UNIVERSITÀ DELLA CALABRIA

DIPARTIMENTO DI INGEGNERIA INFORMATICA, MODELLISTICA,
ELETTRONICA E SISTEMISTICA

Corso di Laurea Magistrale in Ingegneria Informatica

Artificial Intelligence Project

PROF. SCARCELLO FRANCESCO
ING. BONO ANTONIO

PASCUZZI GIOVANNI MAT. 252285
MONEA STEFANO FRANCESCO MAT. 256942

ANNO ACCADEMICO 2023/2024

Indice

Capitolo 1

Modelling

L'obiettivo principale di questo elaborato è quello di modellare, implementare e risolvere problemi di pianificazione usando il linguaggio PDDL e algoritmi personalizzati. Il dominio del problema di pianificazione è definito da uno scenario ispirato ai servizi di produzione industriale, dove stazioni di lavoro devono essere rifornite di materiali specifici da agenti robotici.

1.1 Introduzione al Dominio

Nella sezione seguente viene descritto tutto quello che è stato ritenuto utile per la modellazione del dominio in PDDL. In particolare sono stati ritenuti utili, e di conseguenza, definiti diversi tipi di oggetti che rappresentano le entità principali coinvolte nel problema di pianificazione.

1.1.1 Types

Tra gli oggetti definiti troviamo:

→ **Location**: Rappresenta una posizione o un luogo specifico all'interno dell'ambiente di produzione industriale. Le varie location possono includere il magazzino centrale, le varie stazioni di lavoro e altre aree rilevanti per il movimento e la consegna dei materiali. Tale oggetto viene utilizzato per specificare dove si trovano gli agenti robotici, le scatole e le stazioni di lavoro;

→ **Box**: Rappresenta una scatola utilizzata per contenere e trasportare i materiali. Ogni scatola può essere riempita con un determinato contenuto come valvole, bulloni o attrezzi, che nel nostro caso vengono modellati in modo astratto con l'uso di *Content*;

→ **Content**: Rappresenta i diversi materiali che possono essere contenuti nelle scatole. Esempi di contenuti includono valvole, bulloni, attrezzi e altri componenti necessari per le stazioni di lavoro;

→ **Robot**: Rappresenta un agente robotico incaricato di svolgere le attività di trasporto e consegna dei materiali. I robot possono muoversi tra le posizioni, riempire e svuotare scatole e consegnare i materiali alle stazioni di lavoro;

→ **WorkStation**: Rappresenta una stazione di lavoro all'interno dell'ambiente di produzione industriale.

1.1.2 Predicates

Nel dominio sono stati definiti vari predicati, che descrivono le proprietà e le relazioni tra gli oggetti appartenenti ad esso. Questi predicati sono essenziali per modellare correttamente lo stato del sistema e per definire le pre-condizioni e le post-condizioni delle azioni. Tra quelli definiti ritroviamo:

← **at_loc** : Questo predicato indica che un oggetto specifico (?obj), che può essere un agente robotico, una stazione di lavoro, una scatola o un contenuto, si trovi in una certa posizione (?loc);

← **is_empty**: Questo predicato indica che una scatola specifica (?box) è vuota;

← **filled**: Questo predicato indica che una scatola specifica (?box) è riempita con un determinato contenuto (?cont);

← **at_ws**: Questo predicato indica che un oggetto specifico (?obj), che può essere un robot, una scatola o un contenuto, si trova presso una stazione di lavoro specifica (?ws);

← **carrying**: Questo predicato indica che un robot specifico (?robot) sta trasportando una scatola specifica (?box);

← **free**: Questo predicato indica che un robot specifico (?robot) è libero, ovvero non sta trasportando alcuna scatola;

← **connected**: Questo predicato indica che due posizioni (?loc1 e ?loc2) sono collegate tra loro;

← **is_warehouse**: Questo predicato indica in che specifica posizione (?loc) è il magazzino centrale.

1.1.3 Actions

Infine, come ultimo passo per la definizione del nostro dominio troviamo le azioni; esse consentono agli agenti robotici di interagire con l'ambiente circostante e di effettuare le operazioni necessarie per giungere al soddisfacimento del goal. Un'azione è costituita dai parametri (oggetti necessari per il conseguimento dell'obiettivo), dalle precondizioni (condizioni da verificare prima di eseguire l'azione) e dagli effetti (cosa accade in seguito all'esecuzione dell'azione).
In particolare per il nostro dominio sono state definite le seguenti azioni:

... **move_to_loc**: Azione che rappresenta lo spostamento dell'agente robotico, partendo da una posizione iniziale arrivando ad una posizione adiacente.

- **parameters**: il robot che esegue l'azione, posizione iniziale, posizione finale;
- **precondition**:
  - il robot si trovi nella posizione iniziale;
  - le due posizioni (iniziale e finale), sono tra loro interconnesse.
- **effects**:
  - l'agente robotico non si trovi più nella posizione iniziale;
  - l'agente robotico si trova nella posizione finale.

... **enter_ws**: Azione che rappresenta l'ingresso da parte dell'agente robotico all'intero di una workstation;

- **parameters**: l'agente robotico che si occupa di effettuare l'azione, una location ed una workstation;
- **preconditions**:
  * sia la workstation che l'agente robotico devono essere nella stessa posizione;
  * il robot non deve essere già dentro la workstation e deve trovarsi alla locazione passata in input;
  * la workstation ed il robo-agent devono trovarsi nella stessa location;
- **effects**:
  * l'agente robotico non si trova nella location e si trova dentro la workstation;

... **exit_ws**: Tale azione viene usata per rappresentare l'uscita del robot da una stazione di lavoro.

- **parameters**: l'agente robotico che si occupa di effettuare l'azione, una location ed una workstation;
- **preconditions**:
  * l'agente robotico si trova dentro la workstation e non nella location passata in input;
  * la workstation si trova nella location passata nei parametri in input;
- **effects**:
  * l'agente robotico si troverà nella location e non più dentro la workstation.

... **put_down_box_in_ws**: Tale azione viene usata per far si che il robot possa posare la scatola che sta trasportando all'interno di una workstation:

- **parameters**: l'agente robotico che esegue l'azione, la scatola che sta trasportando e la workstation;
- **preconditions**:
  * l'agente robotico deve trovarsi nella workstation passata come parametro;
  * l'agente robotico deve trasportare la scatola che deve posare all'interno della workstation;
- **effects**:
  * l'agente robotico non sta più trasportando la scatola;
  * l'agente robotico diventa disponibile a caricare altre scatole su se stesso;
  * la scatola si trova all'interno della workstation;

... **put_down_box_in_loc**: Simile alla precedente, questa azione permette al robot di posare una scatola nella posizione in cui si trova:

- **parameters**: l'agente robotico che esegue l'azione, la scatola che sta trasportando e la locazione:
- **preconditions**:
  * l'agente robotico deve trovarsi nella locazione passata come parametro;
  * l'agente robotico deve trasportare la scatola che deve posare all'interno della locazione;
- **effects**:
  * l'agente robotico non sta più trasportando la scatola;
  * l'agente robotico diventa disponibile a caricare altre scatole su se stesso;
  * la scatola si trova nella locazione passata come parametro;

... **put_up_box_from_ws**: Questa azione permette al robot di sollevare una scatola presente in una workstation:

- **parameters**: l'agente robotico che esegue l'azione, la scatola e la workstation.

- **preconditions**:
  - la scatola deve trovarsi nella workstation passata come parametro;
  - l'agente robotico non deve trasportare altre scatole;
  - l'agente robotico deve trovarsi nella workstation passata come parametro;
- **effects**:
  - l'agente robotico non sarà più libero;
  - l'agente robotico trasporta la scatola passata come parametro;

... **put_up_box_from_loc**: Azione analoga a quella precedente, permette all'agente robotico di sollevare una scatola da una location;

- **parameters**: l'agente robotico che esegue l'azione, la scatola e la locazione;
- **preconditions**:
  - la scatola deve trovarsi nella locazione passata come parametro;
  - l'agente robotico non deve trasportare altre scatole;
  - l'agente robotico deve trovarsi nella locazione passata come parametro;
- **effects**:
  - l'agente robotico non sarà più libero;
  - l'agente robotico trasporta la scatola passata come parametro;

... **empty_box_in_ws**: L'agente robotico svuota una scatola nella workstation, trasferendo il contenuto nella stessa.

- **parameters**: l'agente robotico che esegue l'azione, la scatola, il contenuto e la workstation;
- **preconditions**:
  - l'agente robotico deve essere libero, quindi non deve trasportare qualcosa;

    * l'agente robotico deve trovarsi nella workstation passata come parametro;
    * la scatola deve essere riempita con il contenuto passato come parametro;
  - **effects**:
    * la scatola non sarà più riempita con il contenuto;
    * la scatola diventerà vuota;
    * il contenuto si troverà nella workstation passata come parametro;

... **fill_box_in_ws**: Con questa azione, l'agente robotico può riempire una scatola vuota prendendo il contenuto dalla workstation;

  - **parameters**: l'agente robotico che deve eseguire l'azione, la scatola, la workstation e il contenuto;
  - **preconditions**:
    * l'agente robotico deve essere libero e deve trovarsi nella workstation passata come parametro;
    * la scatola deve essere vuota e deve trovarsi nella workstation passata come parametro;
    * il contenuto deve essere nella workstation;
  - **effects**:
    * il contenuto non si troverà più nella workstation passata in input;
    * la scatola non sarà più vuota e sarà riempita con il contenuto passato come parametro;

... **empty_box_in_loc**: L'agente robotico svuota una scatola in una location, se disponibile, trasferendo il contenuto nella stessa posizione.

  - **parameters**: l'agente robotico che esegue l'azione, il contenuto, la scatola e una locazione;
  - **preconditions**:

    * l'agente robotico deve essere libero e deve trovarsi nella stessa location passata in input, stessa della scatola;
    * la scatola deve essere riempita col contenuto passato come parametro e deve trovarsi nella stessa location del robot;
  - **effects**:
    * la scatola sarà vuota e non più riempita con il contenuto passato come parametro;
    * il contenuto si troverà nella location passata in input;

... **fill_box_in_loc**: Il robo-agent se disponibile, e si trova in una location con un box vuoto, lo riempie prendendo il contenuto dalla stessa location.

  - **parameters**: l'agente robotico che esegue l'azione, la scatola, il contenuto e la locazione;
  - **preconditions**:
    * la locazione non deve essere una workstation;
    * l'agente robotico deve essere libero e trovarsi nella stessa location della scatola e del contenuto, posizione che viene fornita in input;
    * la scatola deve essere vuota;
  - **effects**:
    * il contenuto non si troverà più in location;
    * la scatola sarà riempita con il contenuto;
    * la scatola non sarà più vuota;

... **fill_box_in_warehouse**: L'agente robotico può riempire una scatola vuota senza consumare alcun contenuto specifico solo se il robot e la scatola si trovano nella warehouse e la scatola è vuota;

  - **parameters**: l'agente robotico che esegue l'azione, la scatola, il contenuto e la locazione;
  - **preconditions**:
    * la warehouse, la scatola, il contenuto e il robo-agent devono trovarsi tutti nella stessa location;

    - la scatola deve essere vuota;
    - l'agente robotico deve essere libero;
  - **effects**:
    - la scatola è riempita con il contenuto e non è più vuota;

1.2 Easy Istance

Per verificare che il dominio sia correttamente inizializzato, creiamo un istanza molto semplice del problema, di cui le specifiche di seguito riportate:

- Tre location presenti nel dominio;
- Due scatole per il trasporto del materiale;
- Due contenuti (bolts e valves) che devono essere posati dentro le scatole;
- Un Robot-Agent per l'esecuzione del lavoro;
- Due workstation nella quale possono essere svolte diverse operazioni.

Lo stato iniziale si presenta cosi:

- Il robot-agent si trova nella warehouse ed è libero;
- La scatola (box1) è vuota e si trova nella warehouse;
- La scatola (box2) si trova nella warehouse ed è riempito con bolts;
- Bolts e Valves si trovano nella warehouse;
- La workstation1 si trova in location1;
- La workstation2 si trova in location2;
- Location1 è connessta a Location2 e quest'ultima è connessa alla Warehouse, mentre la warehouse è connessa alla location1;

Il nostro obiettivo è il seguente:

- Box1 deve trovarsi nella workstation1 e deve essere riempito con bolts;
- Box2 deve trovarsi nella workstation2 e deve essere riempito con valves;

```
(define (problem easy_istance)
  (:domain manufactor_services)
  (:objects
    loc1 loc2 warehouse - location
    box1 box2 - box
    bolts valves - content
    robot - robot
    ws1 ws2 - workstation
  )
  (:init
    (at_loc robot warehouse)
    (at_loc box1 warehouse)
    (at_loc box2 warehouse)
    (at_loc bolts warehouse)
    (at_loc valves warehouse)

    (is_empty box1)
    (is_empty box2)
    (free robot)

    (connected loc1 loc2)
    (connected loc2 warehouse)
    (connected warehouse loc1)

    (at_loc ws1 loc1)
    (at_loc ws2 loc2)
    (is_warehouse warehouse)
  )

  (:goal
    (and
      (at_ws box1 ws1)
      (at_ws box2 ws2)
```

```
      (filled box1 bolts)
      (filled box2 valves)
    )
  )
)
```

Utilizzando il framework PDDL4J, e come planner uno di default, FastForward, otteniamo la seguente soluzione:

```
problem instantiation done successfully (67 actions, 40 fluents)

* Starting ENFORCED_HILL_CLIMBING search with FAST_FORWARD heuristic
* ENFORCED_HILL_CLIMBING search succeeded

found plan as follows:

00: ( fill_box_in_warehouse robot box1 bolts warehouse) [0]
01: (fill_box_in_warehouse robot box2 valves warehouse) [0]
02: (          put_up_box_from_loc robot box1 warehouse) [0]
03: (                  move_to_loc robot warehouse loc1) [0]
04: (                           enter_ws robot loc1 ws1) [0]
05: (                 put_down_box_in_ws robot box1 ws1) [0]
06: (                            exit_ws robot loc1 ws1) [0]
07: (                       move_to_loc robot loc1 loc2) [0]
08: (                   move_to_loc robot loc2 warehouse) [0]
09: (          put_up_box_from_loc robot box2 warehouse) [0]
10: (                  move_to_loc robot warehouse loc1) [0]
11: (                       move_to_loc robot loc1 loc2) [0]
12: (                           enter_ws robot loc2 ws2) [0]
13: (                 put_down_box_in_ws robot box2 ws2) [0]

time spent:       0.09 seconds parsing
                  0.13 seconds encoding
                  0.02 seconds searching
                  0.24 seconds total time

memory used:      0.34 MBytes for problem representation
                  0.00 MBytes for searching
                  0.34 MBytes total
```

Capitolo 2

Classical Planning

In questa sezione ci concentreremo nello sviluppo di un planner personalizzato, sfruttando il framework PDDL4J, per la risoluzione di due istanze fornitoci.

2.1 Istance 1

Per la prima istanza si farà riferimento al dominio riportato in precedenza, e lo stato di partenza per la prima istanza è riportato brevemente di seguito:

- Tutte le scatole sono inizialmente collocate in un'unica location denominata "magazzino centrale" (*warehouse*). Questo implica che non vi sono scatole sparse in altre location, semplificando la gestione iniziale delle risorse;
- Tutti i contenuti che dovranno essere caricati nelle scatole si trovano inizialmente nel magazzino centrale;
- Nel magazzino centrale non sono presenti workstation;
- Un singolo agente robotico è posizionato nel magazzino centrale con il compito di consegnare le scatole;

Da sottolineare che non vengono forniti alcun tipo di restrizioni particolari sul numero di scatole disponibili. Questa flessibilità permette di modellare il numero di scatole in base alle esigenze specifiche della soluzione progettuale adottata, nel nostro caso saranno definite 3 scatole.

```
(define (problem istance_1)
  (:domain  manufacturing_services)
  (:objects
    robot - robot
    warehouse location1 location2 - location
    workstation1 workstation2 workstation3 - workstation
    box1 box2 box3 - box
    bolt screw - content
  )
```

Per quanto riguarda lo stato iniziale:

```
    (:init
    (at_loc robot warehouse)

    (is_warehouse warehouse)

    (at_loc box1 warehouse)
    (at_loc box2 warehouse)
    (at_loc box3 warehouse)

    (at_loc bolt warehouse)
    (at_loc screw warehouse)

    (is_empty box1)
    (is_empty box2)
    (is_empty box3)

    (at_loc workstation1 location1)
    (at_loc workstation2 location2)
    (at_loc workstation3 location2)

    (connected warehouse location1)
    (connected location1 location2)
    (connected location1 warehouse)
```

```
    (connected location2 location1)

    (free robot)
  )
```

Prestiamo una piccola attenzione a questi due predicati:

```
(connected warehouse location1)
(connected location1 location2)
(connected location1 warehouse)
(connected location2 location1)
```

Come possiamo notare sono ripetuti dove però varia la posizione degli argomenti, questo è stato inserito appositamente per simulare il fatto che il predicato sia transitivo. Per capire meglio cosa intendiamo per *predicato transitivo* supponiamo l'esempio:

```
amico(X,Y)
```

```
con amico(X,Y) intendiamo che X è amico di Y, ma se X è amico
di Y, allora sicuramente anche Y sarà amico di X.
Per modellare questo concetto viene aggiunto un nuovo predicato
nella forma:
```

```
amico(Y,X)
```

Per quanto riguarda il nostro obiettivo è il seguente:

```
(:goal
    (and
      ; Almeno una workstation che necessita di un bullone
      (at_ws bolt workstation2)
      ; Almeno una workstation che non necessita di nulla
      (not (at_ws bolt workstation1))
      (not (at_ws screw workstation1))
      ; Almeno una workstation che necessita di bulloni e viti
      (at_ws bolt workstation3)
      (at_ws screw workstation3)
    )
```

)

Prima di passare alla definizione del planner, focalizziamoci prima sulla definizione della seconda istanza.

2.2 Istance 2

Per risolvere il problema descritto nella seconda istanza, bisogna considerare le seguenti estensioni rispetto all'istanza precedentemente descritta. Di seguito sono riportati i requisiti riassuntivi ed importanti:

- Ogni agente robotico ha un carrello con una capacità di carico massima, che può essere diversa per ogni agente;
- Gli agenti robotici possono caricare scatole nel trasportatore fino al raggiungimento della capacità massima;
- Per ogni agente robotico, è necessario contare e tenere traccia di:
  - I quali scatole sono su ogni trasportatore;
  - II quante scatole ci sono in totale su ogni trasportatore, in modo che i trasportatori non possano essere sovraccaricati.

Prima di passare ad analizzare come è stato definita l'istanza del problema, lasceremo al lettore, l'interpretazione di due strade alternative, anticipando già che la strada secondaria sarà poi quella adottata come soluzione finale.

2.2.1 1 Way

Richiamando uno dei punti principali dell'elenco precedente:

- Ogni agente robotico ha un carrello con una capacità di carico massima, che può essere diversa per ogni agente;

Ci rendiamo conto di aver bisogno subito dei *:fluents*, in modo tale da ottenere delle variabili che possono cambiare valore durante l'esecuzione del piano. Di seguito verranno elencate le differenze rispetto al dominio descritto in *Modelling*:

```
(:requirements :strips :typing :equality :fluents)
```

Innanzitutto nei *:requirements* andiamo ad aggiungere *:fluents*.

```
(:types
    location box content robot workstation carrier
)
```

Notiamo l'aggiunta di un nuovo tipo, *carries*, usato appunto per rappresentare il carrello con la quale gli oggetti si muovono. Per quanto riguarda i predicati, possiamo notare:

```
(joined ?robot - robot ?carrier - carrier)
```

Dove:

- **joined**: viene usato per rappresentare il fatto che un determinato robo-agent possieda un carrello.

Notiamo anche l'assenza del predicato *free*, in quanto in questa istanza il robot non trasporta più le diverse scatole su se stesso ma utilizza il carrello.

```
(:functions
    (capacity ?carrier - carrier)
    (load ?carrier - carrier)
)
```

Abbiamo l'aggiunta di queste due nuovi funzioni, che indicano rispettivamente la capacità massima e il carico corrente del carrello.
Ogni azione definita nel dominio della sezione precedente è stata dovuta ridefinire andando a considerare questo nuovo tipo *carrier*, per esempio supponiamo questa azione:

```
(:action move_to_loc
    :parameters (?r - robot ?from
    ?to - location ?car - carrier)
:precondition (and
    (at_loc ?r ?from)
    (connected ?from ?to)
```

```
        (not (is_warehouse ?to))
        (joined ?r ?car)
        )
    )
    :effect (and
        (not (at_loc ?r ?from))
        (at_loc ?r ?to)
    )
)
```

Chiaramente in questo caso, cosi come nei prossimi, dobbiamo tenere in considerazione del carrello.
A questo punto capite le modifiche apportate al dominio, ritorniamo alla definizione della nostra istanza, lo stato iniziale si presenta cosi:

- Inizialmente tutte le scatole sono posizionate nella warehouse;
- Tutti i contenuti da caricare nelle scatole sono nella warehouse;
- I robo-agent sono collocati nella warehouse;
- Ogni robo-agent è inizialmente vuoto;
- La capacità di un robo-agent è un valore che deve essere maggiore di 1.

```
    (:objects
        robot - robot
        carrier - carrier
        warehouse location1 location2 - location
        workstation1 workstation2 workstation3 - workstation
        box1 box2 box3 - box
        bolt screw - content
    )

    (:init
        (at_loc robot warehouse)
        (joined robot carrier)
```

```
        (is_warehouse warehouse)

        (at_loc box1 warehouse)
        (at_loc box2 warehouse)
        (at_loc box3 warehouse)
        (at_loc bolt warehouse)
        (at_loc screw warehouse)

        (is_empty box1)
        (is_empty box2)
        (is_empty box3)

        (at_loc workstation1 location1)
        (at_loc workstation2 location2)
        (at_loc workstation3 location2)

        (connected warehouse location1)
        (connected location1 location2)
        (connected location1 warehouse)
        (connected location2 location1)

        (= (capacity carrier) 2)
        (= (load carrier) 0)
    )
```

L'obiettivo è il seguente:

```
    (:goal
        (and
            (at_ws bolt workstation2)

            (not (at_ws bolt workstation1))
            (not (at_ws screw workstation1))
```

```
        (at_ws bolt workstation3)
        (at_ws screw workstation3)
    )
)
```

Questa prima strada è da abbandonare temporaneamente, in quanto il planner che verrà definito successivamente mediante il framework PDDL4J, non è in grado di risolvere i problemi che fanno uso dei *:fluents*. Per valutare la correttezza però abbiamo usato un editor online, editor.planning.domains, che ci ha fornito la seguente soluzione:

```
(fill_box_in_warehouse robot box1 bolt warehouse carrier)
(load_boxes_from_loc robot box1 warehouse carrier)
(move_to_loc robot warehouse location1 carrier)
(move_to_loc robot location1 location2 carrier)
(enter_ws robot location2 workstation2 carrier)
(put_down_box_in_ws robot box1 workstation2 carrier)
(empty_box_in_ws robot box1 workstation2 bolt carrier)
(exit_ws robot location2 workstation2 carrier)
(move_to_loc robot location2 location1 carrier)
(move_to_warehouse robot location1 warehouse carrier)
(fill_box_in_warehouse robot box2 screw warehouse carrier)
(load_boxes_from_loc robot box2 warehouse carrier)
(move_to_loc robot warehouse location1 carrier)
(move_to_loc robot location1 location2 carrier)
(enter_ws robot location2 workstation3 carrier)
(put_down_box_in_ws robot box2 workstation3 carrier)
(empty_box_in_ws robot box2 workstation3 screw carrier)
(exit_ws robot location2 workstation3 carrier)
(move_to_loc robot location2 location1 carrier)
(move_to_warehouse robot location1 warehouse carrier)
(fill_box_in_warehouse robot box3 bolt warehouse carrier)
(load_boxes_from_loc robot box3 warehouse carrier)
(move_to_loc robot warehouse location1 carrier)
```

```
(move_to_loc robot location1 location2 carrier)
(enter_ws robot location2 workstation3 carrier)
(put_down_box_in_ws robot box3 workstation3 carrier)
(empty_box_in_ws robot box3 workstation3 bolt carrier)
```

2.2.2 2 Way

In questa strada invece verranno apportate modifiche al dominio precedente per andare a rimuovere i *:fluents*, in particolare avremo:

```
(:requirements :strips :typing :equality :adl)
(:types
    location box content robot workstation carrier slot
)
```

Notiamo subito l'aggiunta di un nuovo *types*, ossia *slot* che indica una locazione disponibile per posizionare una scatola sul carrello.
Le modifiche apportate ai predicati sono:

```
(:predicates
    (joined ?robot - robot ?carrier - carrier)
    (handle ?car - carrier ?slot - slot)
    (free ?slot - slot)
)
```

In particolare avremo:

- Aggiunta del predicato *joined*, esso serve per indicare quale carrello appartiene al robot;
- Aggiunta del predicato *handle*, per esprimere il concetto di spazio nel carrello, in altri termini per dire quanti *slot* disponibili ha un carrello;
- Aggiunta del predicato *free*, dove viene usato per stabilire se un determinato slot è libero o meno;

Non avremo più le *:functions* nel nostro dominio e le varie azioni sono state modificate andando a considerare l'aggiunta del tipo *slot*.
La differenza sostanziale la possiamo vedere in:

```
  ---------------------- PRIMA -------------------
  (:action move_to_warehouse
      :parameters (?r - robot ?from ?to - location ?car - carrier)
      :precondition (and
          (at_loc ?r ?from)
          (connected ?from ?to)
          (is_warehouse ?to)
          (joined ?r ?car)
          (= (load ?car) 0)
      )
      :effect (and
          (not (at_loc ?r ?from))
          (at_loc ?r ?to)
      )
  )
  ----------------------DOPO -------------------
(:action move_to_warehouse
  :parameters (?r - robot ?from ?to - location ?car - carrier)
  :precondition (and
      (at_loc ?r ?from)
      (connected ?from ?to)
      (is_warehouse ?to)
      (joined ?r ?car)
      (forall (?slot - slot)
          (and
              (handle ?car ?slot)
              (free ?slot)
              (joined ?r ?car)
              (not (loading ?car))
          )
      )
  )
  :effect (and
```

```
        (not (at_loc ?r ?from))
        (at_loc ?r ?to)
    )
  )
```

Passeremo ora alla spiegazione del planner implementato.

2.3 Planner Details

La sezione attuale fornisce una descrizione dettagliata del planner implementato mediante il framework PDDL4J. In particolare avremo che facendo riferimento alla guida, fornita dagli sviluppatori del framework stesso, avremo la classe del nostro planner denominato "*ManufacturingServicesPlanner*" che estende la classe astratta *AbstractPlanner*.
Oltre i metodo di base sono stati aggiunti i seguenti metodi e variabili:

```
/*
 * Campo aggiuntivo per:
 * useNewHeuristic = per poter usare l'ueristica
 * definita ad Hoc
 */
private int useNewHeuristic;
```

Di conseguenza i relativi setter e getter, dove da sottolineare in questo caso:

```
//Metodo per settare la nuova euristica
// 0 -> Si usa l'algorimto di ricerca con un eristica di base
// 1 -> Si usa l'algoritmo di ricerca con l'euristica
//definita ad hoc
@CommandLine.Option(names = {"-en", "--heuristicNew"},
defaultValue = "0",
        description = "Set for the heuristic : 1")
public void setHeuristicNew(int heuristicsNew) {
    this.useNewHeuristic = heuristicsNew;
}
```

In questo caso è stata concessa la possibilità all'utente di poter definire da riga di comando direttamente quale euristica utilizzare:

- **-en 0**: si usa l'algoritmo di ricerca con un euristica di base definita con il comando -*n*;
- **-en 1**: si usa l'algoritmo di ricerca con l'euristica custom, in questo caso non è necessario inserire da riga di comando -*n*.

Prima di parlare dell'algoritmo A* modificato, vi è da sottolineare, come richiesto dalla guida, la definizione di una nuova classe denominata *Node*, utile per la definizione dell'algoritmo di ricerca custom. Brevemente riportata di seguito l'euristica:

```
public final class Node extends State {


     //Il genitore del nodo.
    private Node parent;
    //L'azione applicata per raggiungere tale nodo
    private int action;
    //Il costo per raggiungere il nodo dal nodo iniziale
    private double cost;
    //La distanza stimata per raggiungere il goal
    private double heuristic;
    //La profondità del nodo
    private int depth;

    public Node(State state) {
        super(state);
    }

    public Node(State state, Node parent, int action, double cost, double heurist
        super(state);
        this.parent = parent;
        this.action = action;
        this.cost = cost;
```

```
        this.heuristic = heuristic;
        this.depth = -1;
    }

    public Node(State state, Node parent, int action,
    double cost, int depth, double heuristic) {
        super(state);
        this.parent = parent;
        this.action = action;
        this.cost = cost;
        this.depth = depth;
        this.heuristic = heuristic;
    }

   //Setter e g+Getter delle diverse variabili di classe
   //Funzione che si occua di calcolare il risultato di F.
    public final double getValueF(double weight) {
        return weight * this.heuristic + this.cost;
    }
}
```

Di seguito viene riportato l'algoritmo modificato di A*:

```
//Algoritmo A* modificato ad hoc
public Plan customAstar(Problem problem){

    //Costruiamo la nostra euristica
    ManufacturingServicesHeuristic heuristic =
    new ManufacturingServicesHeuristic(problem);

    // Ci prendiamo lo stato iniziale
    final State init = new State(problem.getInitialState());

    // Inizializziamo l'insieme dei nodi visitati
    final Set<Node> close = new HashSet<>();
```

```
final double weight = this.getHeuristicWeight();
final PriorityQueue<Node> open = new PriorityQueue<>
(100, new Comparator<Node>() {
    public int compare(Node n1, Node n2) {
       double f1 = weight * n1.getHeuristic() + n1.getCost();
       double f2 = weight * n2.getHeuristic() + n2.getCost();
        return Double.compare(f1, f2);
    }
});

// Definiamo la radice dell'albero di ricerca
final Node root = new Node(init, null, -1, 0,
heuristic.estimate(init, problem.getGoal()));

open.add(root);
Plan plan = null;

// Settiamo il timeout per la ricerca
final int timeout = this.getTimeout() * 1000;
long time = 0;

// Inizio ricerca
while (!open.isEmpty() && plan == null && time < timeout) {

    // Aggiungiamo il nodo a quelli visitati
    final Node current = open.poll();
    close.add(current);

    //Se il goal è soddisfatto allora restituiamo il plan
    if (current.satisfy(problem.getGoal())) {
        return this.extractPlan(current, problem);
    } else {
```

```
            //Altrimenti proviamo ad applicare le azioni al nodo
            for (int i = 0; i < problem.getActions().size(); i++) {
                // Ci prendiamo le azioni
                Action a = problem.getActions().get(i);
                // Verifichiamo se l'azione è applicabile al problema
                if (a.isApplicable(current)) {
                    Node next = new Node(current);
                    // Applichiamo l'effetto dell'azione
                    final List<ConditionalEffect> effects =
                    a.getConditionalEffects();
                    for (ConditionalEffect ce : effects) {
                        if (current.satisfy(ce.getCondition())) {
                            next.apply(ce.getEffect());
                        }
                    }
                    //Andiamo ad inserire le informazioni al nuovo nodo
                    final double g = current.getCost() + 1;
                    if (!close.contains(next)) {
                        next.setCost(g);
                        next.setParent(current);
                        next.setAction(i);
                        next.setHeuristic(heuristic.customEstimate(next,
                        problem.getGoal(), a, current.getHeuristic()));
                        open.add(next);
                    }
                }
            }
        }
    }
    //Restituiamo il plan -> Null se vuoto
    return plan;
}
```

Prestiamo adesso attenzione al metodo *solve* di seguito riportato:

```
//Metodo principale della classe
@Override
public Plan solve(final Problem problem) throws ProblemNotSupportedException {
    //Andiamo a vedere per prima cosa se il problema è risolvibile dal plan costr
    if (!this.isSupported(problem)) {
    throw new ProblemNotSupportedException("Cannot solve the problem");}

    //ASTAR
    //mediante la seguente variabile di classe andiamo a scegliere
    l'euristica di riferimento
    if (this.getHeuristicNew() == 1){
        LOGGER.info("* Starting A* Search with
        Manufacturing Services Heuristic \n");
        //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();
        //Lanciamo l'algoritmo
        final Plan plan = this.customAstar(problem);
        final long end = System.currentTimeMillis();

        MemoryUsage afterHeapMemoryUsage = memoryBean.getHeapMemoryUsage();
        long afterUsedMemory = afterHeapMemoryUsage.getUsed();
        long memoryUsedByCustomAstar = afterUsedMemory - beforeUsedMemory;
        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 plan;
    } else{
        //In questo caso usiamo l'algoritmo A* con una
        delle euristiche fornite dal framework
        LOGGER.info("* Starting A* search with heuristic: "
        + this.getHeuristic()+"\n");
        StateSpaceSearch search =
        StateSpaceSearch.getInstance(SearchStrategy.Name.ASTAR,
                    this.getHeuristic(), this.getHeuristicWeight(),
                    this.getTimeout());
        LOGGER.info("* Starting A* search \n");
        // Cerchiamo una soluzione
        Plan plan = search.searchPlan(problem);
        if (plan != null) {
            LOGGER.info("* A* search succeeded\n");
            this.getStatistics().setTimeToSearch(search.getSearchingTime());
            this.getStatistics().setMemoryUsedToSearch(
            search.getMemoryUsed());
        } else {
            LOGGER.info("* A* search failed\n");
        }
        return plan;
    }
}
```

Nel metodo *solver* viene mostrato come si fa uso del campo della classe introdotto precedentemente, rispettivamente per la scelta dell'euristica da usare in combinata all'algortimo A*.

2.4 Heuristics

Le funzioni euristiche usate per risolvere le istanze precedentemente esposte sono:

- **Fast-Forward**: L'euristica FastForward, spesso abbreviata come FF, è un'euristica basata sui piani rilassati. È stata introdotta da J. Hoffmann e deriva dall'algoritmo Graphplan, il quale provvede alla costruzione di un grafo a livelli alternati di proposizioni e azioni. Tale grafo serve a rappresentare le possibili evoluzioni di uno stato iniziale verso uno stato finale;

- **AdjustedSum**: L'euristica AdjustedSum è un'euristica additiva, proposta da Nguyen and Kambhampati, che tenta di migliorare la stima sommando i costi delle singole sotto-attività, ma con aggiustamenti per tenere conto delle interazioni tra le attività.

- **ManufacturingServicesHeuristic**: Un euristica costruita ad hoc, di cui successivamente verrà descritto brevemente il codice, che tende a stimare l'euristica dei nodi futuri in base al valore dell'euristica del nodo corrente e dell'azione che bisogna effettuare.

L'euristica definita da noi si presenta in questo modo:

```
public final class ManufacturingServicesHeuristic
    extends RelaxedGraphHeuristic {

    public ManufacturingServicesHeuristic(Problem problem) {
        super(problem);
        super.setAdmissible(false);
    }

    @Override
    public int estimate(State state, Condition goal) {
        super.setGoal(goal);
        super.expandRelaxedPlanningGraph(state);
        return super.isGoalReachable() ? 1 : Integer.MAX_VALUE;
    }

    @Override
```

```
    public double estimate(Node node, Condition goal) {
        return this.estimate((State) node, goal);
    }

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

        double heuristicValue = currentHeuristic;

        if(a.getName().contains("move"))
            heuristicValue -= 1;
        else if (a.getName().contains("fill")
            || a.getName().contains("ws")
            || a.getName().contains("load") )
            heuristicValue -= 0.8;
        else if (a.getName().equals("empty_box_in_loc"))
            heuristicValue += 0.5;
        return super.isGoalReachable() ? heuristicValue
            : Integer.MAX_VALUE;
    }
}
```

Il fuzionamento dell'euristica è il seguente: partendo dal valore dell'euristica dello stato corrente, si calcola il valore di uno stato futuro in base all'azione che viene compiuta:

- **Movimento**: viene sottratto 1 al valore corrente dell'euristica;
- **Scaricamenti e Caricamenti**: viene sottratto 0.8 all'euristica corrente se l'azione da eseguire è un'azione che riguarda la workstation, il caricamento e il riempimento di box;

- **Svuotamento box nelle location**: questa azione che viene penalizzata, in quanto noi vogliamo che venga massimizzato il più possibile la consegna degli oggetti nelle diverse workstation.

Possiamo notare la definizione del metodo *Estimate* che eredita dalla classe *RelaxedGraphHeuristic*, dove vi è stata appostata una modifica al metodo principale, in quanto restitusce un valore compreso tra 1 e *INTEGER.MAX_VALUE*. Questo perchè in questo modo garantiamo che nella prima iterazione l'euristica non possa diventare negativa. Infatti riportando brevemente una delle prime righe del metodo *custom* presente nella classe principale del planner:

```
final Node root = new Node(init, null, -1, 0,
    heuristic.estimate(init, problem.getGoal()));
```

Che rappresenta sostanzialmente la definizione del nodo radice, abbiamo che viene invocato direttamente *estimate*, visto precedentemente, e se non si fosse apportata tale modifica, il calcolo delle euristiche precedenti sarebbe potuto diventare negativo. Il nuovo metodo dell'euristica, invece, viene cosi richiamato:

```
if (!close.contains(next)) {
    next.setCost(g);
    next.setParent(current);
    next.setAction(i);
    next.setHeuristic(heuristic.customEstimate
    (next, problem.getGoal(), a, current.getHeuristic()));
}
```

2.5 Result

Seguiranno tutti i planner ottenuti con l'algoritmo a* e le rispettive euristiche:

2.5.1 Istance 1

Il plan trovato da **Fast-Forward**, è il seguente:

```
00: ( fill_box_in_warehouse robot box1 bolt warehouse) [0]
01: (        put_up_box_from_loc robot box1 warehouse) [0]
02: (           move_to_loc robot warehouse location1) [0]
03: (           move_to_loc robot location1 location2) [0]
04: (           enter_ws robot location2 workstation2) [0]
05: (      put_down_box_in_ws robot box1 workstation2) [0]
06: (    empty_box_in_ws robot box1 workstation2 bolt) [0]
07: (            exit_ws robot location2 workstation2) [0]
08: (           move_to_loc robot location2 location1) [0]
09: (           move_to_loc robot location1 warehouse) [0]
10: ( fill_box_in_warehouse robot box2 bolt warehouse) [0]
11: (        put_up_box_from_loc robot box2 warehouse) [0]
12: (           move_to_loc robot warehouse location1) [0]
13: (           move_to_loc robot location1 location2) [0]
14: (           enter_ws robot location2 workstation3) [0]
15: (      put_down_box_in_ws robot box2 workstation3) [0]
16: (    empty_box_in_ws robot box2 workstation3 bolt) [0]
17: (            exit_ws robot location2 workstation3) [0]
18: (           move_to_loc robot location2 location1) [0]
19: (           move_to_loc robot location1 warehouse) [0]
20: (fill_box_in_warehouse robot box3 screw warehouse) [0]
21: (        put_up_box_from_loc robot box3 warehouse) [0]
22: (           move_to_loc robot warehouse location1) [0]
23: (           move_to_loc robot location1 location2) [0]
24: (           enter_ws robot location2 workstation3) [0]
25: (      put_down_box_in_ws robot box3 workstation3) [0]
26: (   empty_box_in_ws robot box3 workstation3 screw) [0]

time spent:       0.03 seconds parsing
                  0.05 seconds encoding
                  0.08 seconds searching
                  0.16 seconds total time
```

```
memory used:      0.52 MBytes for problem representation
                  0.00 MBytes for searching
                  0.52 MBytes total
```

Il plan trovato da **A*** con euristica *Fast-Forward*, è il seguente:

```
00: ( fill_box_in_warehouse robot box1 bolt warehouse) [0]
01: (        put_up_box_from_loc robot box1 warehouse) [0]
02: (           move_to_loc robot warehouse location1) [0]
03: (           move_to_loc robot location1 location2) [0]
04: (           enter_ws robot location2 workstation2) [0]
05: (      put_down_box_in_ws robot box1 workstation2) [0]
06: (    empty_box_in_ws robot box1 workstation2 bolt) [0]
07: (            exit_ws robot location2 workstation2) [0]
08: (           move_to_loc robot location2 location1) [0]
09: (           move_to_loc robot location1 warehouse) [0]
10: ( fill_box_in_warehouse robot box2 bolt warehouse) [0]
11: (fill_box_in_warehouse robot box3 screw warehouse) [0]
12: (        put_up_box_from_loc robot box2 warehouse) [0]
13: (           move_to_loc robot warehouse location1) [0]
14: (           move_to_loc robot location1 location2) [0]
15: (           enter_ws robot location2 workstation3) [0]
16: (      put_down_box_in_ws robot box2 workstation3) [0]
17: (    empty_box_in_ws robot box2 workstation3 bolt) [0]
18: (            exit_ws robot location2 workstation3) [0]
19: (           move_to_loc robot location2 location1) [0]
20: (           move_to_loc robot location1 warehouse) [0]
21: (        put_up_box_from_loc robot box3 warehouse) [0]
22: (           move_to_loc robot warehouse location1) [0]
23: (           move_to_loc robot location1 location2) [0]
24: (           enter_ws robot location2 workstation3) [0]
25: (      put_down_box_in_ws robot box3 workstation3) [0]
26: (   empty_box_in_ws robot box3 workstation3 screw) [0]

time spent:       0.03 seconds parsing
```

```
                 0.06 seconds encoding
                 1.88 seconds searching
                 1.97 seconds total time

memory used:      0.52 MBytes for problem representation
                 15.36 MBytes for searching
                 15.88 MBytes total
```

Il plan individuato da **A*** con euristica *Ajusted Sum* è il seguente:

```
00: (fill_box_in_warehouse robot box2 screw warehouse) [0]
01: ( fill_box_in_warehouse robot box3 bolt warehouse) [0]
02: ( fill_box_in_warehouse robot box1 bolt warehouse) [0]
03: (        put_up_box_from_loc robot box3 warehouse) [0]
04: (           move_to_loc robot warehouse location1) [0]
05: (           move_to_loc robot location1 location2) [0]
06: (           enter_ws robot location2 workstation3) [0]
07: (      put_down_box_in_ws robot box3 workstation3) [0]
08: (    empty_box_in_ws robot box3 workstation3 bolt) [0]
09: (            exit_ws robot location2 workstation3) [0]
10: (           move_to_loc robot location2 location1) [0]
11: (           move_to_loc robot location1 warehouse) [0]
12: (        put_up_box_from_loc robot box2 warehouse) [0]
13: (           move_to_loc robot warehouse location1) [0]
14: (           move_to_loc robot location1 location2) [0]
15: (           enter_ws robot location2 workstation3) [0]
16: (      put_down_box_in_ws robot box2 workstation3) [0]
17: (   empty_box_in_ws robot box2 workstation3 screw) [0]
18: (            exit_ws robot location2 workstation3) [0]
19: (           move_to_loc robot location2 location1) [0]
20: (           move_to_loc robot location1 warehouse) [0]
21: (        put_up_box_from_loc robot box1 warehouse) [0]
22: (           move_to_loc robot warehouse location1) [0]
23: (           move_to_loc robot location1 location2) [0]
24: (           enter_ws robot location2 workstation2) [0]
```

```
25: (      put_down_box_in_ws robot box1 workstation2) [0]
26: (    empty_box_in_ws robot box1 workstation2 bolt) [0]

time spent:       0.03 seconds parsing
                  0.07 seconds encoding
                  1.69 seconds searching
                  1.79 seconds total time

memory used:      0.52 MBytes for problem representation
                 13.69 MBytes for searching
                 14.21 MBytes total
```

Il plan individuato con **A*** e l'euristica costruita ad hoc è il seguente:

```
00: ( fill_box_in_warehouse robot box1 bolt warehouse) [0]
01: (        put_up_box_from_loc robot box1 warehouse) [0]
02: (           move_to_loc robot warehouse location1) [0]
03: (           move_to_loc robot location1 location2) [0]
04: (           enter_ws robot location2 workstation3) [0]
05: (      put_down_box_in_ws robot box1 workstation3) [0]
06: (    empty_box_in_ws robot box1 workstation3 bolt) [0]
07: (            exit_ws robot location2 workstation3) [0]
08: (           move_to_loc robot location2 location1) [0]
09: (           move_to_loc robot location1 warehouse) [0]
10: ( fill_box_in_warehouse robot box3 bolt warehouse) [0]
11: (        put_up_box_from_loc robot box3 warehouse) [0]
12: (           move_to_loc robot warehouse location1) [0]
13: (           move_to_loc robot location1 location2) [0]
14: (           enter_ws robot location2 workstation2) [0]
15: (      put_down_box_in_ws robot box3 workstation2) [0]
16: (    empty_box_in_ws robot box3 workstation2 bolt) [0]
17: (            exit_ws robot location2 workstation2) [0]
18: (           move_to_loc robot location2 location1) [0]
19: (           move_to_loc robot location1 warehouse) [0]
20: (fill_box_in_warehouse robot box2 screw warehouse) [0]
```

```
21: (        put_up_box_from_loc robot box2 warehouse) [0]
22: (           move_to_loc robot warehouse location1) [0]
23: (           move_to_loc robot location1 location2) [0]
24: (           enter_ws robot location2 workstation3) [0]
25: (      put_down_box_in_ws robot box2 workstation3) [0]
26: (   empty_box_in_ws robot box2 workstation3 screw) [0]

time spent:       0.03 seconds parsing
                  0.06 seconds encoding
                  4.67 seconds searching
                  4.76 seconds total time

memory used:      0.52 MBytes for problem representation
                 31.90 MBytes for searching
                 32.41 MBytes total
```

Algoritmo	N°Azioni	Tempo Totale	Spazio Totale
FF	26	0.16 s	0.52 MBytes
A*+FF	26	1.97 s	15.88 MBytes
A*+AS	26	1.79 s	14.21 MBytes
A*+MSH	26	4.76 s	32.41 MBytes

Tabella 2.1: Tabella riassuntiva risultati ottenuti

2.6 Istance 2

Il plan individuato per l'istanza 2 precedentemente illustrata, dall'algoritmo **Fast-Forward** è il seguente:

```
00: ( fill_box_in_warehouse robot box1 bolt warehouse carrier) [0]
01: (  load_boxes_from_loc robot box1 warehouse carrier slot1) [0]
02: (fill_box_in_warehouse robot box2 screw warehouse carrier) [0]
03: (  load_boxes_from_loc robot box2 warehouse carrier slot2) [0]
04: (           move_to_loc robot warehouse location1 carrier) [0]
```

```
05: (           move_to_loc robot location1 location2 carrier) [0]
06: (           enter_ws robot location2 workstation3 carrier) [0]
07: (put_down_box_in_ws robot box2 workstation3 carrier slot1) [0]
08: (   empty_box_in_ws robot box2 workstation3 screw carrier) [0]
09: (put_down_box_in_ws robot box1 workstation3 carrier slot2) [0]
10: (    empty_box_in_ws robot box1 workstation3 bolt carrier) [0]
11: (            exit_ws robot location2 workstation3 carrier) [0]
12: (           move_to_loc robot location2 location1 carrier) [0]
13: (     move_to_warehouse robot location1 warehouse carrier) [0]
14: ( fill_box_in_warehouse robot box3 bolt warehouse carrier) [0]
15: (  load_boxes_from_loc robot box3 warehouse carrier slot1) [0]
16: (           move_to_loc robot warehouse location1 carrier) [0]
17: (           move_to_loc robot location1 location2 carrier) [0]
18: (           enter_ws robot location2 workstation2 carrier) [0]
19: (put_down_box_in_ws robot box3 workstation2 carrier slot1) [0]
20: (    empty_box_in_ws robot box3 workstation2 bolt carrier) [0]

time spent:       0.03 seconds parsing
                  0.06 seconds encoding
                  1.36 seconds searching
                  1.45 seconds total time

memory used:      0.78 MBytes for problem representation
                  0.00 MBytes for searching
                  0.78 MBytes total
```

Il plan individuato da **A*** con euristica *FF* è il seguente:

```
00: (fill_box_in_warehouse robot box1 screw warehouse carrier) [0]
01: ( fill_box_in_warehouse robot box2 bolt warehouse carrier) [0]
02: (  load_boxes_from_loc robot box2 warehouse carrier slot2) [0]
03: (  load_boxes_from_loc robot box1 warehouse carrier slot3) [0]
04: ( fill_box_in_warehouse robot box3 bolt warehouse carrier) [0]
05: (  load_boxes_from_loc robot box3 warehouse carrier slot1) [0]
06: (           move_to_loc robot warehouse location1 carrier) [0]
```

```
07: (           move_to_loc robot location1 location2 carrier) [0]
08: (           enter_ws robot location2 workstation3 carrier) [0]
09: (put_down_box_in_ws robot box1 workstation3 carrier slot3) [0]
10: (   empty_box_in_ws robot box1 workstation3 screw carrier) [0]
11: (put_down_box_in_ws robot box3 workstation3 carrier slot1) [0]
12: (    empty_box_in_ws robot box3 workstation3 bolt carrier) [0]
13: (            exit_ws robot location2 workstation3 carrier) [0]
14: (           enter_ws robot location2 workstation2 carrier) [0]
15: (put_down_box_in_ws robot box2 workstation2 carrier slot2) [0]
16: (    empty_box_in_ws robot box2 workstation2 bolt carrier) [0]

time spent:       0.04 seconds parsing
                  0.07 seconds encoding
                  0.44 seconds searching
                  0.54 seconds total time

memory used:      0.78 MBytes for problem representation
                  0.93 MBytes for searching
                  1.71 MBytes total
```

Il plan individuato da **A*** con euristica *Ajusted Sum* è il seguente:

```
00: ( fill_box_in_warehouse robot box2 bolt warehouse carrier) [0]
01: ( fill_box_in_warehouse robot box3 bolt warehouse carrier) [0]
02: (  load_boxes_from_loc robot box3 warehouse carrier slot2) [0]
03: (fill_box_in_warehouse robot box1 screw warehouse carrier) [0]
04: (  load_boxes_from_loc robot box2 warehouse carrier slot3) [0]
05: (  load_boxes_from_loc robot box1 warehouse carrier slot1) [0]
06: (           move_to_loc robot warehouse location1 carrier) [0]
07: (           move_to_loc robot location1 location2 carrier) [0]
08: (           enter_ws robot location2 workstation3 carrier) [0]
09: (put_down_box_in_ws robot box1 workstation3 carrier slot1) [0]
10: (put_down_box_in_ws robot box3 workstation3 carrier slot2) [0]
11: (    empty_box_in_ws robot box3 workstation3 bolt carrier) [0]
12: (   empty_box_in_ws robot box1 workstation3 screw carrier) [0]
```

```
13: (            exit_ws robot location2 workstation3 carrier) [0]
14: (          enter_ws robot location2 workstation2 carrier) [0]
15: (put_down_box_in_ws robot box2 workstation2 carrier slot3) [0]
16: (    empty_box_in_ws robot box2 workstation2 bolt carrier) [0]

time spent:       0.03 seconds parsing
                  0.06 seconds encoding
                  1.77 seconds searching
                  1.87 seconds total time

memory used:      0.78 MBytes for problem representation
                  7.08 MBytes for searching
                  7.86 MBytes total
```

Il plan individuato da **A*** con l'euristica definita nella sezione precedente è:

```
00: (fill_box_in_warehouse robot box3 screw warehouse carrier) [0]
01: (  load_boxes_from_loc robot box3 warehouse carrier slot2) [0]
02: ( fill_box_in_warehouse robot box1 bolt warehouse carrier) [0]
03: ( fill_box_in_warehouse robot box2 bolt warehouse carrier) [0]
04: (  load_boxes_from_loc robot box2 warehouse carrier slot3) [0]
05: (  load_boxes_from_loc robot box1 warehouse carrier slot1) [0]
06: (          move_to_loc robot warehouse location1 carrier) [0]
07: (          move_to_loc robot location1 location2 carrier) [0]
08: (          enter_ws robot location2 workstation3 carrier) [0]
09: (put_down_box_in_ws robot box2 workstation3 carrier slot1) [0]
10: (put_down_box_in_ws robot box3 workstation3 carrier slot3) [0]
11: (   empty_box_in_ws robot box3 workstation3 screw carrier) [0]
12: (    empty_box_in_ws robot box2 workstation3 bolt carrier) [0]
13: (            exit_ws robot location2 workstation3 carrier) [0]
14: (          enter_ws robot location2 workstation2 carrier) [0]
15: (put_down_box_in_ws robot box1 workstation2 carrier slot2) [0]
16: (    empty_box_in_ws robot box1 workstation2 bolt carrier) [0]

time spent:       0.05 seconds parsing
```

```
                  0.07 seconds encoding
                148.45 seconds searching
                148.57 seconds total time

memory used:      0.78 MBytes for problem representation
                934.51 MBytes for searching
                935.29 MBytes total
```

Algoritmo	N°Azioni	Tempo Totale	Spazio Totale
FF	20	1.45 s	0.78 MBytes
A*+FF	16	0.54 s	1.71 MBytes
A*+AS	16	1.87 s	7.86 MBytes
A*+MSH	16	148.57 s	935.29 MBytes

Tabella 2.2: Tabella riassuntiva risultati ottenuti

Possiamo notare come **A*+MSH** esplode in maniera esponenziale sia in tempo che in spazio occupato. Per ottimizzare tali quantità si potrebbero fare ricorso a tecniche di pruning dell'albero.

Capitolo 3

Temporal Planning & Robotics

3.1 Temporal Planning

In questa penultima sezione, viene rappresentata la conversione del dominio introdotto in precedenza, affinché possa generare una sequenza temporale di azioni caratterizzate da una durata. Un problema di pianificazione temporale consiste nel trovare una serie/sequenza di azioni che, oltre ad essere eseguibili causalmente (come nella pianificazione classica), siano anche pianificabili rispettando un insieme di vincoli temporali sulle durate delle azioni, lungo una linea temporale senza limiti. Un'azione durativa è una formulazione di un'azione che richiede un certo tempo per essere completata. La quantità di tempo può essere espressa come valore fisso o come valore variabile. Similmente alle azioni tradizionali, è possibile specificare degli effetti ed è inoltre possibile esprimere una condizione utilizzando il termine "condition" invece di "precondition". Questo cambiamento semantico è stato introdotto per rappresentare il fatto che un'azione durativa può avere condizioni che devono essere vere non solo all'inizio dell'azione, ma anche alla fine o per tutta la sua durata.In questo contesto, il costrutto "at start" seguito da un predicato specifica che la condizione espressa deve essere valida prima dell'inizio dell'azione, mentre il costrutto "over all" specifica che la condizione deve essere valida sia prima dell'inizio che per tutta la durata dell'azione. Similmente, tali costrutti possono essere utilizzati all'interno della specificazione degli effetti di ogni azione.
Riportiamo brevemente i cambiamenti al dominio descritto in precedenza:

```
(define (domain gestione_scarti)
  (:requirements :strips :typing :durative-actions :equality
    :numeric-fluents)
  (:types
    location box content robot workstation carrier
  )
```

Abbiamo l'aggiunta di *durative-actions* e di *fluents*, e non vi è più presente il types *slot*, in quanto in questo caso useremo delle funzioni, di seguito riportate, per esprimere il concetto di spazio nel carrello.

```
    (:functions
        (capacity ?carrier - carrier)
        (load ?carrier - carrier)
    )
```

Per quanto riguarda le azioni invece, devono subbire un leggero cambiamento, come anticipato precedentemente, nel dettaglio vediamo subito una differenza:

```
--------------------------- PRIMA----------------------------
    (:action move_to_loc
        :parameters (?r - robot ?from ?to - location ?car - carrier)
        :precondition (and
            (at_loc ?r ?from)
            (connected ?from ?to)
            (not (is_warehouse ?to))
            (joined ?r ?car)
        )
    :effect (and
        (not (at_loc ?r ?from))
        (at_loc ?r ?to)
    )
   )
---------------------------- DOPO ----------------------------
    (:durative-action move_to_loc
        :parameters (?r - robot ?from ?to - location ?car - carrier)
```

```
        :duration (= ?duration 3)
        :condition (and
            (at start(at_loc ?r ?from))
            (over all(connected ?from ?to))
            (over all(not (is_warehouse ?to)))
            (over all(joined ?r ?car)))
        :effect (and
            (at start(not (at_loc ?r ?from)))
            (at end(at_loc ?r ?to))
        )
    )
```

Come possiamo notare non abbiamo più *preconditions* ma *conditions*, ed inoltre abbiamo tre nuovi costrutti:

- *at_start*: esso seguito da un predicato, specifica che la condizione da questo espressa deve essere valida prima del penrformarsi dell'azione;
- *at_end*: esso seguito da un predicato, specifica che la condizione da questo espressa sarà valida al completamento dell'azione;
- *over all*: esso seguito da un predicato, specifica che la condizione da questo espressa deve essere valida prima del penrformarsi dell'azione e per tutta la durata della stessa;

Vengono riportate giusto per leggibilità due azioni:

```
    (:durative-action load_boxes_from_ws
        :parameters (?r - robot ?box - box ?ws - workstation ?car - carrier)
        :duration (= ?duration 2)
        :condition (and
            (at start (at_ws ?box ?ws))
            (over all (at_ws ?r ?ws))
            (over all (joined ?r ?car))
            (over all (< (load ?car) (capacity ?car))))
        :effect (and
            (at end (carrying ?car ?box))
```

```
            (at end (increase (load ?car) 1))
            (at start (not (at_ws ?box ?ws)))
    )

    (:durative-action put_down_box_in_ws
        :parameters (?r - robot ?box - box ?ws - workstation ?car - carrier)
        :duration (= ?duration 2)
        :condition (and
            (over all (at_ws ?r ?ws))
            (at start (carrying ?car ?box))
            (over all (joined ?r ?car))
        )
        :effect (and
            (at start (not (carrying ?car ?box)))
            (at end (at_ws ?box ?ws))
            (at end (decrease (load ?car) 1)))
    )
```

Come possiamo notare, la sintassi è molto simile alle azioni definite nel plannig classico, se non fosse che, in questo caso, bisogna specificare la durata dell'azione e quando si devono verificare i singoli predicati.
Per la risoluzione in questo caso del nostro problema, la cui descrizione rimane invariata se non per l'istanzazione della dimensione del carrello, di seguito riportato:

```
(= (capacity carrier) 2)  ; Capacità iniziale del carrier
(= (load carrier) 0)      ; Carico iniziale del carrier
```

si è fatto uso del framework *planutils*, scegliendo come planner di risolutore **LPG-td**, in quanto supporta la presenza di durative actions, inoltre è stato l'unico solver, in grado di risolvere il problema in tempi brevi. LPG-td, consente di specificare diverse tipologie di ricerca:

- *speed*: consente di ottenere un plan nel minor tempo possibile e lo restituisce in un file di output, specificato anch'esso nel comando di esecuzione;

- *quality*: effettua la risoluzione del problema determinando un plan utilizzando un certo quantitativo di risorse di calcolo superiore al fine di migliorare quest'ultimo.

Si è eseguito prima con option pari a *quality*, ottenendo il seguente plan:

```
/home/aiguy/.planutils/packages/lpg-td/bin/lpg-td
-o domain.pddl -f problem.pddl -v off -noout -quality
```

quello che otteniamo è il seguente plan:

```
0.0003: (FILL_BOX_IN_WAREHOUSE ROBOT BOX2 SCREW WAREHOUSE CARRIER)
[D:3.5000; C:1.0000]
3.5005: (LOAD_BOXES_FROM_LOC ROBOT BOX2 WAREHOUSE CARRIER)
[D:2.0000; C:1.0000]
0.0008: (FILL_BOX_IN_WAREHOUSE ROBOT BOX3 BOLT WAREHOUSE CARRIER)
[D:3.5000; C:1.0000]
5.5010: (LOAD_BOXES_FROM_LOC ROBOT BOX3 WAREHOUSE CARRIER)
[D:2.0000; C:1.0000]
7.5012: (MOVE_TO_LOC ROBOT WAREHOUSE LOCATION1 CARRIER)
[D:3.0000; C:1.0000]
10.5015: (MOVE_TO_LOC ROBOT LOCATION1 LOCATION2 CARRIER)
[D:3.0000; C:1.0000]
13.5017: (ENTER_WS ROBOT LOCATION2 WORKSTATION3 CARRIER)
[D:0.0000; C:1.0000]
13.5020: (PUT_DOWN_BOX_IN_WS ROBOT BOX3 WORKSTATION3 CARRIER)
[D:2.0000; C:1.0000]
15.5022: (EXIT_WS ROBOT LOCATION2 WORKSTATION3 CARRIER)
[D:0.0000; C:1.0000]
15.5025: (MOVE_TO_LOC ROBOT LOCATION2 LOCATION1 CARRIER)
[D:3.0000; C:1.0000]
18.5028: (MOVE_TO_LOC ROBOT LOCATION1 LOCATION2 CARRIER)
[D:3.0000; C:1.0000]
21.5030: (ENTER_WS ROBOT LOCATION2 WORKSTATION3 CARRIER)
[D:0.0000; C:1.0000]
21.5033: (PUT_DOWN_BOX_IN_WS ROBOT BOX2 WORKSTATION3 CARRIER)
```

```
[D:2.0000; C:1.0000]
23.5035: (EMPTY_BOX_IN_WS ROBOT BOX2 WORKSTATION3 SCREW CARRIER)
[D:3.5000; C:1.0000]
21.5037: (EMPTY_BOX_IN_WS ROBOT BOX3 WORKSTATION3 BOLT CARRIER)
[D:3.5000; C:1.0000]
25.0040: (LOAD_BOXES_FROM_WS ROBOT BOX2 WORKSTATION3 CARRIER)
[D:2.0000; C:1.0000]
27.0042: (EXIT_WS ROBOT LOCATION2 WORKSTATION3 CARRIER)
[D:0.0000; C:1.0000]
27.0045: (MOVE_TO_LOC ROBOT LOCATION2 LOCATION1 CARRIER)
[D:3.0000; C:1.0000]
30.0047: (MOVE_TO_WAREHOUSE ROBOT LOCATION1 WAREHOUSE CARRIER)
[D:3.0000; C:1.0000]
33.0050: (PUT_DOWN_BOX_IN_LOC ROBOT BOX2 WAREHOUSE CARRIER)
[D:2.0000; C:1.0000]
35.0052: (FILL_BOX_IN_WAREHOUSE ROBOT BOX2 BOLT WAREHOUSE CARRIER)
[D:3.5000; C:1.0000]
38.5055: (LOAD_BOXES_FROM_LOC ROBOT BOX2 WAREHOUSE CARRIER)
[D:2.0000; C:1.0000]
40.5057: (MOVE_TO_LOC ROBOT WAREHOUSE LOCATION1 CARRIER)
[D:3.0000; C:1.0000]
43.5060: (MOVE_TO_LOC ROBOT LOCATION1 LOCATION2 CARRIER)
[D:3.0000; C:1.0000]
46.5062: (ENTER_WS ROBOT LOCATION2 WORKSTATION2 CARRIER)
[D:0.0000; C:1.0000]
46.5065: (PUT_DOWN_BOX_IN_WS ROBOT BOX2 WORKSTATION2 CARRIER)
[D:2.0000; C:1.0000]
48.5067: (EMPTY_BOX_IN_WS ROBOT BOX2 WORKSTATION2 BOLT CARRIER)
[D:3.5000; C:1.0000]
```

L'output prodotto sembra dunque un plan di qualità accettabile infatti la sequenza di azioni proposta risulta infatti ragionevolmente ottimizza, si può notare nelle prime azioni come l'agente robotico consumi tutto lo spazio disponibile sul carrello. Successivamente si è passati a questa riga di codice,

utilizzando un option pari a *speedy*:

```
/home/aiguy/.planutils/packages/lpg-td/bin/lpg-td
-o domain.pddl -f problem.pddl -noout -seed 41 -n 3
```

Producendo questo output:

```
Plan computed:
   Time: (ACTION) [action Duration; action Cost]
 0.0000: (FILL_BOX_IN_WAREHOUSE ROBOT BOX3 BOLT WAREHOUSE CARRIER)
 [D:3.50; C:1.00]
 3.5000: (LOAD_BOXES_FROM_LOC ROBOT BOX3 WAREHOUSE CARRIER)
 [D:2.00; C:1.00]
 5.5000: (MOVE_TO_LOC ROBOT WAREHOUSE LOCATION1 CARRIER)
 [D:3.00; C:1.00]
 8.5000: (MOVE_TO_LOC ROBOT LOCATION1 LOCATION2 CARRIER)
 [D:3.00; C:1.00]
 11.5000: (ENTER_WS ROBOT LOCATION2 WORKSTATION2 CARRIER)
 [D:0.00; C:1.00]
 11.5000: (PUT_DOWN_BOX_IN_WS ROBOT BOX3 WORKSTATION2 CARRIER)
 [D:2.00; C:1.00]
 13.5000: (EMPTY_BOX_IN_WS ROBOT BOX3 WORKSTATION2 BOLT CARRIER)
 [D:3.50; C:1.00]
 15.0000: (LOAD_BOXES_FROM_WS ROBOT BOX3 WORKSTATION2 CARRIER)
 [D:2.00; C:1.00]
 17.0000: (EXIT_WS ROBOT LOCATION2 WORKSTATION2 CARRIER)
 [D:0.00; C:1.00]
 17.0000: (MOVE_TO_LOC ROBOT LOCATION2 LOCATION1 CARRIER)
 [D:3.00; C:1.00]
 20.0000: (MOVE_TO_WAREHOUSE ROBOT LOCATION1 WAREHOUSE CARRIER)
 [D:3.00; C:1.00]
 23.0000: (PUT_DOWN_BOX_IN_LOC ROBOT BOX3 WAREHOUSE CARRIER)
 [D:2.00; C:1.00]
 23.0000: (FILL_BOX_IN_WAREHOUSE ROBOT BOX2 BOLT WAREHOUSE CARRIER)
 [D:3.50; C:1.00]
```

```
 25.0000: (FILL_BOX_IN_WAREHOUSE ROBOT BOX3 SCREW WAREHOUSE CARRIER)
 [D:3.50; C:1.00]
 26.5000: (LOAD_BOXES_FROM_LOC ROBOT BOX2 WAREHOUSE CARRIER)
 [D:2.00; C:1.00]
 28.5000: (LOAD_BOXES_FROM_LOC ROBOT BOX3 WAREHOUSE CARRIER)
 [D:2.00; C:1.00]
 30.5000: (MOVE_TO_LOC ROBOT WAREHOUSE LOCATION1 CARRIER)
 [D:3.00; C:1.00]
 33.5000: (MOVE_TO_LOC ROBOT LOCATION1 LOCATION2 CARRIER)
 [D:3.00; C:1.00]
 36.5000: (ENTER_WS ROBOT LOCATION2 WORKSTATION3 CARRIER)
 [D:0.00; C:1.00]
 36.5000: (PUT_DOWN_BOX_IN_WS ROBOT BOX3 WORKSTATION3 CARRIER)
 [D:2.00; C:1.00]
 36.5000: (PUT_DOWN_BOX_IN_WS ROBOT BOX2 WORKSTATION3 CARRIER)
 [D:2.00; C:1.00]
 38.5000: (EMPTY_BOX_IN_WS ROBOT BOX2 WORKSTATION3 BOLT CARRIER)
 [D:3.50; C:1.00]
 38.5000: (EMPTY_BOX_IN_WS ROBOT BOX3 WORKSTATION3 SCREW CARRIER)
 [D:3.50; C:1.00]

Solution number: 3
Total time:      8.94
Search time:     8.94
Actions:         23
Duration:        42.000
Plan quality:    23.000
Total Num Flips: 14353
```

3.2 Robotic Planning

L'ultima sezione è dedicata ad una pianificazione robotica si occupa della progettazione e dello sviluppo di algoritmi e strategie che permettono a robot

autonomi di pianificare e programmare le loro azioni per raggiungere obiettivi specifici in ambienti dinamici e incerti. Per pianificare le azioni, il robot deve avere una rappresentazione dell'ambiente in cui opera e deve essere in grado di elaborare un piano che lo conduca al raggiungimento degli obiettivi prefissati. Per affrontare il secondo punto del terzo task richiesto, è necessario di implementare il problema descritto nella sezione precedente utilizzando il ROS2 Planning System (Plansys2), una piattaforma open source basata su ROS (Robot Operating System 2).
Plansys2 offre strumenti per la pianificazione, l'esecuzione e la gestione delle attività per robot autonomi e supporta la pianificazione temporale.
All'interno di Plansys2, vi troviamo le "fake actions" sono azioni simulate o rappresentate in modo tale da consentire al sistema di pianificazione, di gestire situazioni specifiche o di modellare comportamenti desiderati che non necessariamente corrispondono ai comportamenti reali del robot o dell'agente. Queste azioni sono utilizzate per rappresentare situazioni particolari, come ad esempio il fatto che il robot "rimanga in attesa" in un determinato stato o posizione per un certo periodo di tempo, e possono essere impiegate per gestire le tempistiche, includendo pause e/oritardi.
Le fake action sono state definite in linguaggio C++ e riproducono le azioni durative definite nel dominio. Viene riportato brevemente il codice di una sola fake-action, corrispondente ad :

```
#include <memory>
#include <algorithm>
#include<vector>
#include<string>

#include "plansys2_executor/ActionExecutorClient.hpp"
#include "rclcpp/rclcpp.hpp"
#include "rclcpp_action/rclcpp_action.hpp"

using namespace std::chrono_literals;
class FillBoxInWarehouse : public
    plansys2::ActionExecutorClient {
```

```
public:
  FillBoxInWarehouse()
  : plansys2::ActionExecutorClient("fillboxinwarehouse", 250ms)
  {
    progress_ = 0.0;
  }

private:
  void do_work()
  { std :: vector <std :: string > arguments = get_arguments () ;
    if (progress_ < 1.0) {
      progress_ += 0.2;
      send_feedback(progress_, "Robot "+arguments [0]+ " is filling "+
            arguments [1]+ " in "+ arguments [3] + " with some "+
            arguments [2]+ " carring a "+
            arguments [4]);
    } else {
      finish(true, progress_, "Robot "+arguments [0]+ " is filling "+
            arguments [1]+ " in "+ arguments [3] + " with some "+
            arguments [2]+ " carring a "+

            arguments [4]);
      progress_ = 0.0;
      std::cout << std::endl;
    }

    std::cout << "\r\e[K" << std::flush;
    std::cout << "Robot "+arguments [0]+ " is filling "+
            arguments [1]+ " in "+ arguments [3] + " with some "+
            arguments [2]+ " carring a "+
            arguments [4]+" . . .
            [ " << std::min(100.0, progress_ * 100.0) << "% ]  " <<
```

```
            std::flush;
  }
  float progress_;
};

int main(int argc, char ** argv)
{
  rclcpp::init(argc, argv);
  auto node = std::make_shared<FillBoxInWarehouse>();

  node->set_parameter(rclcpp::
  Parameter("action_name", "FILLBOXINWAREHOUSE"));
  node->trigger_transition(lifecycle_msgs::
  msg::Transition::TRANSITION_CONFIGURE);

  rclcpp::spin(node->get_node_base_interface());
  rclcpp::shutdown();
  return 0;
}
```

Il task seguente ha richiesto la definizione di un file Python di launch, di seguito proposto:

```
import os

from ament_index_python.packages import get_package_share_directory
from launch import LaunchDescription
from launch.actions import DeclareLaunchArgument, IncludeLaunchDescription
from launch.launch_description_sources import PythonLaunchDescriptionSource
from launch.substitutions import LaunchConfiguration
from launch_ros.actions import Node

def generate_launch_description():
```

```
# Get the launch directory
example_dir = get_package_share_directory('manufactoring_service_robots')
namespace = LaunchConfiguration('namespace')

declare_namespace_cmd = DeclareLaunchArgument(
    'namespace',
    default_value='',
    description='Namespace')

plansys2_cmd = IncludeLaunchDescription(
    PythonLaunchDescriptionSource(os.path.join(
        get_package_share_directory('plansys2_bringup'),
        'launch',
        'plansys2_bringup_launch_monolithic.py')),
    launch_arguments={
      'model_file': example_dir + '/pddl/domain.pddl',
      'namespace': namespace
      }.items())

# Specify the actions
empty_box_in_loc_cmd = Node(
    package='manufactoring_service_robots',
    executable='empty_box_in_loc_action_node',
    name='empty_box_in_loc_action_node',
    namespace=namespace,
    output='screen',
    parameters=[])

empty_box_in_ws_cmd = Node(
    package='manufactoring_service_robots',
    executable='empty_box_in_ws_action_node',
    name='empty_box_in_ws_action_node',
    namespace=namespace,
```

```
        output='screen',
        parameters=[])

    enter_ws_cmd = Node(
        package='manufactoring_service_robots',
        executable='enter_ws_action_node',
        name='enter_ws_action_node',
        namespace=namespace,
        output='screen',
        parameters=[])   # Create the launch description and populate

    exit_ws_cmd = Node(
        package='manufactoring_service_robots',
        executable='exit_ws_action_node',
        name='exit_ws_action_node',
        namespace=namespace,
        output='screen',
        parameters=[])

    fill_box_in_loc_cmd = Node(
        package='manufactoring_service_robots',
        executable='fill_box_in_loc_action_node',
        name='fill_box_in_loc_action_node',
        namespace=namespace,
        output='screen',
        parameters=[])

    fill_box_in_warehouse_cmd = Node(
        package='manufactoring_service_robots',
        executable='fill_box_in_warehouse_action_node',
        name='fill_box_in_warehouse_action_node',
        namespace=namespace,
        output='screen',
```

```
        parameters=[])

    fill_box_in_ws_cmd = Node(
        package='manufactoring_service_robots',
        executable='fill_box_in_ws_action_node',
        name='fill_box_in_ws_action_node',
        namespace=namespace,
        output='screen',
        parameters=[])

    load_boxes_from_loc_cmd = Node(
        package='manufactoring_service_robots',
        executable='load_boxes_from_loc_action_node',
        name='load_boxes_from_loc_action_node',
        namespace=namespace,
        output='screen',
        parameters=[])

    load_boxes_from_ws_cmd = Node(
        package='manufactoring_service_robots',
        executable='load_boxes_from_ws_action_node',
        name='load_boxes_from_ws_action_node',
        namespace=namespace,
        output='screen',
        parameters=[])

    move_to_loc_cmd = Node(
        package='manufactoring_service_robots',
        executable='move_to_loc_action_node',
        name='move_to_loc_action_node',
        namespace=namespace,
        output='screen',
        parameters=[])
```

```
move_to_warehouse_cmd = Node(
    package='manufactoring_service_robots',
    executable='move_to_warehouse_action_node',
    name='move_to_warehouse_action_node',
    namespace=namespace,
    output='screen',
    parameters=[])


put_down_box_in_loc_cmd = Node(
    package='manufactoring_service_robots',
    executable='put_down_box_in_loc_action_node',
    name='put_down_box_in_loc_action_node',
    namespace=namespace,
    output='screen',
    parameters=[])


put_down_box_in_ws_cmd = Node(
    package='manufactoring_service_robots',
    executable='put_down_box_in_ws_action_node',
    name='put_down_box_in_ws_action_node',
    namespace=namespace,
    output='screen',
    parameters=[])

ld = LaunchDescription()

ld.add_action(declare_namespace_cmd)

# Declare the launch options
```

```
    ld.add_action(plansys2_cmd)

    ld.add_action(empty_box_in_loc_cmd)
    ld.add_action(empty_box_in_ws_cmd)
    ld.add_action(enter_ws_cmd)
    ld.add_action(exit_ws_cmd)
    ld.add_action(fill_box_in_loc_cmd)
    ld.add_action(fill_box_in_warehouse_cmd)
    ld.add_action(fill_box_in_ws_cmd)
    ld.add_action(load_boxes_from_loc_cmd)
    ld.add_action(load_boxes_from_ws_cmd)
    ld.add_action(move_to_loc_cmd)
    ld.add_action(move_to_warehouse_cmd)
    ld.add_action(put_down_box_in_loc_cmd)
    ld.add_action(put_down_box_in_ws_cmd)

    return ld
```

A questo punto si è proceduto a descrivere l'istanza del problema da risolvere nel file *commands* facendo riferimento al problem file precedentemente realizzato:

```
set instance ROBOT robot
set instance CARRIER carrier
set instance WAREHOUSE location
set instance LOCATION1 location
set instance LOCATION2 location

set instance WORKSTATION1 workstation
set instance WORKSTATION2 workstation
set instance WORKSTATION3 workstation

set instance BOX1 box
set instance BOX2 box
set instance BOX3 box
```

```
set instance BOLT content
set instance SCREW content

set function (= (capacity CARRIER) 2)
set function (= (load CARRIER) 0)

set predicate (atloc ROBOT WAREHOUSE)

set predicate (iswarehouse WAREHOUSE)

set predicate (atloc BOX1 WAREHOUSE)
set predicate (atloc BOX2 WAREHOUSE)
set predicate (atloc BOX3 WAREHOUSE)

set predicate (atloc BOLT WAREHOUSE)
set predicate (atloc SCREW WAREHOUSE)

set predicate (joined ROBOT CARRIER)

set predicate (isempty BOX1)
set predicate (isempty BOX2)
set predicate (isempty BOX3)

set predicate (atloc WORKSTATION1 LOCATION1)
set predicate (atloc WORKSTATION2 LOCATION2)
set predicate (atloc WORKSTATION3 LOCATION2)

set predicate (connected WAREHOUSE LOCATION1)
set predicate (connected LOCATION1 LOCATION2)
set predicate (connected LOCATION1 WAREHOUSE)
set predicate (connected LOCATION2 LOCATION1)
```

Come ultima operazione si è andati a configurare il file **CMakeList** come

segue:

```
cmake_minimum_required(VERSION 3.5)
project(manufactoring_service_robots)
find_package(ament_cmake REQUIRED)
find_package(rclcpp REQUIRED)
find_package(rclcpp_action REQUIRED)
find_package(plansys2_msgs REQUIRED)
find_package(plansys2_executor REQUIRED)
set(CMAKE_CXX_STANDARD 17)

set(dependencies
    rclcpp
    rclcpp_action
    plansys2_msgs
    plansys2_executor
)

add_executable(empty_box_in_loc_action_node src/empty_box_in_loc_action_node.cpp)
ament_target_dependencies(empty_box_in_loc_action_node ${dependencies})

add_executable(empty_box_in_ws_action_node src/empty_box_in_ws_action_node.cpp)
ament_target_dependencies(empty_box_in_ws_action_node ${dependencies})

add_executable(enter_ws_action_node src/enter_ws_action_node.cpp)
ament_target_dependencies(enter_ws_action_node ${dependencies})

add_executable(exit_ws_action_node src/exit_ws_action_node.cpp)
ament_target_dependencies(exit_ws_action_node ${dependencies})

add_executable(fill_box_in_loc_action_node src/fill_box_in_loc_action_node.cpp)
ament_target_dependencies(fill_box_in_loc_action_node ${dependencies})

add_executable(fill_box_in_warehouse_action_node src/fill_box_in_warehouse_action
```

```
ament_target_dependencies(fill_box_in_warehouse_action_node ${dependencies})

add_executable(fill_box_in_ws_action_node src/fill_box_in_ws_action_node.cpp)
ament_target_dependencies(fill_box_in_ws_action_node ${dependencies})

add_executable(load_boxes_from_loc_action_node src/load_boxes_from_loc_action_nod
ament_target_dependencies(load_boxes_from_loc_action_node ${dependencies})

add_executable(load_boxes_from_ws_action_node src/load_boxes_from_ws_action_node.
ament_target_dependencies(load_boxes_from_ws_action_node ${dependencies})

add_executable(move_to_loc_action_node src/move_to_loc_action_node.cpp)
ament_target_dependencies(move_to_loc_action_node ${dependencies})

add_executable(move_to_warehouse_action_node src/move_to_warehouse_action_node.cp
ament_target_dependencies(move_to_warehouse_action_node ${dependencies})

add_executable(put_down_box_in_loc_action_node src/put_down_box_in_loc_action_nod
ament_target_dependencies(put_down_box_in_loc_action_node ${dependencies})

add_executable(put_down_box_in_ws_action_node src/put_down_box_in_ws_action_node.
ament_target_dependencies(put_down_box_in_ws_action_node ${dependencies})

install(DIRECTORY launch pddl DESTINATION share/${PROJECT_NAME})

install(TARGETS
  empty_box_in_loc_action_node
  empty_box_in_ws_action_node
  enter_ws_action_node
  exit_ws_action_node
  fill_box_in_loc_action_node
  fill_box_in_warehouse_action_node
  fill_box_in_ws_action_node
```

```
  load_boxes_from_loc_action_node
  load_boxes_from_ws_action_node
  move_to_loc_action_node
  move_to_warehouse_action_node
  put_down_box_in_loc_action_node
  put_down_box_in_ws_action_node
  ARCHIVE DESTINATION lib
  LIBRARY DESTINATION lib
  RUNTIME DESTINATION lib/${PROJECT_NAME}
)

if(BUILD_TESTING)
  find_package(ament_lint_auto REQUIRED)
  ament_lint_auto_find_test_dependencies()

  find_package(ament_cmake_gtest REQUIRED)
endif()

ament_export_dependencies(${dependencies})

ament_package()
```

Lanciando il sistema realizzato con il plan precedentemente generato, otteniamo il seguente risultato:

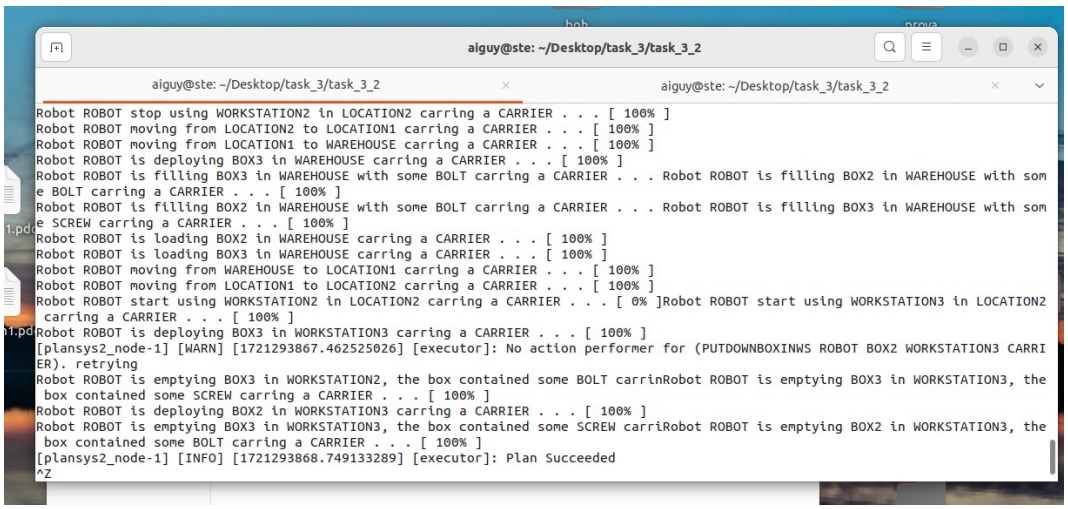

Figura 3.1: Contenuto terminale 1

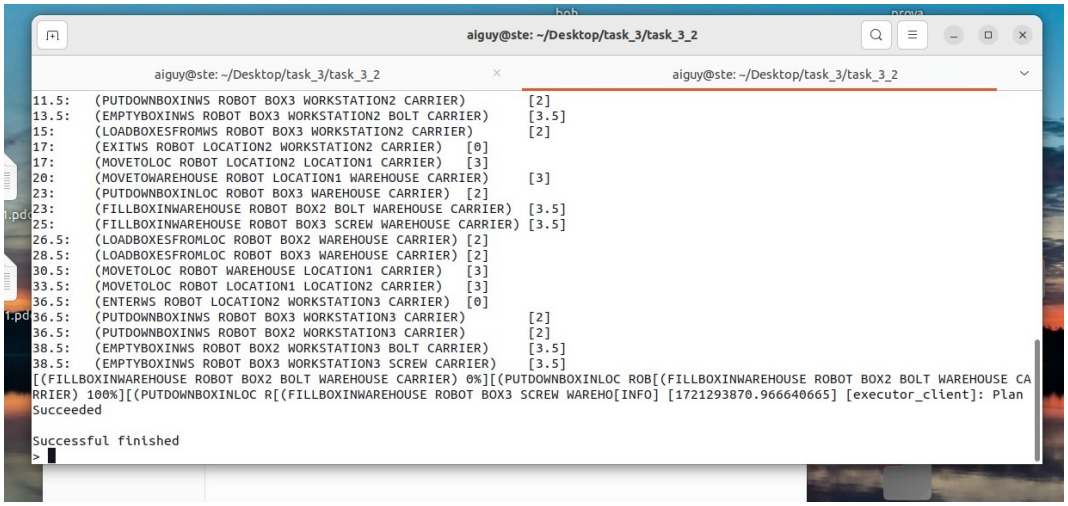

Figura 3.2: Contenuto terminale 2

Per poter eseguire tale programma si devono seguire i seguenti passi:

- *Primo Terminale*:
  - Compilare i file C++ presenti nella cartella src:

    ```
    colcon build --symlink-install
    ```

  - Eseguire il comando:

    ```
    source install/setup.bash
    ```

  - Lanciare Ros:

```
ros2 launch manufactoring_service_robots plansys2_msl.py
```

- *Secondo Terminale*:
  - Avviare il terminale di PlanSys2 con il comando:

    ```
    ros2 run plansys2_terminal plansys2_terminal
    ```

  - Dare in input il file **commands** tramite il comando:

    ```
    source ./launch/commands
    ```

  - Eseguire il plan con il comando:

    ```
    run plan-file <percorso assoluto al plan.txt>
    ```

Capitolo 4

Organizzazione dell'archivio consegnato

L'archivio è così organizzato:

- Cartella *task 1*: All'interno della cartella è presente il file PDDL del dominio sviluppato seguendo i punti richiesti dalla traccia;
- Cartella *task 2*: In questa cartella è presente uno script *run.sh* che permette di avviare la risoluzione dei problemi (*./eseguibile.sh*). Inoltre sono presenti le seguenti sotto cartelle:
  - **src**: contiene i file Java riguardanti l'implementazione dell'euristica e degli algoritmi di ricerca;
  - **pddl**: contiene i file, in formato .pddl, riguardanti la modellazione del dominio e delle istanze richieste;
  - **lib**: contiene la libreria PDDL4J in formato .jar;
- *task 3*: a sua volta è suddivisa in due sottocartelle:
  - *31*: Contiene i file PDDL del dominio e dell'istanza 2 opportunamente modificati per il Temporal Planning. Inoltre è presente il piano generato da lpg-td;
  - *32*: Contiene il contenuto del problema 3.2 espresso nella traccia;