("deliver-to-workstation")) {
    String obj = Constants.get(a.getValueOfParameter(3));
    String loc = Constants.get(a.getValueOfParameter(2));

    //Check if the state is rendundant
    int hash = newCheckerBoxContent.getProcessedBoxHash();
    if(rendundantBoxState.contains(hash)) rendundantBoxState.remove(hash);

    //Update the content and box state
    newCheckerBoxContent.updateBoxLocation(obj, loc, true);
}
else if (a.getName().equals("deliver-to-location")) {
    String obj = Constants.get(a.getValueOfParameter(2));
    String loc = Constants.get(a.getValueOfParameter(1));

    //Check if the state is rendundant
    int hash = newCheckerBoxContent.getProcessedBoxHash();
    if(rendundantBoxState.contains(hash)) rendundantBoxState.remove(hash);

    //Update the content and box state
    newCheckerBoxContent.updateBoxLocation(obj, loc, true);
}
```

Aggiornamento con operazione di Deliver

Nel caso in cui una box viene rilasciata allora può essere nuovamente presa in considerazione al fine di raggiungere il goal, di conseguenza, la si rimuove dalla lista.

```
else if (a.getName().equals("empty-box-workstation")) {
    String box = Constants.get(a.getValueOfParameter(1));
    String obj = Constants.get(a.getValueOfParameter(2));
    String loc = Constants.get(a.getValueOfParameter(5));

    //Update the content and box state
    newCheckerBoxContent.updateContentLocation(box, loc, obj, true, true);
}
else if (a.getName().equals("empty-box-location")) {
    String box = Constants.get(a.getValueOfParameter(1));
    String obj = Constants.get(a.getValueOfParameter(2));
    String loc = Constants.get(a.getValueOfParameter(3));

    //Update the content and box state
    newCheckerBoxContent.updateContentLocation(box, loc, obj, true, false);
}
```

Aggiornamento operazione di empty

```
else if (a.getName().equals("fill-box-from-location")) {
    String box = Constants.get(a.getValueOfParameter(1));
    String obj = Constants.get(a.getValueOfParameter(2));
    String loc = Constants.get(a.getValueOfParameter(3));

    //Update the content and box state
    newCheckerBoxContent.updateContentLocation(box, loc,obj, false, false);

    //Check if the state is rendundant
    int hash = newCheckerBoxContent.getProcessedContentHash();
    if(rendundantContentState.contains(hash)) continue;
    else rendundantContentState.add(hash);
}
else if (a.getName().equals("fill-box-from-workstation")) {
    String box = Constants.get(a.getValueOfParameter(1));
    String obj = Constants.get(a.getValueOfParameter(2));
    String loc = Constants.get(a.getValueOfParameter(5));

    //Update the content and box state
    newCheckerBoxContent.updateContentLocation(box, loc, obj, false, true);

    //Check if the state is rendundant
    int hash = newCheckerBoxContent.getProcessedContentHash();
    if(rendundantContentState.contains(hash)) continue;
    else rendundantContentState.add(hash);
}
```

Aggiornamento operazione di fill-box

Nel caso dei contenuti, a differenza delle box che possono essere riutilizzate, una volta che vengono processati non devono essere nuovamente riconsiderati proprio per questo motivo non si rimuovono dalla lista.

NOTA: Nel caso dell'istanza 1 non è necessario l'utilizzo dell'hashing delle box.

2.4.3 Euristica

```java
public double customEstimate(State state, Condition goal, Action a, double currentHeuristic) {
    super.setGoal(goal);
    super.expandRelaxedPlanningGraph(state);

    BitVector positiveFluents = goal.getPositiveFluents(); //Get the positive Fluents
    int satisfiedFluents = 0;

    //Count the positive fluents that are satsified in the actual state
    for (int i = positiveFluents.nextSetBit(0); i >= 0; i = positiveFluents.nextSetBit(i + 1)) {
        BitVector fluentCheck = new BitVector();
        fluentCheck.set(i);

        if (state.satisfy(new Condition(fluentCheck, new BitVector()))) {
            satisfiedFluents++;
        }
    }

    BitVector negativeFluents = goal.getNegativeFluents(); //Get the negative Fluents

    //Count the positive fluents that are satsified in the actual state
    for (int i = negativeFluents.nextSetBit(0); i >= 0; i = negativeFluents.nextSetBit(i + 1)) {
        BitVector fluentCheck = new BitVector();
        fluentCheck.set(i);

        if (!state.satisfy(new Condition(fluentCheck, new BitVector()))) {
            satisfiedFluents++;
        }
    }

    //Calculate the difference between the total flows of the goal and those satisfied by the current state
    int missingFluents = (positiveFluents.cardinality() + negativeFluents.cardinality()) - satisfiedFluents;

    //If there is a move action put a penalty
    double movePenalty = a.getName().contains("move") ? 1.5 : 0.0;

    return missingFluents + movePenalty + (0.1 * currentHeuristic);
}
```

Metodo customEstimate

L’euristica utilizzata valuta quanto uno stato è vicino a soddisfare un obiettivo basato su due tipi di fluents, positivi e negativi.

Il metodo conta quanti fluents dell'obiettivo sono soddisfatti nello stato corrente e quanti ne mancano ancora.

Dato che nel pruning non è stato possibile effettuare un taglio alle operazioni di move poiché unica azione che può avvenire in qualsiasi condizione del sistema.

Per ogni spostamento dell’agent viene applicata una penalità; tramite questa scelta si è ovviato al problema di analisi di percorsi nell’albero che utilizzano un numero elevato di move, in quanto maggiore è il numero di move e maggiore è il valore che quel percorso avrà. Di conseguenza l’euristica si allontanerà più velocemente da plan che hanno un numero elevato di movimenti che poterebbero a risultati non ottimali.

2.4.4 Presentazione Risultati Ottenuti

Istanza 1

Si fa riferimento al problema visto precedentemente al punto [2.1.2]

Risultato *A**	**Risultato FF**

* A* search succeeded
 	
found plan as follows:
 	
00: (fill-box-from-location agent1 box1 tool2 warehouse) [0]
01: (fill-box-from-location agent1 box2 bolt3 warehouse) [0]
02: (fill-box-from-location agent1 box3 valve2 warehouse) [0]
03: (pick-up-from-location agent1 box1 warehouse) [0]
04: (move agent1 warehouse loc1) [0]
05: (deliver-to-workstation agent1 ws3 loc1 box1) [0]
06: (empty-box-workstation agent1 box1 tool2 tool ws3 loc1) [0]
07: (move agent1 loc1 warehouse) [0]
08: (pick-up-from-location agent1 box2 warehouse) [0]
09: (move agent1 warehouse loc1) [0]
10: (deliver-to-workstation agent1 ws1 loc1 box2) [0]
11: (empty-box-workstation agent1 box2 bolt3 bolt ws1 loc1) [0]
12: (move agent1 loc1 warehouse) [0]
13: (pick-up-from-location agent1 box3 warehouse) [0]
14: (move agent1 warehouse loc1) [0]
15: (deliver-to-workstation agent1 ws3 loc1 box3) [0]
16: (empty-box-workstation agent1 box3 valve2 valve ws3 loc1) [0]
17: (pick-up-from-workstation agent1 ws3 loc1 box1) [0]
18: (move agent1 loc1 warehouse) [0]
19: (deliver-to-location agent1 warehouse box1) [0]
20: (fill-box-from-location agent1 box1 valve3 warehouse) [0]
21: (pick-up-from-location agent1 box1 warehouse) [0]
22: (move agent1 warehouse loc3) [0]
23: (move agent1 loc3 loc2) [0]
24: (deliver-to-workstation agent1 ws2 loc2 box1) [0]
25: (empty-box-workstation agent1 box1 valve3 valve ws2 loc2) [0]
 	
time spent: 0.04 seconds parsing
  0.31 seconds encoding
  3.52 seconds searching
  3.87 seconds total time
 	
memory used: 4.82 MBytes for problem representation
  197.53 MBytes for searching
  202.35 MBytes total
	* Starting ENFORCED_HILL_CLIMBING search with FAST_FORWARD heuris
	* ENFORCED_HILL_CLIMBING search succeeded
	
	found plan as follows:
	
	00: (fill-box-from-location agent1 box1 valve1 warehouse) [0]
	01: (pick-up-from-location agent1 box1 warehouse) [0]
	02: (move agent1 warehouse loc3) [0]
	03: (move agent1 loc3 loc2) [0]
	04: (deliver-to-workstation agent1 ws2 loc2 box1) [0]
	05: (empty-box-workstation agent1 box1 valve1 valve ws2 loc2) [0]
	06: (pick-up-from-workstation agent1 ws2 loc2 box1) [0]
	07: (move agent1 loc2 loc3) [0]
	08: (move agent1 loc3 warehouse) [0]
	09: (deliver-to-location agent1 warehouse box1) [0]
	10: (fill-box-from-location agent1 box1 bolt1 warehouse) [0]
	11: (pick-up-from-location agent1 box1 warehouse) [0]
	12: (move agent1 warehouse loc1) [0]
	13: (deliver-to-workstation agent1 ws1 loc1 box1) [0]
	14: (empty-box-workstation agent1 box1 bolt1 bolt ws1 loc1) [0]
	15: (pick-up-from-workstation agent1 ws1 loc1 box1) [0]
	16: (move agent1 loc1 warehouse) [0]
	17: (deliver-to-location agent1 warehouse box1) [0]
	18: (fill-box-from-location agent1 box1 valve2 warehouse) [0]
	19: (pick-up-from-location agent1 box1 warehouse) [0]
	20: (move agent1 warehouse loc1) [0]
	21: (deliver-to-workstation agent1 ws3 loc1 box1) [0]
	22: (empty-box-workstation agent1 box1 valve2 valve ws3 loc1) [0]
	23: (move agent1 loc1 warehouse) [0]
	24: (fill-box-from-location agent1 box2 tool1 warehouse) [0]
	25: (pick-up-from-location agent1 box2 warehouse) [0]
	26: (move agent1 warehouse loc1) [0]
	27: (deliver-to-workstation agent1 ws3 loc1 box2) [0]
	28: (empty-box-workstation agent1 box2 tool1 tool ws3 loc1) [0]
	
	time spent: 0.04 seconds parsing
	  0.32 seconds encoding
	  2.02 seconds searching
	  2.38 seconds total time
	
	memory used: 4.82 MBytes for problem representation
	  0.00 MBytes for searching
	  4.82 MBytes total

Istanza 2 - Facile

Per mostrare le differenze di efficienza che si ottengono utilizzando più robot con i cart sono state utilizzati due differenti problemi analizzati precedentemente al punto [2.2.2]

Risultato *A**	**Risultato *FF***
found plan as follows: 00: (fill-box-from-location agent1 box1 bolt2 warehouse) [0] 01: (fill-box-from-location agent1 box2 bolt1 warehouse) [0] 02: (fill-box-from-location agent1 box3 tool1 warehouse) [0] 03: (pick-up-from-location agent1 box3 warehouse cart1 slot1) [0] 04: (pick-up-from-location agent1 box1 warehouse cart1 slot2) [0] 05: (pick-up-from-location agent2 box2 warehouse cart2 slot3) [0] 06: (move agent1 warehouse loc2) [0] 07: (deliver-to-workstation agent1 ws2 loc2 box1 cart1 slot2) [0] 08: (empty-box-workstation agent1 box1 bolt2 bolt ws2 loc2) [0] 09: (pick-up-from-workstation agent1 ws2 loc2 box1 cart1 slot2) [0] 10: (move agent1 loc2 warehouse) [0] 11: (deliver-to-location agent1 warehouse box1 cart1 slot2) [0] 12: (fill-box-from-location agent1 box1 valve2 warehouse) [0] 13: (move agent1 warehouse loc2) [0] 14: (deliver-to-workstation agent1 ws4 loc2 box3 cart1 slot1) [0] 15: (pick-up-from-location agent2 box1 warehouse cart2 slot4) [0] 16: (empty-box-workstation agent1 box3 tool1 tool ws4 loc2) [0] 17: (move agent2 warehouse loc1) [0] 18: (deliver-to-workstation agent2 ws1 loc1 box2 cart2 slot3) [0] 19: (empty-box-workstation agent2 box2 bolt1 bolt ws1 loc1) [0] 20: (deliver-to-workstation agent2 ws3 loc1 box1 cart2 slot4) [0] 21: (empty-box-workstation agent2 box1 valve2 valve ws3 loc1) [0] time spent: 0.04 seconds parsing 0.21 seconds encoding 3.32 seconds searching 3.57 seconds total time memory used: 2.40 MBytes for problem representation 33.08 MBytes for searching 35.48 MBytes total	* Starting ENFORCED_HILL_CLIMBING search with FAST_FORWARD heuristic * ENFORCED_HILL_CLIMBING search succeeded found plan as follows: 00: (fill-box-from-location agent1 box1 valve1 warehouse) [0] 01: (pick-up-from-location agent1 box1 warehouse cart1 slot1) [0] 02: (fill-box-from-location agent1 box2 bolt1 warehouse) [0] 03: (fill-box-from-location agent1 box3 tool1 warehouse) [0] 04: (pick-up-from-location agent1 box2 warehouse cart1 slot2) [0] 05: (pick-up-from-location agent2 box3 warehouse cart2 slot3) [0] 06: (move agent2 warehouse loc2) [0] 07: (deliver-to-workstation agent2 ws4 loc2 box3 cart2 slot3) [0] 08: (empty-box-workstation agent2 box3 tool1 tool ws4 loc2) [0] 09: (move agent1 warehouse loc1) [0] 10: (deliver-to-workstation agent1 ws3 loc1 box1 cart1 slot1) [0] 11: (empty-box-workstation agent1 box1 valve1 valve ws3 loc1) [0] 12: (pick-up-from-workstation agent1 ws3 loc1 box1 cart1 slot1) [0] 13: (deliver-to-workstation agent1 ws1 loc1 box2 cart1 slot2) [0] 14: (empty-box-workstation agent1 box2 bolt1 bolt ws1 loc1) [0] 15: (move agent1 loc1 warehouse) [0] 16: (deliver-to-location agent1 warehouse box1 cart1 slot1) [0] 17: (fill-box-from-location agent1 box1 bolt2 warehouse) [0] 18: (pick-up-from-location agent1 box1 warehouse cart1 slot1) [0] 19: (move agent1 warehouse loc2) [0] 20: (deliver-to-workstation agent1 ws2 loc2 box1 cart1 slot1) [0] 21: (empty-box-workstation agent1 box1 bolt2 bolt ws2 loc2) [0] time spent: 0.04 seconds parsing 0.18 seconds encoding 0.40 seconds searching 0.62 seconds total time memory used: 2.40 MBytes for problem representation 0.00 MBytes for searching 2.40 MBytes total

Istanza 2 - Difficile

Il problema fa riferimento al problema descritto al punto [2.2.2]

Risultato *A**	**Risultato FF**
* A* search succeeded found plan as follows: 00: (fill-box-from-location agent1 box1 valve1 warehouse) [0] 01: (fill-box-from-location agent1 box2 bolt2 warehouse) [0] 02: (fill-box-from-location agent1 box3 valve2 warehouse) [0] 03: (pick-up-from-location agent1 box3 warehouse cart1 slot1) [0] 04: (pick-up-from-location agent1 box1 warehouse cart1 slot2) [0] 05: (pick-up-from-location agent2 box2 warehouse cart2 slot3) [0] 06: (move agent1 warehouse loc1) [0] 07: (deliver-to-workstation agent1 ws3 loc1 box1 cart1 slot2) [0] 08: (empty-box-workstation agent1 box1 valve1 valve ws3 loc1) [0] 09: (pick-up-from-workstation agent1 ws3 loc1 box1 cart1 slot2) [0] 10: (move agent1 loc1 warehouse) [0] 11: (deliver-to-location agent1 warehouse box1 cart1 slot2) [0] 12: (fill-box-from-location agent1 box1 tool1 warehouse) [0] 13: (move agent2 warehouse loc1) [0] 14: (deliver-to-workstation agent2 ws1 loc1 box2 cart2 slot3) [0] 15: (pick-up-from-location agent1 box1 warehouse cart1 slot2) [0] 16: (empty-box-workstation agent2 box2 bolt2 bolt ws1 loc1) [0] 17: (move agent1 warehouse loc1) [0] 18: (deliver-to-workstation agent1 ws3 loc1 box1 cart1 slot2) [0] 19: (empty-box-workstation agent1 box1 tool1 tool ws3 loc1) [0] 20: (pick-up-from-workstation agent1 ws3 loc1 box1 cart1 slot2) [0] 21: (move agent1 loc1 warehouse) [0] 22: (deliver-to-location agent1 warehouse box1 cart1 slot2) [0] 23: (fill-box-from-location agent1 box1 tool2 warehouse) [0] 24: (move agent2 loc1 warehouse) [0] 25: (move agent1 warehouse loc3) [0] 26: (move agent1 loc3 loc2) [0] 27: (pick-up-from-location agent2 box1 warehouse cart2 slot3) [0] 28: (move agent2 warehouse loc4) [0] 29: (deliver-to-workstation agent1 ws2 loc2 box3 cart1 slot1) [0] 30: (empty-box-workstation agent1 box3 valve2 valve ws2 loc2) [0] 31: (deliver-to-workstation agent2 ws6 loc4 box1 cart2 slot3) [0] 32: (empty-box-workstation agent2 box1 tool2 tool ws6 loc4) [0] time spent: 0.04 seconds parsing 0.33 seconds encoding 239.27 seconds searching 239.64 seconds total time memory used: 4.53 MBytes for problem representation 627.44 MBytes for searching 631.97 MBytes total	* Starting ENFORCED_HILL_CLIMBING search with FAST_FORWARD heuristic * ENFORCED_HILL_CLIMBING search succeeded found plan as follows: 00: (fill-box-from-location agent1 box1 bolt1 warehouse) [0] 01: (pick-up-from-location agent1 box1 warehouse cart1 slot1) [0] 02: (move agent1 warehouse loc1) [0] 03: (deliver-to-workstation agent1 ws1 loc1 box1 cart1 slot1) [0] 04: (empty-box-workstation agent1 box1 bolt1 bolt ws1 loc1) [0] 05: (fill-box-from-location agent2 box2 valve1 warehouse) [0] 06: (fill-box-from-location agent2 box3 tool1 warehouse) [0] 07: (pick-up-from-location agent2 box2 warehouse cart2 slot3) [0] 08: (pick-up-from-location agent2 box3 warehouse cart2 slot4) [0] 09: (pick-up-from-workstation agent1 ws1 loc1 box1 cart1 slot1) [0] 10: (move agent2 warehouse loc3) [0] 11: (move agent2 loc3 loc2) [0] 12: (deliver-to-workstation agent2 ws2 loc2 box2 cart2 slot3) [0] 13: (empty-box-workstation agent2 box2 valve1 valve ws2 loc2) [0] 14: (move agent1 loc1 warehouse) [0] 15: (move agent1 warehouse loc4) [0] 16: (move agent2 loc2 loc3) [0] 17: (move agent2 loc3 warehouse) [0] 18: (move agent1 loc4 warehouse) [0] 19: (move agent2 warehouse loc4) [0] 20: (deliver-to-workstation agent2 ws6 loc4 box3 cart2 slot4) [0] 21: (empty-box-workstation agent2 box3 tool1 tool ws6 loc4) [0] 22: (deliver-to-location agent1 warehouse box1 cart1 slot1) [0] 23: (fill-box-from-location agent1 box1 tool2 warehouse) [0] 24: (pick-up-from-location agent1 box1 warehouse cart1 slot1) [0] 25: (move agent1 warehouse loc1) [0] 26: (deliver-to-workstation agent1 ws3 loc1 box1 cart1 slot1) [0] 27: (empty-box-workstation agent1 box1 tool2 tool ws3 loc1) [0] 28: (pick-up-from-workstation agent1 ws3 loc1 box1 cart1 slot1) [0] 29: (move agent1 loc1 warehouse) [0] 30: (deliver-to-location agent1 warehouse box1 cart1 slot1) [0] 31: (fill-box-from-location agent1 box1 valve2 warehouse) [0] 32: (pick-up-from-location agent1 box1 warehouse cart1 slot1) [0] 33: (move agent1 warehouse loc1) [0] 34: (deliver-to-workstation agent1 ws3 loc1 box1 cart1 slot1) [0] 35: (empty-box-workstation agent1 box1 valve2 valve ws3 loc1) [0] time spent: 0.04 seconds parsing 0.28 seconds encoding 1.04 seconds searching 1.36 seconds total time memory used: 4.53 MBytes for problem representation 0.00 MBytes for searching 4.53 MBytes total

Istanza 2 con numeric fluents

Il problema fa riferimento al problema descritto al punto [2.3.2]

```
Plan length: 33 step(s).
Makespan   : 83.264
Rescheduled Makespan   : 83.264
Search time: 0.37 seconds - Walltime: 0.440576 seconds
Total time: 0.43 seconds - Walltime: 0.522627 seconds
0.00100000: (fill-box-from-location agent2 box1 bolt2 warehouse) [3.00000000]
3.01100000: (pick-up-from-location agent2 box1 warehouse cart2) [2.00000000]
5.02200000: (move agent2 warehouse loc1) [5.00000000]
10.03300000: (deliver-to-workstation agent2 ws1 loc1 box1 cart2) [2.00000000]
12.04400000: (empty-box-workstation agent2 box1 bolt2 bolt ws1 loc1) [3.00000000]
0.00100000: (fill-box-from-location agent1 box3 valve1 warehouse) [3.00000000]
3.01200000: (pick-up-from-location agent1 box3 warehouse cart1) [2.00000000]
5.02300000: (move agent1 warehouse loc3) [5.00000000]
10.03400000: (move agent1 loc3 loc2) [5.00000000]
15.04400000: (deliver-to-workstation agent1 ws2 loc2 box3 cart1) [2.00000000]
17.05400000: (empty-box-workstation agent1 box3 valve1 valve ws2 loc2) [3.00000000]
15.05400000: (move agent2 loc1 warehouse) [5.00000000]
20.06400000: (fill-box-from-location agent2 box2 tool2 warehouse) [3.00000000]
23.07400000: (pick-up-from-location agent2 box2 warehouse cart2) [2.00000000]
25.08400000: (move agent2 warehouse loc1) [5.00000000]
30.09400000: (deliver-to-workstation agent2 ws3 loc1 box2 cart2) [2.00000000]
32.10400000: (empty-box-workstation agent2 box2 tool2 tool ws3 loc1) [3.00000000]
35.11400000: (pick-up-from-workstation agent2 ws3 loc1 box2 cart2) [2.00000000]
37.12400000: (move agent2 loc1 warehouse) [5.00000000]
42.13400000: (deliver-to-location agent2 warehouse box2 cart2) [2.00000000]
44.14400000: (fill-box-from-location agent2 box2 valve2 warehouse) [3.00000000]
47.15400000: (pick-up-from-location agent2 box2 warehouse cart2) [2.00000000]
49.16400000: (move agent2 warehouse loc1) [5.00000000]
54.17400000: (deliver-to-workstation agent2 ws3 loc1 box2 cart2) [2.00000000]
56.18400000: (empty-box-workstation agent2 box2 valve2 valve ws3 loc1) [3.00000000]
59.19400000: (pick-up-from-workstation agent2 ws3 loc1 box2 cart2) [2.00000000]
61.20400000: (move agent2 loc1 warehouse) [5.00000000]
66.21400000: (deliver-to-location agent2 warehouse box2 cart2) [2.00000000]
68.22400000: (fill-box-from-location agent2 box2 tool1 warehouse) [3.00000000]
71.23400000: (pick-up-from-location agent2 box2 warehouse cart2) [2.00000000]
73.24400000: (move agent2 warehouse loc4) [5.00000000]
78.25400000: (deliver-to-workstation agent2 ws6 loc4 box2 cart2) [2.00000000]
80.26400000: (empty-box-workstation agent2 box2 tool1 tool ws6 loc4) [3.00000000]
```

Capitolo 3: Task 3

Il task tre consiste in due punti:

- Temporal planning: convertire il dominio definito nel capitolo precedente (in particolare per la seconda istanza) in modo da utilizzare le "durative action", cioè azioni che hanno una durata.
- Robotics planning: implementare con PlanSys2 il problema utilizzando le "fake action" sfruttando i piani ottenuti in precedenza.

3.1 Temporal Planning

3.1.1 Dominio

È stata utilizzata la versione del dominio che fa uso dei *fluents*. I tipi sono rimasti invariati; a subire modifiche sono state le azioni che verranno illustrate in seguito.

Un'altra differenza rispetto al dominio con i *fluents* riportato nel capitolo 2 è l'utilizzo di un predicato *free* che fa riferimento ad un *agent*. Il suo utilizzo è necessario al fine di evitare situazioni inconsistenti: ad esempio, stesso *agent* che effettua una *fill* contemporaneamente ad una *pick-up.*

Nella definizione delle azioni sono di fondamentale importanza le seguenti keywords:

- At start: indica che l'espressione che la segue deve essere vera all'inizio dell'azione.
- Over all: indica che l'espressione che la segue deve essere vera dall'inizio alla fine dell'azione.
- At end: indica che l'espressione che la segue deve essere vera alla fine dell'azione.

Rispetto le azioni definite nelle precedenti definizioni del dominio, si aggiunge la possibilità di definire la durata attraverso *:duration* e *:precondition* lascia il posto a *:condition*, dove sono contenute le condizioni che devono essere vere affinché l'azione possa essere eseguita. Gli effetti rimangono contenuti in *:effect*.
Sia nelle *condition* sia negli *effect* si sfruttano le key woards sopra definite.

È stato deciso di utilizzare le seguenti durate:

- Operazioni di *fill*: 3 secondi.
- Operazioni di *empty*: 3 secondi.

- Operazioni di *pick-up*: 2 secondi.
- Operazioni di *move*: 5 secondi.
- Operazioni di *delivery*: 2 secondi.

Di seguito sono riportate le azioni ridefinite tenendo conto che i parametri per ciascuna di esse i parametri rimangono invariati.

- **fill-box-from-location**

```
    (:durative-action fill-box-from-location
        :parameters (?agent - agent ?box - box ?content - content ?location - location)
        :duration (= ?duration 3)
        :condition (and
            (at start (free ?agent))
            (at start (is-empty ?box))
            (at start (at ?content ?location))
            (over all (at ?agent ?location))
            (over all (at ?box ?location))
        )
        :effect (and
            (at start (not (free ?agent)))
            (at start (not (is-empty ?box)))
            (at start (not (at ?content ?location)))
            (at end (contain ?box ?content))
            (at end (free ?agent))
        )
    )
```

- Durata: 3 secondi.
- Condizioni che devono essere vere affinché l'azione possa essere eseguita:
    - All'inizio dell'azione:
        - L'agent deve essere libero.
        - La box da riempire deve essere vuota.
        - Il contenuto deve trovarsi alla location.
    - Per tutta la durata dell'azione:
        - L'agent deve trovarsi alla location.
        - La box deve trovarsi alla location.

- Effetti:
    - All'inizio dell'azione:
        - L'agent non è libero.
        - La box non è più vuota.
        - Il contenuto non è più alla location.
    - Alla fine dell'azione:
        - La box contiene il contenuto.
        - L'agent è libero.

- **fill-box-from-workstation**

```
   (:durative-action fill-box-from-workstation
       :parameters (?agent - agent ?box - box ?content - content ?workstation - workstation ?contentType - contentType ?location - location)
       :duration (= ?duration 3)
       :condition (and
           (at start (free ?agent))
           (at start (is-empty ?box))
           (at start (contain ?workstation ?content))
           (at start (workstation-has-type ?workstation ?contentType))
           (over all (at ?agent ?location))
           (over all (at ?workstation ?location))
           (over all (contain ?workstation ?box))
           (over all (is-type ?content ?contentType))
       )
       :effect (and
           (at start (not (free ?agent)))
           (at start (not (is-empty ?box)))
           (at start (not (contain ?workstation ?content)))
           (at end (not (workstation-has-type ?workstation ?contentType)))
           (at end (contain ?box ?content))
           (at end (free ?agent))
       )
   )
```

- Durata: 3 secondi.
- Condizioni che devono essere vere affinché l'azione possa essere eseguita:
    - All'inizio dell'azione:

        - L’agent deve essere libero.
        - La box da riempire deve essere vuota.
        - La workstation deve avere il contenuto e deve possedere quel tipo di contenuto.
    - Per tutta la durata dell’azione:
        - L’agent deve trovarsi alla location.
        - La workstation deve trovarsi nella location.
        - La workstation deve contenere la box.
        - Il contenuto deve essere del tipo specificato.
- Effetti:
    - All’inizio dell’azione:
        - L’agent non è più libero.
        - La box non è più vuota.
        - La workstation non contiene più il contenuto e il tipo del contenuto.
    - Alla fine:
        - La box contiene il contenuto.
        - L’agent è libero.

- **empty-box-location**

```
   (:durative-action empty-box-location
       :parameters (?agent - agent ?box - box ?content - content ?location - location)
       :duration (= ?duration 3)
       :condition (and
           (at start (free ?agent))
           (at start (contain ?box ?content))
           (over all (at ?agent ?location))
           (over all (at ?box ?location))
       )
       :effect (and
           (at start (not(free ?agent)))
           (at start (not (contain ?box ?content)))
           (at end (at ?content ?location))
           (at end (is-empty ?box))
           (at end (free ?agent))
       )
```

- Durata: 3 secondi.
- Condizioni che devono essere vere affinché l'azione possa essere eseguita:
    - All'inizio:
        - L'agent deve essere libero.
        - La box deve contenere il contenuto.
    - Per tutta la durata:
        - L'agent deve trovarsi alla location.
        - La box deve trovarsi alla location.
- Effetti:
    - All'inizio:
        - L'agent non è più libero.
        - La box non contiene più il contenuto.
    - Alla fine:
        - Il contenuto si trova nella location.
        - La box è vuota.
        - L'agent è libero.

- **empty-box-workstation**

```
    (:durative-action empty-box-workstation
        :parameters (?agent - agent ?box - box ?content - content ?contentType - 
contentType ?workstation - workstation ?location - location)
        :duration (= ?duration 3)
        :condition (and
            (at start (free ?agent))
            (at start (contain ?box ?content))
            (over all (at ?agent ?location))
            (over all (is-type ?content ?contentType))
            (over all (at ?workstation ?location))
            (over all (contain ?workstation ?box))
        )
        :effect (and
            (at start (not (free ?agent)))
            (at start (not (contain ?box ?content)))
            (at end (workstation-has-type ?workstation ?contentType))
            (at end (is-empty ?box))
            (at end (contain ?workstation ?content))
            (at end (free ?agent))
        )
    )
```

- Durata: 3 secondi.
- Condizioni che devono essere vere affinché l'azione possa essere eseguita:
    - All'inizio:
        - L'agent deve essere libero.
        - La box deve contenere il contenuto.
    - Per tutta la durata:
        - L'agent deve trovarsi alla location.
        - Il contenuto deve essere del tipo specificato.
        - La workstation deve trovarsi alla location.
        - La box deve trovarsi alla location.

- Effetti:
    - All'inizio:
        - L'agent non è più libero.
        - La box non contiene più il contenuto.
    - Alla fine:
        - La workstation ha il tipo del contenuto.
        - La box è vuota.
        - La workstation possiede il contenuto.
        - L'agent è libero.

- **pick-up-from-location**

```
   (:durative-action pick-up-from-location
       :parameters (?agent - agent ?box - box ?location - location ?carrier - carrier)
       :duration (= ?duration 2)
       :condition (and
           (at start (free ?agent))
           (over all (< (curr-carrier-load ?carrier)(capacity ?carrier)))
           (at start (at ?box ?location))
           (over all (at ?agent ?location))
           (over all (agent-has-carrier ?agent ?carrier))

       )
       :effect (and
           (at start (not (free ?agent)))
           (at start (not (at ?box ?location)))
           (at end (increase (curr-carrier-load ?carrier) 1))
           (at end (carrier-has-box ?carrier ?box))
           (at end (free ?agent))
       )
   )
```

- Durata: 2 secondi.

- Condizioni che devono essere vere affinché l'azione possa essere eseguita:
    - All'inizio:
        - L'agent deve essere libero.
        - La box deve trovarsi alla location.
    - Per tutta la durata:
        - Il carico corrente deve essere minore della capacità del carrier.
        - L'agent deve trovarsi alla location.
        - L'agent deve avere il carrier.
- Effetti:
    - All'inizio:
        - L'agent non è più libero.
        - La box non è più alla location.
    - Alla fine:
        - Il carico corrente viene aumentato di 1.
        - Il carrier ha la box.
        - L'agent è libero.

- **pick-up-from-workstation**

```
    (:durative-action pick-up-from-workstation
        :parameters (?agent - agent ?workstation - workstation ?location -
location ?box - box                  ?carrier - carrier)
        :duration (= ?duration 2)
        :condition (and
            (at start (free ?agent))
            (at start (contain ?workstation ?box))
            (over all (< (curr-carrier-load ?carrier)(capacity ?carrier)))
            (over all (at ?workstation ?location))
            (over all (at ?agent ?location))
            (over all (agent-has-carrier ?agent ?carrier))
        )
        :effect (and
            (at start (not (free ?agent)))
            (at start (not (contain ?workstation ?box)))
            (at end (increase (curr-carrier-load ?carrier) 1))
            (at end (carrier-has-box ?carrier ?box))
            (at end (free ?agent))
        )
    )
```

- Durata: 2 secondi
- Condizioni che devono essere vere affinché l'azione possa essere eseguita:
    - All'inizio:
        - L'agent deve essere libero.
        - La workstation deve contenere la box.
    - Per tutta la durata:
        - Il carico corrente deve essere minore della capacità del carrier.
        - La workstation deve trovarsi alla location.
        - L'agent deve trovarsi alla location.
        - L'agent deve avere il carrier.
- Effetti:
    - All'inizio:

        - L'agent non è più libero.
        - La workstation non contiene più la box.
    - Alla fine:
        - Il carico corrente viene aumentato di 1.
        - Il carrier ha la box.
        - L'agent è libero.

- **move**

```
   (:durative-action pick-up-from-workstation
       :parameters (?agent - agent ?workstation - workstation ?location -
location ?box - box                  ?carrier - carrier)
       :duration (= ?duration 2)
       :condition (and
           (at start (free ?agent))
           (at start (contain ?workstation ?box))
           (over all (< (curr-carrier-load ?carrier)(capacity ?carrier)))
           (over all (at ?workstation ?location))
           (over all (at ?agent ?location))
           (over all (agent-has-carrier ?agent ?carrier))
       )
       :effect (and
           (at start (not (free ?agent)))
           (at start (not (contain ?workstation ?box)))
           (at end (increase (curr-carrier-load ?carrier) 1))
           (at end (carrier-has-box ?carrier ?box))
           (at end (free ?agent))
       )
   )
```

- Durata: 2 secondi.
- Condizioni che devono essere vere affinché l'azione possa essere eseguita:

    - All’inizio:
        - L’agent deve essere libero.
        - La workstation deve contenere la box.
    - Per tutta la durata:
        - Il carico corrente deve essere minore della capacità del carrier.
        - La workstation deve trovarsi alla location.
        - L’agent deve trovarsi alla location.
        - L’agent deve avere il carrier.
- Effetti:
    - All’inizio:
        - L’agent non è più libero.
        - La workstation non contiene più la box.
    - Alla fine:
        - Il carico corrente viene aumentato di 1.
        - Il carrier ha la box.
        - L’agent è libero.

- **delivery-to-location**

```
    (:durative-action deliver-to-location
        :parameters (?agent - agent ?location - location ?box - box ?carrier -
carrier)
        :duration (= ?duration 2)
        :condition (and
            (at start (free ?agent))
            (over all (carrier-has-box ?carrier ?box))
            (over all (at ?agent ?location))
            (over all (agent-has-carrier ?agent ?carrier))
        )
        :effect (and
            (at start (not (free ?agent)))
            (at end (decrease (curr-carrier-load ?carrier) 1))
            (at end (not (carrier-has-box ?carrier ?box)))
            (at end (at ?box ?location))
            (at end (free ?agent))
        )
    )
```

- Durata: 2 secondi.
- Condizioni che devono essere vere affinché l'azione possa essere eseguita:
    - All'inizio:
        - L'agent deve essere libero.
    - Per tutta la durata:
        - Il carrier ha la box.
        - L'agent deve trovarsi alla location.
        - L'agent ha il carrier.
- Effetti:
    - All'inizio:
        - L'agent non è più libero.
    - Alla fine:
        - Il carico corrente viene decrementato di 1.

- Il carrier non ha più la box.
- La box si trova alla location.
- L'agent è libero.

- **delivery-to-workstation**

```
   (:durative-action deliver-to-workstation
       :parameters (?agent - agent ?workstation - workstation ?location -
location ?box - box ?carrier - carrier)
       :duration (= ?duration 2)
       :condition (and
           (at start (free ?agent))
           (over all(carrier-has-box ?carrier ?box))
           (over all (agent-has-carrier ?agent ?carrier))
           (over all (at ?agent ?location))
           (over all (at ?workstation ?location))
       )
       :effect (and
           (at start (not (free ?agent)))
           (at end (not (carrier-has-box ?carrier ?box)))
           (at end (decrease (curr-carrier-load ?carrier) 1))
           (at end (contain ?workstation ?box))
           (at end (free ?agent))
       )
   )
```

- Durata: 2 secondi.
- Condizioni che devono essere vero affinché l'azione possa essere eseguita:
  - All'inizio:
    - L'agent deve essere libero.
  - Per tutta la durata:
    - Il carrier deve avere la box.
    - L'agent deve avere il carrier.
    - L'agent si trova alla location.
    - La workstation deve essere alla location.

- Effetti:
    - All’inizio:
        - L’agent non è più libero.
    - Per tutta la durata:
        - Il carico corrente viene decrementato di 1.
        - Il carrier non ha più la box.
        - La workstation contiene la box.
        - L’agent è libero.

3.1.2 Risultati

Il problema utilizzato di cui sono riportati i piani è il problema presentato ed utilizzato nel capitolo 2 in particolare nella seconda istanza.

Di seguito sono riportati i piani ottenuti eseguendo tre diversi planner messi a disposizione da *PlanUtils*: Optic, TFD, POPF

- Optic

```
; Plan found with metric 69.020
; States evaluated so far: 83987
; States pruned based on pre-heuristic cost lower bound: 0
; Time 273.02
0.000: (fill-box-from-location agent1 box2 bolt1 warehouse)  [3.000]
0.000: (move agent2 warehouse loc4)  [5.000]
3.001: (pick-up-from-location agent1 box2 warehouse cart1)  [2.000]
5.001: (move agent2 loc4 warehouse)  [5.000]
5.002: (move agent1 warehouse loc1)  [5.000]
10.002: (fill-box-from-location agent2 box3 tool2 warehouse)  [3.000]
10.003: (deliver-to-workstation agent1 ws1 loc1 box2 cart1)  [2.000]
12.004: (empty-box-workstation agent1 box2 bolt1 bolt ws1 loc1)  [3.000]
13.003: (pick-up-from-location agent2 box3 warehouse cart2)  [2.000]
15.004: (move agent2 warehouse loc4)  [5.000]
15.005: (move agent1 loc1 warehouse)  [5.000]
20.005: (deliver-to-workstation agent2 ws6 loc4 box3 cart2)  [2.000]
20.006: (fill-box-from-location agent1 box1 valve1 warehouse)  [3.000]
22.006: (empty-box-workstation agent2 box3 tool2 tool ws6 loc4)  [3.000]
23.007: (pick-up-from-location agent1 box1 warehouse cart1)  [2.000]
25.007: (pick-up-from-workstation agent2 ws6 loc4 box3 cart2)  [2.000]
25.008: (move agent1 warehouse loc3)  [5.000]
27.008: (move agent2 loc4 warehouse)  [5.000]
30.009: (move agent1 loc3 loc2)  [5.000]
32.009: (deliver-to-location agent2 warehouse box3 cart2)  [2.000]
34.010: (fill-box-from-location agent2 box3 tool1 warehouse)  [3.000]
35.010: (deliver-to-workstation agent1 ws2 loc2 box1 cart1)  [2.000]
37.011: (empty-box-workstation agent1 box1 valve1 valve ws2 loc2)  [3.000]
37.011: (pick-up-from-location agent2 box3 warehouse cart2)  [2.000]
39.012: (move agent2 warehouse loc1)  [5.000]
40.012: (pick-up-from-workstation agent1 ws2 loc2 box1 cart1)  [2.000]
42.013: (move agent1 loc2 loc3)  [5.000]
44.013: (deliver-to-workstation agent2 ws3 loc1 box3 cart2)  [2.000]
46.014: (empty-box-workstation agent2 box3 tool1 tool ws3 loc1)  [3.000]
47.014: (move agent1 loc3 warehouse)  [5.000]
52.015: (deliver-to-location agent1 warehouse box1 cart1)  [2.000]
54.016: (fill-box-from-location agent1 box1 valve2 warehouse)  [3.000]
57.017: (pick-up-from-location agent1 box1 warehouse cart1)  [2.000]
59.018: (move agent1 warehouse loc1)  [5.000]
64.019: (deliver-to-workstation agent1 ws3 loc1 box1 cart1)  [2.000]
66.020: (empty-box-workstation agent1 box1 valve2 valve ws3 loc1)  [3.000]
```

Piano generato da Optic

- TFD

```
Plan length: 33 step(s).
Makespan    : 83.264
Rescheduled Makespan    : 83.264
Search time: 0.37 seconds - Walltime: 0.440576 seconds
Total time: 0.43 seconds - Walltime: 0.522627 seconds
0.00100000: (fill-box-from-location agent2 box1 bolt2 warehouse) [3.00000000]
3.01100000: (pick-up-from-location agent2 box1 warehouse cart2) [2.00000000]
5.02200000: (move agent2 warehouse loc1) [5.00000000]
10.03300000: (deliver-to-workstation agent2 ws1 loc1 box1 cart2) [2.00000000]
12.04400000: (empty-box-workstation agent2 box1 bolt2 bolt ws1 loc1) [3.00000000]
0.00100000: (fill-box-from-location agent1 box3 valve1 warehouse) [3.00000000]
3.01200000: (pick-up-from-location agent1 box3 warehouse cart1) [2.00000000]
5.02300000: (move agent1 warehouse loc3) [5.00000000]
10.03400000: (move agent1 loc3 loc2) [5.00000000]
15.04400000: (deliver-to-workstation agent1 ws2 loc2 box3 cart1) [2.00000000]
17.05400000: (empty-box-workstation agent1 box3 valve1 valve ws2 loc2) [3.00000000]
15.05400000: (move agent2 loc1 warehouse) [5.00000000]
20.06400000: (fill-box-from-location agent2 box2 tool2 warehouse) [3.00000000]
23.07400000: (pick-up-from-location agent2 box2 warehouse cart2) [2.00000000]
25.08400000: (move agent2 warehouse loc1) [5.00000000]
30.09400000: (deliver-to-workstation agent2 ws3 loc1 box2 cart2) [2.00000000]
32.10400000: (empty-box-workstation agent2 box2 tool2 tool ws3 loc1) [3.00000000]
35.11400000: (pick-up-from-workstation agent2 ws3 loc1 box2 cart2) [2.00000000]
37.12400000: (move agent2 loc1 warehouse) [5.00000000]
42.13400000: (deliver-to-location agent2 warehouse box2 cart2) [2.00000000]
44.14400000: (fill-box-from-location agent2 box2 valve2 warehouse) [3.00000000]
47.15400000: (pick-up-from-location agent2 box2 warehouse cart2) [2.00000000]
49.16400000: (move agent2 warehouse loc1) [5.00000000]
54.17400000: (deliver-to-workstation agent2 ws3 loc1 box2 cart2) [2.00000000]
56.18400000: (empty-box-workstation agent2 box2 valve2 valve ws3 loc1) [3.00000000]
59.19400000: (pick-up-from-workstation agent2 ws3 loc1 box2 cart2) [2.00000000]
61.20400000: (move agent2 loc1 warehouse) [5.00000000]
66.21400000: (deliver-to-location agent2 warehouse box2 cart2) [2.00000000]
68.22400000: (fill-box-from-location agent2 box2 tool1 warehouse) [3.00000000]
71.23400000: (pick-up-from-location agent2 box2 warehouse cart2) [2.00000000]
73.24400000: (move agent2 warehouse loc4) [5.00000000]
78.25400000: (deliver-to-workstation agent2 ws6 loc4 box2 cart2) [2.00000000]
80.26400000: (empty-box-workstation agent2 box2 tool1 tool ws6 loc4) [3.00000000]
```

Piano generato da TFD

- POPF

```
; States evaluated: 74006
; Cost: 69.020
; Time 261.49
0.000: (fill-box-from-location agent1 box2 bolt1 warehouse)  [3.000]
0.000: (move agent2 warehouse loc4)  [5.000]
3.001: (pick-up-from-location agent1 box2 warehouse cart1)  [2.000]
5.001: (move agent2 loc4 warehouse)  [5.000]
5.002: (move agent1 warehouse loc1)  [5.000]
10.002: (fill-box-from-location agent2 box3 tool2 warehouse)  [3.000]
10.003: (deliver-to-workstation agent1 ws1 loc1 box2 cart1)  [2.000]
12.004: (empty-box-workstation agent1 box2 bolt1 bolt ws1 loc1)  [3.000]
13.003: (pick-up-from-location agent2 box3 warehouse cart2)  [2.000]
15.004: (move agent2 warehouse loc4)  [5.000]
15.005: (move agent1 loc1 warehouse)  [5.000]
20.005: (deliver-to-workstation agent2 ws6 loc4 box3 cart2)  [2.000]
20.006: (fill-box-from-location agent1 box1 valve1 warehouse)  [3.000]
22.006: (empty-box-workstation agent2 box3 tool2 tool ws6 loc4)  [3.000]
23.007: (pick-up-from-location agent1 box1 warehouse cart1)  [2.000]
25.007: (pick-up-from-workstation agent2 ws6 loc4 box3 cart2)  [2.000]
25.008: (move agent1 warehouse loc3)  [5.000]
27.008: (move agent2 loc4 warehouse)  [5.000]
30.009: (move agent1 loc3 loc2)  [5.000]
32.009: (deliver-to-location agent2 warehouse box3 cart2)  [2.000]
34.010: (fill-box-from-location agent2 box3 tool1 warehouse)  [3.000]
35.010: (deliver-to-workstation agent1 ws2 loc2 box1 cart1)  [2.000]
37.011: (pick-up-from-location agent2 box3 warehouse cart2)  [2.000]
37.011: (empty-box-workstation agent1 box1 valve1 valve ws2 loc2)  [3.000]
39.012: (move agent2 warehouse loc1)  [5.000]
40.012: (pick-up-from-workstation agent1 ws2 loc2 box1 cart1)  [2.000]
42.013: (move agent1 loc2 loc3)  [5.000]
44.013: (deliver-to-workstation agent2 ws3 loc1 box3 cart2)  [2.000]
46.014: (empty-box-workstation agent2 box3 tool1 tool ws3 loc1)  [3.000]
47.014: (move agent1 loc3 warehouse)  [5.000]
52.015: (deliver-to-location agent1 warehouse box1 cart1)  [2.000]
54.016: (fill-box-from-location agent1 box1 valve2 warehouse)  [3.000]
57.017: (pick-up-from-location agent1 box1 warehouse cart1)  [2.000]
59.018: (move agent1 warehouse loc1)  [5.000]
64.019: (deliver-to-workstation agent1 ws3 loc1 box1 cart1)  [2.000]
66.020: (empty-box-workstation agent1 box1 valve2 valve ws3 loc1)  [3.000]
```

Piano generato da POPF

Di seguito è riportata una tabella riassuntiva con i principali parametri dei planner:

	Optic	TFD	POPF
Stati valutati	83987	-	74006
Tempo di ricerca	273.02	0.37	261.49
Tempo di esecuzione	66.020	80.264	66.020
Numero mosse	36	33	36

Sfruttando la tabella riassuntiva per effettuare un confronto si evince che TFD è il planner che impiega una quantità di tempo per individuare il piano notevolmente minore degli altri due planner. Il piano da esso generato effettua 3 mosse in più ed ha una durata maggiore di 14s (circa il 22%).

I piani generati da Optic e POPF hanno una esecuzione più veloce con un numero di mosse superiori con un numero notevole di stati visitati (rispettivamente 83987 e 74006), il che porta ad un tempo di ricerca elevato.

3.2 Robotics Planning

Di seguito è riportata l'implementazione con ROS2 Planning System (Plansys2) del problema precedentemente analizzato sfruttando le "fake actions", cioè azioni il cui compito è quello di simulare un'azione senza eseguirla effettivamente in modo da testare il sistema di pianificazione e la logica del piano.

Si avrà una "fake action" per ciascuna azione definita nel dominio e le durate possono corrispondere (come nel caso presentato). Esse sono rappresentati da file C++, è riportato a titolo di esempio il codice per l'azione *fill-box-from-location.*

```
class FillBoxFromLocation : public plansys2::ActionExecutorClient
{
public:
  FillBoxFromLocation()
  : plansys2::ActionExecutorClient("fill_box_from_location", 1500ms)
  {
    progress_ = 0.0;
  }
private:
  void do_work()
  { std :: vector <std :: string > arguments = get_arguments () ;
    if (progress_ < 1.0) {
      progress_ += 0.5;
      send_feedback(progress_, "agent "+arguments [0]+ " is filling "+
            arguments [1]+ " in "+ arguments [3] + " with "+
            arguments [2]);
    } else {
      finish(true, progress_, "agent "+arguments [0]+ " is filling "+
            arguments [1]+ " in "+ arguments [3] + " with "+
            arguments [2]);
      progress_ = 0.0;
      std::cout << std::endl;
    }
       std::cout << arguments[0] + " is filling " + arguments [1]+ " in "+
arguments [3] + " with "+
            arguments [2]+" . . . [ " << std::min(100.0, progress_ * 100.0) <<
"% ] " <<std::flush;
  }
  float progress_;
};


int main(int argc, char ** argv)
{
  rclcpp::init(argc, argv);
```

Ogni 1500ms viene eseguito il metodo *do_work()* il quale ha tra i suoi compiti quello di aumentare il progresso di completamento dell'azione di 0.5. L'azione, quindi, durerà 3 secondi (durata che corrisponde alla durata definita nel dominio). Ad ogni invocazione sul terminale verrà mostrato un messaggio di update circa il completamento dell'azione.

Ogni azione per essere eseguita ha bisogno di un nodo (che corrisponde ai file C++), essi devono essere "lanciati" attraverso un file Python.

Un file di fondamentale importanza è il CMakeLists all'interno del quale vengono definiti gli eseguibili e le dipendenze. In generale si ha un eseguibile per ogni nodo che implementa un'azione.

Per le azioni *fill_box_from_location* e *move* si è deciso di utilizzare due nodi e non uno in quanto durante le esecuzioni dei piani veniva mostrato il WARN "*No action performer for move…"* presumibilmente perché nel piano vi sono *move* che avvengono in parallelo (con due agenti diversi) così come *fill_box_from_location*.

Per fare ciò nel file CMakeLists per le due azioni *move* e *fill_box_from_location* sono presenti due eseguibili che nel file Python launch vengono lanciati insieme agli altri.

È riportato un frammento del file CMakeLists:

```
…
add_executable(fill_box_from_location_action_node src/fill_box_from_location_action_node.cpp)
ament_target_dependencies(fill_box_from_location_action_node ${dependencies})


add_executable(fill_box_from_location_action_node1 src/fill_box_from_location_action_node.cpp)
ament_target_dependencies(fill_box_from_location_action_node1 ${dependencies})
…
install(TARGETS
  fill_box_from_location_action_node
  fill_box_from_location_action_node1
…
)
```

Per completezza è riportato anche un frammento del file Python Launch:

```
…
   fill_box_from_location_cmd = Node(
       package='robotic_planning',
       executable='fill_box_from_location_action_nodÈ,
       name='fill_box_from_location_action_nodÈ,
       namespace=namespace,
       output='screen',
       parameters=[])

fill_box_from_location1_cmd = Node(
       package='robotic_planning',
       executable='fill_box_from_location_action_node1',
       name='fill_box_from_location_action_node1',
       namespace=namespace,
       output='screen',
       parameters=[])
…

   ld.add_action(fill_box_from_location_cmd)
    ld.add_action(fill_box_from_location1_cmd)
  …
```

Per il problema non si può sfruttare direttamente il file PDDL, esso è contenuto nel file commands in cui vengono dichiarate le istanze, i predicati, le funzioni e i goal. Per completezza ne è riportato un estratto:

```
set instance agent1 agent
set instance agent2 agent
set instance cart1 carrier
set instance cart2 carrier
…
set predicate (at agent1 warehouse)
set predicate (at agent2 warehouse)
…
set function (= (curr_carrier_load cart1) 0)
set function (= (curr_carrier_load cart2) 0)
…
set goal (and (workstation_has_type ws1 bolt) (workstation_has_type ws2 valve)
(workstation_has_type ws3 valve) (workstation_has_type ws3 tool) (workstation_has_type ws6 tool) )
```

Per ottenere i risultati che sono riportati successivamente:

- Avviare un terminale all'interno della cartella ed eseguire i seguenti comandi in ordine:
  - colcon build --symlink-install
  - install/setup.bash
  - ros2 launch robotic_planning plansys2_msl.py
- Avviare un secondo terminale all'interno di 3.2/src/robotic_planning ed eseguire:
  - ros2 run plansys2_terminal plansys2_terminal <./launch/commands

  L'esecuzione di questo comando causerà un segmentation fault ma le istanze, i predicati così come funzioni e goal rimangono. Si è provato ad utilizzare il comando "*source pathdelFile"* ma ciò comportava un errore.
- Avviare nuovamente il terminale con il comando:
  - ros2 run plansys2_terminal plansys2_terminal
- Nel terminale lanciato è possibile lanciare i piani:
  - run plan-file ./plans/nomePlan.txt

PlanSys2 permette di eseguire piani già definiti, in particolare sono stati utilizzati i piani ottenuti in precedenza. Per motivi di formattazione non sono stati inseriti nella relazione ma sono presenti nella cartella “Task3/3.2/screenshots”.

Nell’esecuzione dei piani molte volte il processo relativo al nodo “*plansys2_node-1”* termina in modo inaspettato causando il fallimento del piano, non è stata trovata né la causa né la soluzione a tale problema, ma in alcuni run ciò non si verifica.