



UNIVERSITÀ DELLA CALABRIA

DIPARTIMENTO DI  
INGEGNERIA INFORMATICA,  
MODELLISTICA, ELETTRONICA  
E SISTEMISTICA

DIMES

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA

INTELLIGENZA ARTIFICIALE

## Relazione Elaborato Finale

*Industrial Manufacturing Service Planning System*

Professori

Prof. Francesco Scarcello

Prof. Antonio Bono

Studenti

Francesco Curcio

Matr. 252371

Lorena Gullone

Matr. 252455

---

Anno Accademico 2023-2024

# Indice

<b>1 Modelling</b>	<b>3</b>
1.1 Domain . . . . .	3
1.1.1 Types . . . . .	4
1.1.2 Predicates . . . . .	5
1.1.3 Actions . . . . .	6
1.2 Problem 1 - Istanza semplificata . . . . .	11
1.2.1 Risultati . . . . .	13
1.3 Problem 2 - Istanza complessa . . . . .	14
1.3.1 Risultati . . . . .	16
<b>2 Classical Planning</b>	<b>18</b>
2.1 Planner personalizzato ed euristica personalizzata . . . . .	18
2.1.1 Planner A* Search . . . . .	18
2.1.2 Euristiche personalizzate . . . . .	30
2.2 Istanza 1 . . . . .	33
2.2.1 Problema . . . . .	33
2.2.2 Risultati . . . . .	35
2.3 Istanza 2 . . . . .	44
2.3.1 Domain - Types . . . . .	44
2.3.2 Domain - Predicates . . . . .	44
2.3.3 Domain - Actions . . . . .	46
2.3.4 Problema - Versione semplificata . . . . .	53
2.3.5 Risultati - Versione semplificata . . . . .	56
2.3.6 Problema - Versione complessa . . . . .	62
2.3.7 Risultati - Versione complessa . . . . .	64
<b>3 Temporal Planning &amp; Robotics</b>	<b>70</b>
3.1 Temporal Planning . . . . .	70
3.1.1 Problema . . . . .	73
3.1.2 Risultati . . . . .	73
3.2 Robot Planning . . . . .	76
<b>4 Organizzazione dell'archivio di lavoro</b>	<b>86</b>

# Introduzione e obiettivo del progetto

Il progetto si colloca all'interno del contesto della pianificazione automatizzata, con specifico riferimento a scenari industriali in cui è necessaria la gestione efficiente dei flussi logistici tra stazioni di lavoro attraverso l'impiego di agenti robotici autonomi.

L'obiettivo principale di questo progetto è la modellazione, implementazione e risoluzione di problemi di pianificazione utilizzando il Planning Domain Definition Language (PDDL). In particolare, il progetto si propone di affrontare i seguenti aspetti:

1. **Modellazione del dominio e dei problemi:** Utilizzare PDDL per rappresentare un sistema industriale complesso in cui stazioni di lavoro, dislocate in diverse aree, necessitano di materiali specifici per svolgere le loro operazioni. Questi materiali devono essere consegnati da agenti robotici, che operano in un ambiente connesso, seguendo percorsi predeterminati. La modellazione deve essere flessibile e adattabile, permettendo l'introduzione di nuovi tipi di contenuti nei materiali e l'espansione del sistema per l'integrazione di più agenti robotici in futuro.
2. **Implementazione di algoritmi di ricerca e ottimizzazione:** Sviluppare e testare algoritmi di ricerca, sia classici che personalizzati, per risolvere le istanze di pianificazione definite. Gli algoritmi devono essere valutati in termini di efficienza computazionale, considerando parametri come il tempo di esecuzione, il numero di stati esplorati e l'uso della memoria. Per questa fase, si ha la possibilità di utilizzare librerie esistenti, come PDDL4J, o di sviluppare soluzioni ex novo.
3. **Integrazione di modelli temporali e implementazione robotica:** Estendere la modellazione del dominio per includere azioni durative e pianificazione temporale. Questo aspetto richiede l'integrazione con architetture software come PlanSys2, utilizzando planner temporali per garantire che le operazioni siano eseguite in modo sincrono ed efficiente nel tempo.

Il risultato atteso di questo progetto è una soluzione di pianificazione che non solo risolve le problematiche attuali, ma che sia anche in grado di dimostrare come l'uso congiunto di PDDL e algoritmi di pianificazione avanzati possa essere integrato in sistemi robotici reali, contribuendo all'automazione e all'ottimizzazione dei processi industriali.

# 1 Modelling

La fase di modellazione rappresenta il cuore del processo di pianificazione, in cui viene definita la struttura formale che descrive il dominio e i problemi da risolvere. Utilizzando il **Planning Domain Definition Language** (PDDL), questa fase permette di specificare le entità, le relazioni e le dinamiche che caratterizzano il sistema industriale oggetto del progetto.

Nell'attuale sezione della relazione vengono spiegati i file principali da considerare in questo contesto: il **Domain.pddl** che descrive le regole generali del dominio, specificando quali sono gli elementi del mondo, le relazioni tra essi, e le azioni possibili; e il **Problem.pddl** che al contrario contiene la descrizione di uno scenario specifico all'interno del dominio modellato.

Questa struttura di modellazione permette di separare la definizione delle regole generali del sistema dalla specificazione dei problemi concreti da risolvere, garantendo una maggiore flessibilità e riusabilità del modello. Nelle sezioni seguenti, verranno approfonditi i dettagli tecnici e concettuali relativi alla creazione e all'implementazione di ciascun componente.

## 1.1 Domain

In questo primo file solitamente sono presenti 3 sezioni:

1. **Types**: contenente la dichiarazione dei diversi tipi di oggetti presenti nel dominio.
2. **Predicates**: contenente le proprietà e le relazioni tra gli oggetti nel dominio. Ogni predicato è una funzione che può essere vera o falsa in un dato stato del mondo.
3. **Actions**: forse la sezione più importante, in cui vengono descritte le azioni che possono essere eseguite nel dominio.

### 1.1.1 Types

I tipi definiti per questo problema sono i seguenti:

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

1  
2  
3  
4  
5  
6

- **Location:** Il tipo **location** rappresenta un punto o un'area specifica all'interno del dominio industriale modellato in PDDL. Le varie location possono includere il magazzino centrale, le diverse workstation e altre zone rilevanti per la consegna dei materiali. Questo tipo è utilizzato per identificare la posizione degli agenti robotici, delle scatole e delle workstation all'interno del dominio.
- **Box:** Il tipo **box** rappresenta una scatola utilizzata per contenere e trasportare materiali. Ogni scatola può essere riempita con un determinato **content**, come valvole, bulloni o attrezzi, che nel nostro caso vengono modellati in modo astratto attraverso il tipo **content** oppure essere vuota.
- **Content:** Il tipo **content** rappresenta i vari materiali che possono essere contenuti nelle **box**.
- **Robot:** Il tipo **robot** rappresenta un agente robotico incaricato di eseguire attività di trasporto e consegna dei materiali. I robot possono spostarsi tra le **location**, riempire e svuotare le **box**, e consegnare i materiali alle **workstation**.
- **Workstation:** Il tipo **workstation** rappresenta una stazione di lavoro specifica all'interno dell'ambiente di produzione industriale; ciascuna workstation viene collocata e afferisce ad una specifica **location**.

### 1.1.2 Predicates

I predicati che invece rappresentano le proprietà e le relazioni tra gli oggetti nel dominio sono:

```
(:predicates
  (at_loc ?obj - (either robot workstation box content) ?loc      1
         - location)
  (at_ws ?obj - (either robot box content) ?ws - workstation)   2
  (is_empty ?box - box)
  (filled ?box - box ?cont - content)                                3
  (carrying ?robot - robot ?box - box)                               4
  (free ?robot - robot)                                              5
  (connected ?loc1 ?loc2 - location)                                 6
  (is_warehouse ?loc - location)                                    7
)
)                                                                           8
)                                                                           9
)                                                                           10
```

- **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 trova in una certa posizione (**?loc**);
- **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**);
- **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**);
- **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.

Sebbene in generale una workstation sia situata in una determinata location, i predicati **at\_loc** e **at\_ws** sono mutuamente esclusivi. Per accedere a una workstation, l'agente robotico deve trovarsi prima nella location che la contiene; tuttavia, una volta che l'agente robotico si trova presso la workstation, nella base di conoscenza non sarà più considerato presente nella location, ma esclusivamente presso la workstation.

### 1.1.3 Actions

Come accennato le azioni permettono agli agenti robotici di interagire con l'ambiente circostante e di svolgere le operazioni necessarie per raggiungere l'obiettivo prefissato. Un'azione è composta da **parametri** (gli oggetti richiesti per il raggiungimento dell'obiettivo), **precondizioni** (le condizioni che devono essere soddisfatte prima dell'esecuzione dell'azione) ed **effetti** (le conseguenze dell'azione ovvero ciò che risulta vero dopo l'esecuzione dell'azione).

- **move\_to\_location**: rappresenta lo spostamento dell'agente robotico, partendo da una posizione iniziale e arrivando ad una posizione adiacente.

```
(:action move_to_location
  :parameters (?r - robot ?from - location ?to - location)
  :precondition (and (at_loc ?r ?from) (connected ?from ?to))
  :effect (and (not (at_loc ?r ?from)) (at_loc ?r ?to))
)
```

1  
2  
3  
4  
5

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

- **enter\_workstation**: azione che rappresenta l'ingresso, da parte dell'agente robotico, all'intero di una workstation;

```
(:action enter_workstation
  :parameters (?r - robot ?loc - location ?ws - workstation)
  :precondition (and (at_loc ?ws ?loc) (at_loc ?r ?loc) (not (
    at_ws ?r ?ws)) )
  :effect (and (not (at_loc ?r ?loc)) (at_ws ?r ?ws))
)
```

1  
2  
3  
4  
5

- **parameters**: l'agente robotico che esegue l'azione, una location e una workstation;
- **precondition**:
  - \* la workstation e l'agente robotico devono trovarsi nella medesima location;
  - \* il robot non deve essere già dentro la workstation;
- **effects**:
  - \* l'agente robotico non si trova nella location ma si trova dentro la workstation;

- **exit\_workstation**: usata per rappresentare l'uscita del robot da una stazione di lavoro.

```
(:action exit_workstation
  :parameters (?r - robot ?loc - location ?ws - workstation)
  :precondition (and (at_loc ?ws ?loc) (not (at_loc ?r ?loc)) (
    at_ws ?r ?ws))
  :effect (and (at_loc ?r ?loc) (not (at_ws ?r ?ws)))
)
```

1  
2  
3  
4  
5

- **parameters**: l'agente robotico che esegue l'azione, una location ed una workstation;
- **precondition**:
  - \* 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\_workstation**: usata per far sì che il robot possa posare la scatola che sta trasportando all'interno di una workstation

```
(:action put_down_box_in_workstation
  :parameters (?r - robot ?box - box ?ws - workstation)
  :precondition (and (at_ws ?r ?ws) (carrying ?r ?box))
  :effect (and (not (carrying ?r ?box)) (free ?r) (at_ws ?box ?ws))
)
```

1  
2  
3  
4  
5

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

- **put\_down\_box\_in\_location**: simile alla passata, permette al robot di posare una scatola nella location in cui si trova

```
(:action put_down_box_in_workstation
  :parameters (?r - robot ?box - box ?ws - workstation)
  :precondition (and (at_ws ?r ?ws) (carrying ?r ?box))
  :effect (and (not (carrying ?r ?box)) (free ?r) (at_ws ?box ?
    ws)))
)
```

1  
2  
3  
4  
5

- **put\_up\_box\_from\_workstation**: permette al robot di sollevare una scatola presente in una workstation:

```
(:action put_up_box_from_workstation
  :parameters (?r - robot ?box - box ?ws - workstation)
  :precondition (and (at_ws ?box ?ws) (at_ws ?r ?ws) (free ?r))
  :effect (and (carrying ?r ?box) (not (free ?r)) (not (at_ws ?
    box ?ws))))
)
```

1  
2  
3  
4  
5

– **parameters**: l’agente robotico che esegue l’azione, la scatola e la workstation.

– **precondition**:

- \* 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\_location**: analoga a quella precedente, permette all’agente robotico di sollevare una scatola da una location

```
(:action put_up_box_from_location
  :parameters (?r - robot ?box - box ?loc - location)
  :precondition (and (at_loc ?box ?loc) (at_loc ?r ?loc) (free ?
    r))
  :effect (and (carrying ?r ?box) (not (free ?r)) (not (at_loc ?
    box ?loc))))
)
```

1  
2  
3  
4  
5

- **empty\_box\_in\_workstation**: azione in cui l'agente robotico svuota una scatola nella workstation, trasferendo il contenuto nella stessa

```
(:action empty_box_in_workstation
  :parameters (?r - robot ?box - box ?ws - workstation ?con -
    content)
  :precondition (and (free ?r) (at_ws ?r ?ws) (at_ws ?box ?ws) (
    filled ?box ?con))
  :effect (and (not (filled ?box ?con)) (is_empty ?box) (at_ws ?
    con ?ws))
)
```

1  
2  
3  
4  
5

- **parameters**: l'agente robotico che esegue l'azione, la scatola, il contenuto e la workstation;
- **precondition**:
  - \* 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 sarà vuota;
  - \* il contenuto si troverà nella workstation passata come parametro;

- **empty\_box\_in\_location**: analoga alla precedente, l'agente robotico svuota una scatola in una location trasferendo il contenuto nella stessa posizione

```
(:action empty_box_in_location
  :parameters (?r - robot ?box - box ?con - content ?loc -
    location)
  :precondition (and (free ?r) (at_loc ?r ?loc) (at_loc ?box ?
    loc) (filled ?box ?con))
  :effect (and (not (filled ?box ?con)) (is_empty ?box) (at_loc
    ?con ?loc))
)
```

1  
2  
3  
4  
5

- **fill\_box\_in\_workstation**: l'agente robotico può riempire una scatola vuota prendendo il contenuto dalla workstation, qualora presente

```
(:action fill_box_in_workstation
  :parameters (?r - robot ?box - box ?ws - workstation ?con -
    content)
  :precondition (and (free ?r) (at_ws ?r ?ws) (at_ws ?box ?ws) (
    at_ws ?con ?ws) (is_empty ?box))
  :effect (and (not (at_ws ?con ?ws)) (filled ?box ?con) (not (
    is_empty ?box)))
)
```

1  
2  
3  
4  
5

- **parameters:** l’agente robotico che esegue l’azione, la scatola, la workstation e il contenuto;
- **precondition:**
  - \* 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;

- **fill\_box\_in\_location:** simile alla precedente, il robot, se disponibile e posizionato in una location con un box vuoto, lo riempie prendendo il contenuto dalla stessa location

```
(:action fill_box_in_location
  :parameters (?r - robot ?box - box ?con - content ?loc -
    location)
  :precondition (and (not (is_warehouse ?loc)) (free ?r) (at_loc
    ?r ?loc) (at_loc ?box ?loc) (at_loc ?con ?loc) (is_empty ?
    box))
  :effect (and (not (at_loc ?con ?loc)) (filled ?box ?con) (not
    (is_empty ?box)))
)
```

- **fill\_box\_in\_warehouse:** azione analoga alle precedenti, questa prescrive che l’agente può riempire una scatola solo se il robot, la scatola e i materiali di interesse si trovano nella warehouse e la scatola è vuota

```
(:action fill_box_in_warehouse
  :parameters (?r - robot ?box - box ?con - content ?loc -
    location)
  :precondition (and (is_warehouse ?loc) (free ?r) (at_loc ?r ?
    loc) (at_loc ?box ?loc) (at_loc ?con ?loc) (is_empty ?box))
  :effect (and (filled ?box ?con) (not (is_empty ?box)))
)
```

## 1.2 Problem 1 - Istanza semplificata

Per verificare quanto descritto nel **Domain.pddl** file e testare le azioni, nonché i predicati definiti, è stata creata una prima istanza del problema le cui specifiche sono riportate di seguito:

```
(define (problem simple_instance) 1
  (:domain industrial_manufacturing_service) 2

  (:objects 3
    location1 location2 warehouse - location
    box1 box2 - box
    bolts valves - content
    robot - robot
    workstation1 workstation2 - workstation
  ) 4

  (:init 5
    ;il robot viene collocato nella warehouse
    (at_loc robot warehouse) 6

    ;si istanziano due box collocate entrambe alla warehouse
    (at_loc box1 warehouse) 7
    (at_loc box2 warehouse) 8

    ;i materiali di interesse (di tipo content) sono nella warehouse
    (at_loc bolts warehouse) 9
    (at_loc valves warehouse) 10

    ;inizialmente le box sono vuote
    (is_empty box1) 11
    (is_empty box2) 12

    ;il robot risulta libero
    (free robot) 13

    ;si stabiliscono le interconnessioni tra le location
    (connected location1 location2) 14
    (connected location2 location1) 15
    (connected location2 warehouse) 16
    (connected warehouse location2) 17
    (connected warehouse location1) 18
    (connected location1 warehouse) 19

    ;si collocano le worstation all'interno delle location cui
    ;afferiscono
    (at_loc workstation1 location1) 20
    (at_loc workstation2 location2) 21

    (is_warehouse warehouse) 22
  ) 23

  (:goal 24
    ;il robot deve essere stato spostato dalla warehouse
    (not (at_loc robot warehouse)) 25
    ;e deve essere stato spostato in location2
    (at_loc robot location2) 26
  ) 27
```

	47
(and	48
; il <b>goal</b> prevede che le due box siano lasciate alle due	49
workstation presenti, ciascuna con un materiale specifico	
(at_ws box1 workstation1)	50
(at_ws box2 workstation2)	51
(filled box1 bolts)	52
(filled box2 valves)	53
)	54
)	55
)	56

- Tre location presenti, tra cui una warehouse;
- Due scatole per il trasporto di materiale;
- Due tipologie di contenuto: bulloni (bolts) e valvole (valves);
- Un unico agente robot che si occuperà del trasporto di quanto elencato fin'ora;
- Due workstation;

Lo **stato iniziale** si presenta con le seguenti caratteristiche:

- L'agente robot inizialmente si trova nella warehouse ed è libero;
- La scatola (box1) è vuota, posizionata nella warehouse, la scatola (box2) presenta le stesse caratteristiche;
- Bulloni e valvole sono presenti inizialmente nella warehouse;
- La workstation1 si trova nella location1 mentre la workstation2 si trova nella location2;
- La rete di connessioni tra le varie location è tale per cui la warehouse è connessa alla location1, la location1 è connessa alla location2 mentre quest'ultima è collegata alla warehouse.

Il **goal** è il seguente:

- La box1 dovrà trovarsi nella workstation1 riempita con bolts;
- La box2 dovrà trovarsi nella workstation2 riempita con valves;

### 1.2.1 Risultati

Utilizzando il framework PDDL4J e un planner standard, in particolare l'**Enforced Hill Climbing con euristica Fast Forward**, è possibile ottenere la seguente soluzione, nonché piano di azioni:

```
problem instantiation done successfully (70 actions, 43 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_location robot box2 warehouse) [0]
03: ( move_to_location robot warehouse location2) [0]
04: ( enter_workstation robot location2 workstation2) [0]
05: (put_down_box_in_workstation robot box2 workstation2) [0]
06: ( exit_workstation robot location2 workstation2) [0]
07: ( move_to_location robot location2 location1) [0]
08: ( move_to_location robot location1 warehouse) [0]
09: ( put_up_box_from_location robot box1 warehouse) [0]
10: ( move_to_location robot warehouse location1) [0]
11: ( enter_workstation robot location1 workstation1) [0]
12: (put_down_box_in_workstation robot box1 workstation1) [0]

time spent:      0.07 seconds parsing
                  0.07 seconds encoding
                  0.01 seconds searching
                  0.15 seconds total time

memory used:    0.35 MBytes for problem representation
                  0.00 MBytes for searching
                  0.35 MBytes total
```

Figura 1: *Risultati ottenuti con l'uso del planner ENFORCED HILL CLIMBING*

Il comando utilizzato da terminale per ottenere il piano è il seguente:

```
java -cp classes:lib/pddl4j-4.0.0.jar fr.uga.pddl4j.planners.
      statespace.FF \
/home/aiguy/Desktop/ProgettoIA_Curcio_Gullone/task1_modelling/pddl/
      domain.pddl" \
/home/aiguy/Desktop/ProgettoIA_Curcio_Gullone/task1_modelling/pddl/
      problem_simple_instance.pddl" \
>> /home/aiguy/Desktop/ProgettoIA_Curcio_Gullone/task1_modelling/
      results/EHC_FF_simple_instance.txt
```

### 1.3 Problem 2 - Istanza complessa

Si è deciso di testare quanto progettato su di una seconda istanza più complessa rispetto alla precedente, il cui **Problem.pddl** è il seguente:

```
(define (problem two_robots_instance_extended)
  (:domain industrial_manufacturing_service)

  (:objects
    location1 location2 location3 warehouse - location
    box1 box2 box3 - box
    bolts valves screws - content
    robot1 robot2 - robot
    workstation1 workstation2 workstation3 - workstation
  )
  (:init
    (at_loc robot1 warehouse)
    (at_loc robot2 location3)

    (at_loc box1 warehouse)
    (at_loc box2 warehouse)
    (at_loc box3 location3)

    (at_loc bolts warehouse)
    (at_loc valves warehouse)
    (at_loc screws location3)

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

    (free robot1)
    (free robot2)

    (connected location1 location2)
    (connected location2 location3)
    (connected location3 warehouse)
    (connected warehouse location1)

    (at_loc workstation1 location1)
    (at_loc workstation2 location2)
    (at_loc workstation3 location3)

    (is_warehouse warehouse)
  )

  (:goal
    (and
      (at_ws box1 workstation1)
      (at_ws box2 workstation2)
      (at_ws box3 workstation3)
      (filled box1 bolts)
    )
  )
)
```

(filled box2 valves)	50
(filled box3 screws)	51
)	52
)	53
)	54

- Quattro location presenti, tra cui una warehouse;
- Tre scatole per il trasporto di materiale;
- Tre tipologie di contenuto: bulloni (bolts), valvole (valves) e viti (screws);
- Due agenti robot, piuttosto che il singolo dell'istanza precedente, che si occuperanno del trasporto di quanto elencato fin'ora;
- Tre workstation;

Lo **stato iniziale** si presenta con le seguenti caratteristiche:

- Gli agenti robot inizialmente si trovano nella warehouse e sono entrambi liberi;
- La scatola (box1) è vuota, posizionata nella warehouse e lo stesso vale per le scatole rimanenti (box2, box3);
- Bulloni, valvole e viti sono presenti inizialmente nella warehouse;
- La workstation1 si trova nella location1, la workstation2 si trova nella location2 mentre la workstation3 si trova nella location3;
- Si noti come la rete di connessioni tra le varie location è tale per cui la warehouse è connessa alla location1, la location1 è connessa alla location2, la location2 alla location3, quest'ultima è a sua volta connessa alla warehouse.

Il **goal** è il seguente:

- La box1 dovrà trovarsi nella workstation1 riempita con i bulloni (bolts);
- La box2 dovrà trovarsi nella workstation2 riempita con valvole (valves);
- La box3 dovrà trovarsi nella workstation3 riempita con le viti (screws).

### 1.3.1 Risultati

In questo caso si è deciso di provare due planner già integrati in PDDL4J, rispettivamente **A\* con euristica Fast Forward**, il cui risultato è il seguente:

```
problem instantiation done successfully (356 actions, 84 fluents)

* Starting ASTAR search with FAST_FORWARD heuristic
* ASTAR search succeeded

found plan as follows:

00: ( fill_box_in_warehouse robot1 box1 bolts warehouse) [0]
01: ( fill_box_in_warehouse robot1 box2 valves warehouse) [0]
02: ( put_up_box_from_location robot1 box2 warehouse) [0]
03: ( move_to_location robot1 warehouse location1) [0]
04: ( move_to_location robot1 location1 location2) [0]
05: ( enter_workstation robot1 location2 workstation2) [0]
06: ( fill_box_in_location robot2 box3 screws location3) [0]
07: ( put_up_box_from_location robot2 box3 location3) [0]
08: ( enter_workstation robot2 location3 workstation3) [0]
09: (put_down_box_in_workstation robot1 box2 workstation2) [0]
10: (put_down_box_in_workstation robot2 box3 workstation3) [0]
11: ( exit_workstation robot2 location3 workstation3) [0]
12: ( move_to_location robot2 location3 warehouse) [0]
13: ( put_up_box_from_location robot2 box1 warehouse) [0]
14: ( move_to_location robot2 warehouse location1) [0]
15: ( enter_workstation robot2 location1 workstation1) [0]
16: (put_down_box_in_workstation robot2 box1 workstation1) [0]

time spent:      0.05 seconds parsing
                  0.11 seconds encoding
                  0.23 seconds searching
                  0.39 seconds total time

memory used:    1.28 MBytes for problem representation
                  0.23 MBytes for searching
                  1.51 MBytes total
```

Figura 2: Risultati ottenuti con l'uso del planner A\*

Il comando utilizzato da terminale per ottenere il piano è il seguente:

```
java -cp classes:lib/pddl4j-4.0.0.jar fr.uga.pddl4j.planners.
      statespace.FF \
/home/aiguy/Desktop/ProgettoIA_Curcio_Gullone/task1_modelling/pddl/
      domain.pddl" \
/home/aiguy/Desktop/ProgettoIA_Curcio_Gullone/task1_modelling/pddl/
      problem_two_robots_instance.pddl" \
>> /home/aiguy/Desktop/ProgettoIA_Curcio_Gullone/task1_modelling/
      results/EHC_FF_two_robots_instance.txt
```

Il secondo planner è stato, come prima, l'**Enforced Hill Climbing con euristica Fast Forward**:

```

problem instantiation done successfully (356 actions, 84 fluents)

* Starting ENFORCED_HILL_CLIMBING search with FAST_FORWARD heuristic
* ENFORCED_HILL_CLIMBING search succeeded

found plan as follows:

00: ( fill_box_in_location robot2 box3 screws location3) []
01: ( put_up_box_from_location robot2 box3 location3) []
02: ( enter_workstation robot2 location3 workstation3) []
03: (put_down_box_in_workstation robot2 box3 workstation3) []
04: ( fill_box_in_warehouse robot1 box1 bolts warehouse) []
05: ( fill_box_in_warehouse robot1 box2 valves warehouse) []
06: ( exit_workstation robot2 location3 workstation3) []
07: ( move_to_location robot2 location3 warehouse) []
08: ( put_up_box_from_location robot1 box1 warehouse) []
09: ( move_to_location robot1 warehouse location1) []
10: ( enter_workstation robot1 location1 workstation1) []
11: (put_down_box_in_workstation robot1 box1 workstation1) []
12: ( put_up_box_from_location robot2 box2 warehouse) []
13: ( move_to_location robot2 warehouse location1) []
14: ( move_to_location robot2 location1 location2) []
15: ( enter_workstation robot2 location2 workstation2) []
16: (put_down_box_in_workstation robot2 box2 workstation2) []

time spent:      0.05 seconds parsing
                  0.15 seconds encoding
                  0.09 seconds searching
                  0.29 seconds total time

memory used:    1.28 MBytes for problem representation
                  0.00 MBytes for searching
                  1.28 MBytes total

```

Figura 3: Risultati ottenuti con l'uso del planner Enforced Hill Climbing

Il comando utilizzato da terminale per ottenere il piano è il seguente:

```

java -cp classes:lib/pddl4j-4.0.0.jar fr.uga.pddl4j.planners.
      statespace.HSP \
/home/aiguy/Desktop/ProgettoIA_Curcio_Gullone/task1_modelling/pddl/
      domain.pddl" \
/home/aiguy/Desktop/ProgettoIA_Curcio_Gullone/task1_modelling/pddl/
      problem_two_robots_instance.pddl" \
>> /home/aiguy/Desktop/ProgettoIA_Curcio_Gullone/task1_modelling/
      results/ASTAR_FF_two_robots_instance.txt

```

## 2 Classical Planning

Il secondo task dell'elaborato consiste nella progettazione di un planner personalizzato sfruttando **PDDL4J** con l'obiettivo finale di risolvere due istanze particolari.

Le soluzioni calcolate dal planner programmato devono inoltre mostrare:

1. Il tempo impiegato per trovare il plan (non oltre i 10 minuti);
2. Il numero di stati valutati;
3. La quantità di memoria utilizzata;

### 2.1 Planner personalizzato ed euristica personalizzata

Si procede dunque a questo punto nel mostrare il codice del planner costruito seguendo la guida apposita ([http://pddl4j.imag.fr/writing\\_your\\_own\\_planner.html](http://pddl4j.imag.fr/writing_your_own_planner.html)):

#### 2.1.1 Planner A\* Search

```
import fr.uga.pddl4j.heuristics.state.StateHeuristic;           1
import fr.uga.pddl4j.parser.DefaultParsedProblem;                 2
import fr.uga.pddl4j.parser.RequireKey;                            3
import fr.uga.pddl4j.plan.Plan;                                    4
import fr.uga.pddl4j.plan.SequentialPlan;                         5
import fr.uga.pddl4j.planners.AbstractPlanner;                   6
import fr.uga.pddl4j.planners.Planner;                            7
import fr.uga.pddl4j.planners.PlannerConfiguration;              8
import fr.uga.pddl4j.planners.ProblemNotSupportedException;        9
import fr.uga.pddl4j.planners.SearchStrategy;                     10
import fr.uga.pddl4j.planners.statespace.search.StateSpaceSearch; 11
import fr.uga.pddl4j.problem.DefaultProblem;                      12
import fr.uga.pddl4j.problem.Problem;                            13
import fr.uga.pddl4j.problem.State;                             14
import fr.uga.pddl4j.problem.operator.Action;                     15
import fr.uga.pddl4j.problem.operator.ConditionalEffect;        16
import org.apache.logging.log4j.LogManager;                       17
import org.apache.logging.log4j.Logger;                           18
import picocli.CommandLine;                                       19
                                                               20
//Import di librerie utili al calcolo della memoria e del tempo
utilizzati                                         21
import java.lang.management.ManagementFactory;                   22
import java.lang.management.MemoryMXBean;                        23
```

```

import java.lang.management.MemoryUsage;                                24
import java.util.*;                                                 25
                                                               26
import java.util.Comparator;                                         27
import java.util.HashSet;                                            28
import java.util.List;                                              29
import java.util.PriorityQueue;                                         30
import java.util.Set;                                               31
                                                               32
@CommandLine.Command(name = "IndustrialProcessPlanner",             33
    version = "IndustrialProcessPlanner 1.0",                         34
    description = "Solves a specified planning problem using A*      35
        search strategy.",                                             
    sortOptions = false,                                              36
    mixinStandardHelpOptions = true,                                     37
    headerHeading = "Usage:%n",                                         38
    synopsisHeading = "%n",                                             39
    descriptionHeading = "%nDescription:%n%n",                           40
    parameterListHeading = "%nParameters:%n",                            41
    optionListHeading = "%nOptions:%n")                                 42
                                                               43

public class IndustrialProcessPlanner extends AbstractPlanner {          44
                                                               45
/* The class logger */
private static final Logger LOGGER = LogManager.getLogger(
    IndustrialProcessPlanner.class.getName());                           46
private double heuristicWeight;                                           47
private StateHeuristic.Name heuristic;                                    48
//useNewHeuristic = per usare l'euristica definita ad Hoc           49
private int useNewHeuristic;                                           50
                                                               51

// Instantiates the planning problem from a parsed problem            52
@Override
public Problem instantiate(DefaultParsedProblem problem) {           53
    final Problem pb = new DefaultProblem(problem);                   54
    pb.instantiate();                                                 55
    return pb;                                                       56
} //instantiate

// Search a solution plan to a specified domain and problem using      57
A*.                                                               58
                                                               59
                                                               60
                                                               61

```

```

@Override 62
public Plan solve(final Problem problem) throws 63
    ProblemNotSupportedException { 64

        //Preliminarmente si verifica se il problema risulta 65
        //risolvibile dal planner
        if (!this.isSupported(problem)) { throw new 66
            ProblemNotSupportedException("Cannot solve the problem");} 67

        //Mediante il seguente check sulla variabile di classe si va 68
        //a scegliere l'euristica di riferimento
        if (this.getHeuristicNew() == 1){ 69

            //Se il valore della variabile di classe e' 1 allora si 70
            //utilizza un'implementazione custom di A* con l'
            //euristica personalizzata 71

            LOGGER.info("* Starting A* Search with 72
                IndustrialProcessHeuristic \n"); 73

            //Per poter tenere traccia della memoria usata dall'
            //algoritmo 74
            MemoryMXBean memoryBean = ManagementFactory. 75
                getMemoryMXBean();
            MemoryUsage beforeHeapMemoryUsage = memoryBean. 76
                getHeapMemoryUsage();
            long beforeUsedMemory = beforeHeapMemoryUsage.getUsed 77
                (); 78

            //Per tenere traccia del tempo usato dall'algoritmo 79
            final long begin = System.currentTimeMillis(); 80

            //Si esegue l'algoritmo 81
            final Plan plan = this.astar(problem); 82
            final long end = System.currentTimeMillis(); 83

            MemoryUsage afterHeapMemoryUsage = memoryBean. 84
                getHeapMemoryUsage(); 85
            long afterUsedMemory = afterHeapMemoryUsage.getUsed() 86
                ; 87

            88
            89

```

```

        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 si usa 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");
        //Si esegue l'algoritmo
        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;
    }

```

```

} // solve                                         122
                                              123
// Returns if a specified problem is supported by the planner. 124
Just ADL problem can be solved by this planner.
@Override                                         125
public boolean isSupported(Problem problem) { 126
    return !problem.getRequirements().contains(RequireKey.
        ACTION_COSTS)
        && !problem.getRequirements().contains(RequireKey. 128
            CONSTRAINTS)
        && !problem.getRequirements().contains(RequireKey. 129
            CONTINUOUS_EFFECTS)
        && !problem.getRequirements().contains(RequireKey. 130
            DERIVED_PREDICATES)
        && !problem.getRequirements().contains(RequireKey. 131
            DURATIVE_ACTIONS)
        && !problem.getRequirements().contains(RequireKey. 132
            DURATION_INEQUALITIES)
        && !problem.getRequirements().contains(RequireKey. 133
            FLUENTS)
        && !problem.getRequirements().contains(RequireKey. 134
            GOAL_UTILITIES)
        && !problem.getRequirements().contains(RequireKey. 135
            METHOD_CONSTRAINTS)
        && !problem.getRequirements().contains(RequireKey. 136
            NUMERIC_FLUENTS)
        && !problem.getRequirements().contains(RequireKey. 137
            OBJECT_FLUENTS)
        && !problem.getRequirements().contains(RequireKey. 138
            PREFERENCES)
        && !problem.getRequirements().contains(RequireKey. 139
            TIMED_INITIAL_LITERAL)
        && !problem.getRequirements().contains(RequireKey. 140
            HIERARCHY);
} // isSupported                                     141
                                              142
// Sets the weight of the heuristic.               143
@CommandLine.Option(names = {"-w", "--weight"}, defaultValue = " 144
    1.0",
    paramString = "<weight>", description = "Set the weight of the 145
        heuristic (preset 1.0).")

```

```

public void setHeuristicWeight(final double weight) { 146
    if (weight <= 0) { 147
        throw new IllegalArgumentException("Weight <= 0"); 148
    } 149
    this.heuristicWeight = weight; 150
} //setHeuristicWeight 151

// Set the name of heuristic used by the planner to the solve a 152
// planning problem. 153

@CommandLine.Option(names = {"-e", "--heuristic"}, defaultValue = 154
    "FAST_FORWARD",
    description = "Set the heuristic : AJUSTED_SUM, AJUSTED_SUM2, 155
    AJUSTED_SUM2M, COMBO, "
    + "MAX, FAST_FORWARD SET_LEVEL, SUM, SUM_MUTEX (preset: 156
    FAST_FORWARD)") 157

public void setHeuristic(StateHeuristic.Name heuristic) { 157
    this.heuristic = heuristic;
} //setHeuristic 159

/* Metodo per settare la nuova euristica 160
 * 0 -> Utilizzo dell'euristica di base 161
 * 1 -> Utilizzo dell'euristica personalizzata 162
 */
163

@CommandLine.Option(names = {"-en", "--heuristicNew"}, 165
    defaultValue = "0",
    description = "Set the heuristic : 1") 166

public void setHeuristicNew(int heuristicsNew) { 167
    this.useNewHeuristic = heuristicsNew;
} //setHeuristicNew 168

public final int getHeuristicNew() { 169
    return this.useNewHeuristic;
} //getHeuristicNew 170

171
172
173

```

Si noti come il metodo **setHeuristicNew** sia necessario per esprimere la possibilità, data all'utente, di cambiare l'euristica utilizzata all'interno del planner a seconda del valore fornito a riga di comando:

- **-en 0**: il planner sfrutta l'euristica di default definita con il comando -n;
- **-en 1**: il planner sfrutta l'euristica custom; a tal proposito l'utente non dovrà specificare a riga di comando l'opzione -n;

In sostanza l'utente può passare un'opzione da linea di comando e il valore viene memorizzato nella variabile **useNewHeuristic** tramite il metodo **setHeuristicNew**. Il metodo **getHeuristicNew** permette di recuperare il valore assegnato, che può essere utilizzato in altre parti del programma.

Il codice prosegue di seguito:

```
// Returns the name of the heuristic used by the planner to solve 1
// a planning problem.
public final StateHeuristic.Name getHeuristic() { 2
    return this.heuristic; 3
} //getHeuristic 4

// Returns the weight of the heuristic 5
public final double getHeuristicWeight() { 6
    return this.heuristicWeight; 7
} //getHeuristicWeight 8

// Search a solution plan for a planning problem using an A* 9
// search strategy. 10

//Algoritmo A* modificato ad hoc 11
public Plan astar(Problem problem){ 12

    //Inizializzazione dell'euristica 13
    IndustrialProcessHeuristic heuristic = new 14
        IndustrialProcessHeuristic(problem); 15

    //Inizializzazione dello stato iniziale che corrisponde allo 16
        stato iniziale del problem 17
    final State init = new State(problem.getInitialState()); 18

    //Insieme per tracciare i nodi già visitati ed evitare 19
        esplorazioni ripetute 20
    final Set<Node> close = new HashSet<>(); 21

    //Insieme per tracciare i nodi già visitati ed evitare 22
        esplorazioni ripetute 23
    final Set<Node> open = new HashSet<>();
```

```

    final double weight = this.getHeuristicWeight();           24
                                                25

    //Coda di priorita' per esplorare i nodi, con priorita'
    // basata su una funzione di costo euristica f(n) = g(n) + w*
    // h(n)
    final PriorityQueue<Node> open = new PriorityQueue<>(100, new        27
        Comparator<Node>() {
            public int compare(Node n1, Node n2) {           28
                double f1 = weight * n1.getHeuristic() + n1.getCost()   29
                    ;
                double f2 = weight * n2.getHeuristic() + n2.getCost()   30
                    ;
                return Double.compare(f1, f2);                  31
            }
        });
                                                32

    //Inizializzazione della radice dell'albero di ricerca      34
    final Node root = new Node(init, null, -1, 0, heuristic.     35
        estimate(init, problem.getGoal()));                      36
                                                37

    //Si inserisce il nodo radice nella coda dei nodi da
    // esplorare
    open.add(root);                                         38
                                                39

    Plan plan = null;                                       40
                                                41

    //Settaggio del timeout per la ricerca
    final int timeout = this.getTimeout() * 1000;          43
    long time = 0;                                         44
                                                45

    //Inizio della ricerca: si procede fintanto che vi sono nodi
    // da visitare e non e' scaduto il timeout
    while (!open.isEmpty() && plan==null && time<timeout) { 47
                                                48

        //Estrazione del nodo con il costo piu' basso dalla coda
        // di priorita'
        final Node current = open.poll();                   50
                                                51

        //Si inserisce il nodo estratto nell'insieme dei nodi
        // esplorati
        close.add(current);                            53
                                                54

```

```

//Se il goal risulta soddisfatto allora si restituisce il
    plan
if (current.satisfy(problem.getGoal())) {
    return this.extractPlan(current, problem);
} else {
    //Altrimenti si esplorano le azioni applicabili nello
    stato corrente
    for (int i=0; i<problem.getActions().size(); i++) {
        Action a = problem.getActions().get(i);
        //Si verifica se l'azione risulta applicabile a
        partire dallo stato corrente
        if (a.isApplicable(current)) {
            //Si crea il prossimo nodo
            Node next = new Node(current);
            //Se l'azione e' lecita allora si considerano
            gli effetti che vengono inseriti in una
            lista
            final List<ConditionalEffect> effects = a.
                getConditionalEffects();
            for (ConditionalEffect ce : effects) {
                if (current.satisfy(ce.getCondition())) {
                    next.apply(ce.getEffect());
                }
            }
            //Si inserisce il prossimo nodo nella coda
            dei nodi da esplorare se non gi
            esplorato
            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);
            }
        }
    }
}

```

```

    }

    // Si restituisce il plan
    return plan;
} //astar

// Extracts a search from a specified node.
private Plan extractPlan(final Node node, final Problem problem)
{
    Node n = node;
    final Plan plan = new SequentialPlan();
    while (n.getAction() != -1) {
        final Action a = problem.getActions().get(n.getAction());
        plan.add(0, a);
        n = n.getParent();
    }
    return plan;
} //extractPlan

public static void main(String[] args) {
    try {
        final IndustrialProcessPlanner planner = new
            IndustrialProcessPlanner();
        CommandLine cmd = new CommandLine(planner);
        cmd.execute(args);
    } catch (IllegalArgumentException e) {
        LOGGER.fatal(e.getMessage());
    }
} //main
}

```

Il planner si poggia a sua volta su due classi fondamentali:

- **Node.java**: rappresenta un'estensione della classe **State** e modella un nodo in un albero di ricerca per algoritmi di pianificazione. Include informazioni sul nodo genitore, l'azione applicata, il costo accumulato, la profondità e un valore euristico per guidare la ricerca.

```
import fr.uga.pddl4j.problem.State; 1

public final class Node extends State { 2

    private Node parent; 3
    private int action; 4
    private double cost; 5
    private double heuristic; 6
    private int depth; 7

    public Node(State state) { 8
        super(state); 9
    } 10

    public Node(State state, Node parent, int action, double 11
        cost, double heuristic) { 12
        super(state); 13
        this.parent = parent; 14
        this.action = action; 15
        this.cost = cost; 16
        this.heuristic = heuristic; 17
        this.depth = -1; 18
    } 19

    public Node(State state, Node parent, int action, double 20
        cost, int depth, double heuristic) { 21
        super(state); 22
        this.parent = parent; 23
        this.action = action; 24
        this.cost = cost; 25
        this.depth = depth; 26
        this.heuristic = heuristic; 27
    } 28

    public final int getAction() { 29
        return action; 30
    } 31

    public final double getCost() { 32
        return cost; 33
    } 34
```

```

        return this.action;                                35
    }                                                    36
    public final void setAction(final int action) {      37
        this.action = action;                            38
    }                                                    39
                                                        40
    public final Node getParent() {                      41
        return parent;                                 42
    }                                                    43
    public final void setParent(Node parent) {          44
        this.parent = parent;                           45
    }                                                    46
                                                        47
    public final double getCost() {                     48
        return cost;                                  49
    }                                                    50
    public final void setCost(double cost) {           51
        this.cost = cost;                            52
    }                                                    53
                                                        54
    public final double getHeuristic() {                55
        return heuristic;                           56
    }                                                    57
    public final void setHeuristic(double estimates) { 58
        this.heuristic = estimates;                  59
    }                                                    60
                                                        61
    public int getDepth() {                           62
        return this.depth;                           63
    }                                                    64
    public void setDepth(final int depth) {           65
        this.depth = depth;                          66
    }                                                    67
                                                        68
    public final double getValueF(double weight) {    69
        return weight * this.heuristic + this.cost;  70
    }                                                    71
}                                                    72

```

### 2.1.2 Euristica personalizzata

L'euroistica ideata per il dominio applicativo preso in esame è contenuta nel file **IndustrialProcessHeuristic.java**. Il codice di seguito:

```
import fr.uga.pddl4j.heuristics.state.RelaxedGraphHeuristic;          1
import fr.uga.pddl4j.planners.statespace.search.Node;                     2
import fr.uga.pddl4j.problem.Problem;                                     3
import fr.uga.pddl4j.problem.State;                                       4
import fr.uga.pddl4j.problem.operator.Action;                                5
import fr.uga.pddl4j.problem.operator.Condition;                            6
import java.lang.reflect.Method;                                           7
import java.util.List;
import java.util.Optional;

public final class IndustrialProcessHeuristic extends RelaxedGraphHeuristic { 8

    public IndustrialProcessHeuristic (Problem problem) {                      9
        super(problem);
        //l'euroistica implementata non risulta ammissibile
        super.setAdmissible(false);                                              10
    }

    @Override
    public int estimate(State state, Condition goal) {                         11
        super.setGoal(goal);
        super.expandRelaxedPlanningGraph(state);
        return super.isGoalReachable() ? 1 : Integer.MAX_VALUE;                12
    } //estimate                                                               13

    @Override
    public double estimate(Node node, Condition goal) {                         14
        return this.estimate((State) node, goal);                                15
    } //estimate                                                               16

    public double customEstimate(State state, Condition goal, Action a, double currentHeuristic) { 17
        super.setGoal(goal);
        this.expandRelaxedPlanningGraph(state);                                 18
        double heuristicValue = currentHeuristic;                             19
        return heuristicValue;                                                 20
    } //customEstimate                                                       21

}
```

```

        if(a.getName().contains("move"))
            heuristicValue -= 1;
        else if (a.getName().contains("fill") || a.getName().contains("workstation") || a.getName().contains("load"))
            heuristicValue -= 0.8;
        else if (a.getName().equals("empty_box_in_location"))
            heuristicValue += 0.5;
        else if (a.getName().equals("empty_box_in_workstation"))
            heuristicValue -= 1.5;
        else if (a.getName().equals("put_down_box_in_location"))
            heuristicValue += 0.5;
        return super.isGoalReachable() ? heuristicValue : Integer.MAX_VALUE;
    } //customEstimate
}

```

Il metodo **customEstimate** implementa una serie di penalizzazioni e incentivi basati sul tipo di azione per orientare il planner verso strategie più efficaci. Di seguito, sono descritti i principali comportamenti:

- **Movimento:** se l'azione da eseguire comporta uno spostamento ('move'), l'euristica viene incentivata con una **riduzione di 1** dal valore corrente. Questo incoraggia i movimenti del robot, considerati essenziali per raggiungere gli obiettivi.
- **Caricamenti e operazioni presso le workstation:** azioni come il riempimento delle scatole ('fill'), le operazioni di caricamento ('load'), e quelle che coinvolgono direttamente le **workstation** vedono una **riduzione di 0.8** del valore corrente dell'euristica. Questo favorisce il progresso delle operazioni di rifornimento, che sono cruciali nel contesto del problema.
- **Svuotamento delle scatole nelle location:** l'azione di svuotamento delle scatole in una location ('empty\_box\_in\_location') viene penalizzata con un **aumento di 0.5** del valore corrente dell'euristica. Tale penalizzazione riflette la preferenza per la consegna dei materiali alle workstation piuttosto che alle location generiche.
- **Svuotamento delle scatole nelle workstation:** al contrario, l'azione di svuotamento delle scatole presso una workstation ('empty\_box\_in\_workstation') è fortemente incentivata con una **riduzione di 1.5** dell'euristica. Questa operazione è considerata strategicamente importante per soddisfare gli obiettivi del problema.
- **Deposito delle scatole in location:** l'azione di deposito delle scatole in una location ('put\_down\_box\_in\_location') subisce una penalizzazione con un **incremento di 0.5** dell'euristica, per scoraggiare il rilascio di scatole in location che non sono stazioni di lavoro.

La classe dell’euristica personalizzata **IndustrialProcessHeuristic** è un’estensione della classe **RelaxedGraphHeuristic**, progettata per calcolare euristiche basate su una versione rilassata del problema originale di ricerca o di pianificazione. Utilizzare un grafo rilassato consente di ridurre la complessità del calcolo euristico, rendendo più veloce il processo di ricerca.

Il metodo **estimate** presente è ereditato dalla classe **RelaxedGraphHeuristic** e viene modificato per garantire che il valore dell’euristica rimanga sempre non negativo, restituendo un valore compreso tra **1** e **INTEGER.MAX\_VALUE**.

Questa scelta evita che l’euristica possa assumere valori negativi durante le prime iterazioni della ricerca, garantendo stabilità nel processo di pianificazione.

L’invocazione del metodo **estimate** nel nodo radice del planner assicura inoltre che l’euristica sia calcolata in modo consistente, con il metodo **customEstimate** utilizzato per aggiornare dinamicamente il valore dell’euristica in base alle azioni eseguite durante la ricerca.

## 2.2 Istanza 1

Le condizioni iniziali per l'Instance 1 sono progettate per rappresentare uno stato del sistema in cui tutti i materiali e le risorse sono centralizzati, e il robot ha il compito di distribuire i materiali alle stazioni di lavoro.

**Si noti che il codice del Domain.pddl file non è stato riportato in quanto corrisponde a quello descritto per il task precedente.**

### 2.2.1 Problem

Per semplicità di lettura si è deciso di riportare solamente il codice del file. La sezione relativa alle condizioni iniziali è riportata di seguito:

```
(define (problem classical_planning_problem) 1
  (:domain industrial_manufacturing_service) 2
  (:objects 3
    robot - robot
    warehouse location1 location2 - location
    workstation1 workstation2 workstation3 - workstation
    box1 box2 box3 - box
    bolt screw - content
  )
  (:init 11
    (at_loc robot warehouse)
    (free robot)
    (is_warehouse warehouse)
    ;I box sono collocati nella warehouse
    (at_loc box1 warehouse)
    (at_loc box2 warehouse)
    (at_loc box3 warehouse)
    ;I materiali da mettere nelle box sono nella warehouse
    (at_loc bolt warehouse)
    (at_loc screw warehouse)
    ;Le scatole sono vuote
  )
)
```

```

(is_empty box1) 29
(is_empty box2) 30
(is_empty box3) 31
                32
; Si collocano le workstation 33
(at_loc workstation1 location1) 34
(at_loc workstation2 location2) 35
(at_loc workstation3 location2) 36
                37
; Si stabiliscono le connessioni 38
(connected warehouse location1) 39
(connected location1 location2) 40
(connected location1 warehouse) 41
(connected location2 location1) 42
)

```

Mentre la sezione contenente il goal è:

```

(:goal 1
  (and 2
    ; Almeno una workstation necessita di un materiale (bullone) 3
    (at_ws bolt workstation2) 4
                5
    ; Almeno una workstation non necessita di nulla 6
    (not (at_ws bolt workstation1)) 7
    (not (at_ws screw workstation1)) 8
                9
    ; Almeno una workstation necessita di pi materiali (bulloni e 10
      viti)
    (at_ws bolt workstation3) 11
    (at_ws screw workstation3) 12
                13
  )
)

```

Si noti come la rete di connessioni tra le varie location è tale per cui la warehouse è connessa alla location1 (e viceversa) e la location1 è connessa alla location2 (e viceversa). Si evidenzia come il predicato **connected** non sia simmetrico nella sua definizione standard ma lo si è reso tale attraverso specifiche istanziazioni nel file del problema. La sua ripetizione con argomenti scambiati permette infatti di simulare il **comportamento transitivo** del singolo predicato **connected**. La transitività del predicato è necessaria per assicurarsi che le location siano tra di loro mutuamente collegate.

### 2.2.2 Risultati

I risultati riportati sono stati ottenuti utilizzando diversi planner build-in quali:

- Enforced Hill Climbing w/ euristica Fast Forward:

```

problem instantiation done successfully (118 actions, 57 fluents      1
)
                                         2
* Starting ENFORCED_HILL_CLIMBING search with FAST_FORWARD          3
    heuristic
* ENFORCED_HILL_CLIMBING search succeeded                           4
                                         5
found plan as follows:                                              6
                                         7
00: (      fill_box_in_warehouse robot box1 bolt warehouse)        7
01: (      put_up_box_from_location robot box1 warehouse)           8
02: (      move_to_location robot warehouse location1)             9
03: (      move_to_location robot location1 location2)            10
04: (      enter_workstation robot location2 workstation2)         11
05: (  put_down_box_in_workstation robot box1 workstation2)        12
06: ( empty_box_in_workstation robot box1 workstation2 bolt)       13
07: (      exit_workstation robot location2 workstation2)         14
08: (      move_to_location robot location2 location1)            15
09: (      move_to_location robot location1 warehouse)             16
10: (      fill_box_in_warehouse robot box2 bolt warehouse)         17
11: (      put_up_box_from_location robot box2 warehouse)           18
12: (      move_to_location robot warehouse location1)             19
13: (      move_to_location robot location1 location2)            20
14: (      enter_workstation robot location2 workstation3)         21
15: (  put_down_box_in_workstation robot box2 workstation3)        22
16: ( empty_box_in_workstation robot box2 workstation3 bolt)       23
17: (      exit_workstation robot location2 workstation3)         24
18: (      move_to_location robot location2 location1)            25
19: (      move_to_location robot location1 warehouse)             26
20: (      fill_box_in_warehouse robot box3 screw warehouse)        27
21: (      put_up_box_from_location robot box3 warehouse)           28
22: (      move_to_location robot warehouse location1)             29
23: (      move_to_location robot location1 location2)            30
24: (      enter_workstation robot location2 workstation3)         31
25: (  put_down_box_in_workstation robot box3 workstation3)        32
26: (empty_box_in_workstation robot box3 workstation3 screw)       33
                                         34
time spent:      0.06 seconds parsing                                35

```

	0.10 seconds encoding	36
	0.09 seconds searching	37
	0.24 seconds total time	38
		39
memory used:	0.52 MBytes for problem representation	40
	0.00 MBytes for searching	41
	0.52 MBytes total	42

Il comando utilizzato da terminale per ottenere il piano è il seguente:

```
java -cp classes:lib/pddl4j-4.0.0.jar fr.uga.pddl4j.planners.  
statespace.FF \  
pddl/instance_2_1/"domain.pddl" \  
pddl/instance_2_1/"problem.pddl" \  
>> results/instance_2_1/EHC_FF_planner_result.txt
```

- A\* Search w/ euristica Adjusted Sum;

```
problem instantiation done successfully (118 actions, 57 fluents  
)  
  
* Starting A* search with heuristic: AJUSTED_SUM  
* Starting A* search  
* A* search succeeded  
  
found plan as follows:  
00: (      fill_box_in_warehouse robot box2 screw warehouse)  
01: (      fill_box_in_warehouse robot box3 bolt warehouse)  
02: (      fill_box_in_warehouse robot box1 bolt warehouse)  
03: (      put_up_box_from_location robot box3 warehouse)  
04: (      move_to_location robot warehouse location1)  
05: (      move_to_location robot location1 location2)  
06: (      enter_workstation robot location2 workstation3)  
07: (      put_down_box_in_workstation robot box3 workstation3)  
08: ( empty_box_in_workstation robot box3 workstation3 bolt)  
09: (      exit_workstation robot location2 workstation3)  
10: (      move_to_location robot location2 location1)  
11: (      move_to_location robot location1 warehouse)  
12: (      put_up_box_from_location robot box2 warehouse)  
13: (      move_to_location robot warehouse location1)  
14: (      move_to_location robot location1 location2)  
15: (      enter_workstation robot location2 workstation3)  
16: (      put_down_box_in_workstation robot box2 workstation3)
```

```

17: (empty_box_in_workstation robot box2 workstation3 screw) 25
18: (exit_workstation robot location2 workstation3) 26
19: (move_to_location robot location2 location1) 27
20: (move_to_location robot location1 warehouse) 28
21: (put_up_box_from_location robot box1 warehouse) 29
22: (move_to_location robot warehouse location1) 30
23: (move_to_location robot location1 location2) 31
24: (enter_workstation robot location2 workstation2) 32
25: (put_down_box_in_workstation robot box1 workstation2) 33
26: (empty_box_in_workstation robot box1 workstation2 bolt) 34
27:                                         35
time spent:          0.06 seconds parsing 36
                  0.10 seconds encoding 37
                  1.84 seconds searching 38
                  2.00 seconds total time 39
28:                                         40
memory used:        0.52 MBytes for problem representation 41
                    13.69 MBytes for searching 42
                    14.21 MBytes total 43

```

- A\* Search w/ euristica Fast Forward;

```

problem instantiation done successfully (118 actions, 57 fluents 1
)
2
* Starting ASTAR search with FAST_FORWARD heuristic 3
* ASTAR search succeeded 4
5
found plan as follows: 6
00: (fill_box_in_warehouse robot box1 bolt warehouse) 7
01: (put_up_box_from_location robot box1 warehouse) 8
02: (move_to_location robot warehouse location1) 9
03: (move_to_location robot location1 location2) 10
04: (enter_workstation robot location2 workstation2) 11
05: (put_down_box_in_workstation robot box1 workstation2) 12
06: (empty_box_in_workstation robot box1 workstation2 bolt) 13
07: (exit_workstation robot location2 workstation2) 14
08: (move_to_location robot location2 location1) 15
09: (move_to_location robot location1 warehouse) 16
10: (fill_box_in_warehouse robot box2 bolt warehouse) 17
11: (fill_box_in_warehouse robot box3 screw warehouse) 18
12: (put_up_box_from_location robot box2 warehouse) 19

```

```

13: (           move_to_location robot warehouse location1) 20
14: (           move_to_location robot location1 location2) 21
15: (           enter_workstation robot location2 workstation3) 22
16: (   put_down_box_in_workstation robot box2 workstation3) 23
17: ( empty_box_in_workstation robot box2 workstation3 bolt) 24
18: (           exit_workstation robot location2 workstation3) 25
19: (           move_to_location robot location2 location1) 26
20: (           move_to_location robot location1 warehouse) 27
21: (           put_up_box_from_location robot box3 warehouse) 28
22: (           move_to_location robot warehouse location1) 29
23: (           move_to_location robot location1 location2) 30
24: (           enter_workstation robot location2 workstation3) 31
25: (   put_down_box_in_workstation robot box3 workstation3) 32
26: (empty_box_in_workstation robot box3 workstation3 screw) 33

34
time spent:      0.05 seconds parsing 35
                  0.09 seconds encoding 36
                  2.68 seconds searching 37
                  2.82 seconds total time 38
39
memory used:    0.52 MBytes for problem representation 40
                  15.36 MBytes for searching 41
                  15.88 MBytes total 42

```

Il planner personalizzato discusso in precedenza è stato testato utilizzando diverse varianti della funzione euristica riportate di seguito.

Il comando utilizzato da terminale per ottenere il piano, in ognuno dei casi, è il seguente:

```

java -cp classes:lib/pddl4j-4.0.0.jar IndustrialProcessPlanner \
pddl/instance_2_1/"domain.pddl" \
pddl/instance_2_1/"problem.pddl" -en 1\
>> results/instance_2_1/
ASTAR_IndustrialProcessHeuristic_planner_resutl_{version}.txt

```

- Variante 1:

```

public double customEstimate(State state, Condition goal, Action 1
    a, double currentHeuristic) {
    super.setGoal(goal);
    this.expandRelaxedPlanningGraph(state);

    double heuristicValue = currentHeuristic; 2
    3
    4
    5
    6
    7
    8
    9
    10
    11
    12
    13
    14
    15
    16
    17
    18

    if(a.getName().contains("move"))
        heuristicValue -= 1;
    else if (a.getName().contains("fill"))
        heuristicValue -= 0.5;
    else if (a.getName().contains("workstation"))
        heuristicValue -= 0.8;
    else if (a.getName().contains("load"))
        heuristicValue -= 0.7;
    else if (a.getName().equals("empty_box_in_loc"))
        heuristicValue += 0.5;
    return super.isGoalReachable() ? heuristicValue : Integer.
        MAX_VALUE;
}

```

Con il relativo risultato:

```

problem instantiation done successfully (118 actions, 57 fluents 1
)
2
* Starting A* Search with MAN_SEV_Heuristic 3
* A* search succeeded 4
5
found plan as follows: 6
7
00: (      fill_box_in_warehouse robot box3 screw warehouse) 8
01: (      put_up_box_from_location robot box3 warehouse) 9
02: (      move_to_location robot warehouse location1) 10
03: (      move_to_location robot location1 location2) 11
04: (      enter_workstation robot location2 workstation3) 12
05: (      put_down_box_in_workstation robot box3 workstation3) 13
06: (empty_box_in_workstation robot box3 workstation3 screw) 14
07: (      exit_workstation robot location2 workstation3) 15
08: (      move_to_location robot location2 location1) 16
09: (      move_to_location robot location1 warehouse) 17

```

```

10: (      fill_box_in_warehouse robot box1 bolt warehouse) 18
11: (      put_up_box_from_location robot box1 warehouse) 19
12: (      move_to_location robot warehouse location1) 20
13: (      move_to_location robot location1 location2) 21
14: (      enter_workstation robot location2 workstation3) 22
15: (      put_down_box_in_workstation robot box1 workstation3) 23
16: ( empty_box_in_workstation robot box1 workstation3 bolt) 24
17: (      exit_workstation robot location2 workstation3) 25
18: (      move_to_location robot location2 location1) 26
19: (      move_to_location robot location1 warehouse) 27
20: (      fill_box_in_warehouse robot box2 bolt warehouse) 28
21: (      put_up_box_from_location robot box2 warehouse) 29
22: (      move_to_location robot warehouse location1) 30
23: (      move_to_location robot location1 location2) 31
24: (      enter_workstation robot location2 workstation2) 32
25: (      put_down_box_in_workstation robot box2 workstation2) 33
26: ( empty_box_in_workstation robot box2 workstation2 bolt) 34
35
time spent:          0.07 seconds parsing 36
                  0.09 seconds encoding 37
                  6.17 seconds searching 38
                  6.32 seconds total time 39
40
memory used:        0.52 MBytes for problem representation 41
                  144.11 MBytes for searching 42
                  144.62 MBytes total 43

```

- **Variante 2:** si tratta di una versione più raffinata della precedente in cui si tengono in considerazione più possibili azioni, facendo variare anche il peso ad esse associato

```

public double customEstimate(State state, Condition goal, Action 1
    a, double currentHeuristic) {
    super.setGoal(goal); 2
    this.expandRelaxedPlanningGraph(state); 3
4
    double heuristicValue = currentHeuristic; 5
6
    if(a.getName().contains("move")) 7
        heuristicValue -= 1; 8
    else if (a.getName().contains("fill")) 9
        heuristicValue -= 0.5; 10
    else if (a.getName().contains("workstation")) 11

```

```

        heuristicValue -= 0.8;
    else if (a.getName().equals("empty_box_in_workstation"))
        heuristicValue -= 1.5;
    else if (a.getName().equals("empty_box_in_location"))
        heuristicValue += 0.8;
    else if (a.getName().equals("put_down_box_in_location"))
        heuristicValue += 0.5;
    return super.isGoalReachable() ? heuristicValue : Integer.
        MAX_VALUE;
}

```

Con il relativo risultato:

```

* Starting A* Search with MAN_SEV_Heuristic
* A* search succeeded

found plan as follows:

00: (      fill_box_in_warehouse robot box2 bolt warehouse)
01: (      put_up_box_from_location robot box2 warehouse)
02: (      move_to_location robot warehouse location1)
03: (      move_to_location robot location1 location2)
04: (      enter_workstation robot location2 workstation3)
05: (      put_down_box_in_workstation robot box2 workstation3)
06: (      empty_box_in_workstation robot box2 workstation3 bolt)
07: (      exit_workstation robot location2 workstation3)
08: (      move_to_location robot location2 location1)
09: (      move_to_location robot location1 warehouse)
10: (      fill_box_in_warehouse robot box1 bolt warehouse)
11: (      fill_box_in_warehouse robot box3 screw warehouse)
12: (      put_up_box_from_location robot box1 warehouse)
13: (      move_to_location robot warehouse location1)
14: (      move_to_location robot location1 location2)
15: (      enter_workstation robot location2 workstation2)
16: (      put_down_box_in_workstation robot box1 workstation2)
17: (      empty_box_in_workstation robot box1 workstation2 bolt)
18: (      exit_workstation robot location2 workstation2)
19: (      move_to_location robot location2 location1)
20: (      move_to_location robot location1 warehouse)
21: (      put_up_box_from_location robot box3 warehouse)
22: (      move_to_location robot warehouse location1)
23: (      move_to_location robot location1 location2)

```

24: (	enter_workstation	robot	location2	workstation3)	30
25: (	put_down_box_in_workstation	robot	box3	workstation3)	31
26: (empty_box_in_workstation	robot	box3	workstation3	screw)	32
					33
time spent:	0.05 seconds	parsing			34
	0.09 seconds	encoding			35
	4.48 seconds	searching			36
	4.62 seconds	total time			37
					38
memory used:	0.52 MBytes	for problem representation			39
	33.40 MBytes	for searching			40
	33.92 MBytes	total			41

- **Variante 3:** precedentemente descritta nel paragrafo 2.1.2, variante definitiva e qui non riportata per semplicità.

problem instantiation done successfully (118 actions, 57 fluents	1					
)	2					
* Starting A* Search with MAN_SEV_Heuristic	3					
* A* search succeeded	4					
	5					
found plan as follows:	6					
	7					
00: (	fill_box_in_warehouse	robot	box1	bolt	warehouse)	8
01: (	put_up_box_from_location	robot	box1	warehouse)	9	
02: (	move_to_location	robot	warehouse	location1)	10	
03: (	move_to_location	robot	location1	location2)	11	
04: (	enter_workstation	robot	location2	workstation2)	12	
05: (	put_down_box_in_workstation	robot	box1	workstation2)	13	
06: (	empty_box_in_workstation	robot	box1	workstation2	bolt)	14
07: (	exit_workstation	robot	location2	workstation2)	15	
08: (	move_to_location	robot	location2	location1)	16	
09: (	move_to_location	robot	location1	warehouse)	17	
10: (	fill_box_in_warehouse	robot	box3	bolt	warehouse)	18
11: (	put_up_box_from_location	robot	box3	warehouse)	19	
12: (	move_to_location	robot	warehouse	location1)	20	
13: (	move_to_location	robot	location1	location2)	21	
14: (	enter_workstation	robot	location2	workstation3)	22	
15: (	put_down_box_in_workstation	robot	box3	workstation3)	23	
16: (	empty_box_in_workstation	robot	box3	workstation3	bolt)	24
17: (	exit_workstation	robot	location2	workstation3)	25	

```

18: (           move_to_location robot location2 location1) 26
19: (           move_to_location robot location1 warehouse) 27
20: (   fill_box_in_warehouse robot box2 screw warehouse) 28
21: (           put_up_box_from_location robot box2 warehouse) 29
22: (           move_to_location robot warehouse location1) 30
23: (           move_to_location robot location1 location2) 31
24: (           enter_workstation robot location2 workstation3) 32
25: (   put_down_box_in_workstation robot box2 workstation3) 33
26: (empty_box_in_workstation robot box2 workstation3 screw) 34
35

time spent:      0.07 seconds parsing 36
                  0.08 seconds encoding 37
                  4.30 seconds searching 38
                  4.45 seconds total time 39
40

memory used:    0.52 MBytes for problem representation 41
                  17.83 MBytes for searching 42
                  18.34 MBytes total 43

```

Ne segue il confronto:

<b>Planner</b>	<b>N° Azioni</b>	<b>Tempo totale</b>	<b>Spazio totale</b>
EHC_FF_heuristic	118	0.24 sec	0.54 mbytes
A*_AS_heuristic	118	2.00 sec	14.21 mbytes
A*_FF_heuristic	118	2.82 sec	15.88 mbytes
-	-	-	-
A*_CustomHeuristic1	118	6.32 sec	144.62 mbytes
A*_CustomHeuristic2	118	4.62 sec	33.92 mbytes
A*_CustomHeuristic3	118	4.45 sec	18.34 mbytes

Si è deciso a questo punto di proseguire con la seconda istanza.

## 2.3 Istanza 2

L'Istanza 2 rappresenta un'evoluzione della prima, introducendo maggiore complessità attraverso l'aggiunta di nuovi elementi e vincoli legati alla capacità di carico del robot.

In particolare ogni robot è dotato di un **carrier**, che ha una capacità massima di carico specificata dal numero di **slot** presenti al suo interno. Questa capacità limita il numero di scatole che un robot può trasportare contemporaneamente nel carrello. La capacità può variare da un robot all'altro.

### 2.3.1 Domain - Types

Come accennato, la seconda istanza presenta una maggiore difficoltà con l'aggiunta di diversi elementi, di seguito il codice del **Domain.pddl**, in particolare la sezione **Types** aggiornata:

```
(:types  
    location 1  
    box       2  
    content   3  
    robot     4  
    workstation 5  
    carrier   6  
    slot      7  
)  
;un carrier puo avere piu slot, ogni slot indica una locazione 8  
;disponibile per posizionare un box sul carrier. 9
```

### 2.3.2 Domain - Predicates

Segue la sezione del file **Predicates**:

```
(:predicates  
    (at_loc ?obj - (either robot workstation box content) ?loc - 1  
        location)  
    (is_empty ?box - box) 2  
    (filled ?box - box ?cont - content) 3  
    (at_ws ?obj - (either robot box content) ?ws - workstation) 4  
    (connected ?loc1 ?loc2 - location) 5  
    (is_warehouse ?loc - location) 6  
    ;nuovi prediciati per Istanza 2 7  
    (carrying ?carrier - carrier ?box - box) 8  
    (free ?slot - slot) 9  
    (attached ?rob - robot ?carrier - carrier) 10  
)
```

```
(holds ?carrier - carrier ?slot - slot)  
)
```

12  
13

Si è deciso di descrivere solamente i predicati specifici per la nuova istanza, per le spiegazioni dei predicati già presenti nell'**Istanza 1** si rimanda al paragrafo 1.1.2.

- **carrying** (nuovo per l'Istanza 2): Questo predicato indica che un carrier specifico (**?carrier**) sta trasportando una scatola specifica (**?box**). È introdotto per gestire la capacità di carico dei robot, assicurando che le scatole siano correttamente assegnate ai carrier per la consegna.
- **free** (nuovo per l'Istanza 2): Questo predicato indica che uno slot specifico (**?slot**) all'interno di un carrier è libero, ovvero disponibile per trasportare una scatola. Serve a gestire la capacità interna del carrier, monitorando quali spazi sono ancora utilizzabili.
- **attached** (nuovo per l'Istanza 2): Questo predicato indica che un carrier specifico (**?carrier**) è agganciato a un robot (**?rob**). È cruciale per assicurarsi che un carrier sia collegato a un robot prima che il robot possa muoverlo o eseguire operazioni di carico/scarico.
- **holds** (nuovo per l'Istanza 2): Questo predicato indica che un carrier specifico (**?carrier**) contiene uno slot specifico (**?slot**). Questo predicato è utile per tracciare la gestione degli slot all'interno di un carrier, monitorando la struttura interna del sistema di trasporto del robot.

### 2.3.3 Domain - Actions

La sezione **Actions** presenta invece:

- **move\_to\_location**: il robot si sposta da una location a un'altra. Questa azione consente al robot di muoversi tra due posizioni adiacenti, escludendo la possibilità di spostarsi verso il magazzino centrale (warehouse) se il robot ha un carrier con almeno uno slot occupato (condizione vincolante dalle specifiche del problema).

```
(:action move_to_location
  :parameters (?rob - robot ?from - location ?carrier -
    carrier)
  :precondition (and
    (at_loc ?rob ?from)
    (connected ?from ?to)
    (attached ?rob ?carrier)
    (not (is_warehouse ?to)))
  )
  :effect (and (not (at_loc ?rob ?from)) (at_loc ?rob ?to)))
)
```

A differenza dell'omonima azione presente nel domain file dell'istanza precedente, in questo caso, è necessario che nelle precondizioni venga specificato che il carrello sia collegato al robot nello spostamento.

- **move\_to\_warehouse**: Il robot si sposta dalla sua posizione attuale verso il magazzino centrale (warehouse). Questa azione permette al robot di tornare al magazzino centrale solo se tutti gli slot del carrier sono liberi.

```
(:action move_to_warehouse
  :parameters (?rob - robot ?from - location ?carrier -
    carrier)
  :precondition (and
    (at_loc ?rob ?from)
    (connected ?from ?to)
    (attached ?rob ?carrier)
    (is_warehouse ?to)
    (forall (?slot - slot)
      (and
        (holds ?carrier ?slot)
        (free ?slot)
      )
    )))
  :effect (and (not (at_loc ?rob ?from)) (at_loc ?rob ?to)))
)
```

Rispetto all'azione standard di spostamento tra location, **move\_to\_location**, in questo caso, nelle precondizioni, si specifica che:

- La destinazione (to) deve essere il magazzino centrale (warehouse).
- Tutti gli slot del carrier devono essere liberi (free).

- **enter\_workstation**: Il robot entra in una stazione di lavoro. Questa azione permette al robot di spostarsi dalla sua posizione attuale alla stazione di lavoro specificata, purché si trovi già nella stessa location della stazione di lavoro e abbia il carrier collegato.

```
(:action enter_workstation
  :parameters (?rob - robot ?loc - location ?ws - workstation
               ?carrier - carrier)
  :precondition (and
    (at_loc ?ws ?loc)
    (at_loc ?rob ?loc)
    (not (at_ws ?rob ?ws))
    (attached ?rob ?carrier)
    )
  :effect (and (not (at_loc ?rob ?loc)) (at_ws ?rob ?ws)))
)
  1
  2
  3
  4
  5
  6
  7
  8
  9
 10
```

- **exit\_workstation**: Il robot esce da una stazione di lavoro. Questa azione consente al robot di lasciare la stazione di lavoro e tornare alla location generale in cui si trova la stazione purché abbia il carrier collegato.

```
(:action exit_workstation
  :parameters (?rob - robot ?loc - location ?ws - workstation ?
               carrier - carrier)
  :precondition (and
    (at_loc ?ws ?loc)
    (not (at_loc ?rob ?loc))
    (at_ws ?rob ?ws)
    (attached ?rob ?carrier)
    )
  :effect (and (not (at_ws ?rob ?ws)) (at_loc ?rob ?loc)))
)
  1
  2
  3
  4
  5
  6
  7
  8
  9
 10
```

- **put\_down\_box\_in\_workstation:** Il robot deposita una scatola presso una stazione di lavoro. Questa azione permette al robot di lasciare una scatola che sta trasportando nella stazione di lavoro specificata, liberando così uno slot del carrier.

```

(:action put_down_box_in_workstation
  :parameters (?rob - robot ?box - box ?ws - workstation ?
               carrier - carrier ?slot - slot)
  :precondition (and
    (at_ws ?rob ?ws)
    (carrying ?carrier ?box)
    (attached ?rob ?carrier)
    (holds ?carrier ?slot)
    (not (free ?slot))
    )
  :effect (and
    (not (carrying ?carrier ?box))
    (free ?slot)
    (at_ws ?box ?ws))
  )
)

```

Analizziamo i punti chiave dell'azione.

– **precondition:**

- \* Il robot deve trovarsi presso la stazione di lavoro (ws).
- \* Il carrier deve trasportare la scatola (box).
- \* Il carrier deve essere collegato al robot.
- \* Lo slot del carrier deve contenere la scatola e non essere libero (not (free ?slot)).

– **effects:**

- \* Il carrier non trasporta più la scatola (box).
- \* Lo slot del carrier è ora libero (free).
- \* La scatola si trova ora presso la stazione di lavoro (ws).

L'esecuzione di questa azione porta il robot a depositare una singola scatola dunque liberare uno slot del carrello; qualora si vogliano scaricare più box presso una stessa workstation sarà necessario ripetere l'azione più volte.

- **put\_down\_box\_in\_location:** Il robot deposita una scatola nella location in cui si trova. Questa azione consente al robot di lasciare una scatola trasportata in una location specifica, liberando lo slot nel carrier. La definizione dell'azione è analoga alla precedente e, per semplicità, non la si riporta.

- **load\_box\_from\_workstation:** Il robot solleva una scatola dalla stazione di lavoro in cui si trova. Questa azione permette al robot di caricare una scatola dalla stazione di lavoro per posizionarla nel carrier, occupando uno slot disponibile.

```

(:action load_box_from_workstation
  :parameters (?rob - robot ?box - box ?ws - workstation ?
               carrier - carrier ?slot - slot)
  :precondition (and
    (at_ws ?box ?ws)
    (at_ws ?rob ?ws)
    (attached ?rob ?carrier)
    (holds ?carrier ?slot)
    (free ?slot)
  )
  :effect (and
    (carrying ?carrier ?box)
    (not (free ?slot))
    (not (at_ws ?box ?ws)))
  )
)

```

Analizziamo i punti chiave dell'azione.

– **precondition:**

- \* La scatola (box) deve trovarsi presso la stazione di lavoro (ws).
- \* Il robot deve trovarsi presso la stazione di lavoro (ws).
- \* Il carrier deve essere collegato al robot.
- \* Lo slot del carrier deve essere libero (free).

– **effects:**

- \* Il carrier ora trasporta la scatola (box).
- \* Lo slot del carrier non è più libero (not (free ?slot)).
- \* La scatola non si trova più presso la stazione di lavoro (not (at\_ws ?box ?ws)).

- **load\_box\_from\_location:** Il robot solleva una scatola dalla location in cui si trova. Questa azione consente al robot di caricare una scatola presente nella location specificata per posizionarla nel carrier, occupando uno slot disponibile. La definizione dell'azione è analoga alla precedente e, per semplicità, non la si riporta.

- **empty\_box\_in\_workstation**: Il robot svuota una scatola presso una stazione di lavoro. Questa azione permette al robot di rimuovere il contenuto di una scatola e depositarlo nella stazione di lavoro, lasciando la scatola vuota.

```

(:action empty_box_in_workstation
  :parameters (?rob - robot ?box - box ?ws - workstation ?con
               - content ?carrier - carrier)
  :precondition (and
    (at_ws ?rob ?ws)
    (at_ws ?box ?ws)
    (filled ?box ?con)
    (attached ?rob ?carrier)
    )
  :effect (and
    (not (filled ?box ?con))
    (is_empty ?box)
    (at_ws ?con ?ws))
  )
)

```

Analizziamo i punti chiave dell'azione.

– **precondition**:

- \* Il robot deve trovarsi presso la stazione di lavoro (ws).
- \* La scatola deve trovarsi presso la stazione di lavoro (ws).
- \* La scatola deve essere riempita con il contenuto specificato (con).
- \* Il carrier deve essere collegato al robot (per garantire la consistenza operativa).

– **effects**:

- \* La scatola non è più riempita con il contenuto (con).
- \* La scatola diventa vuota (is\_empty).
- \* Il contenuto si trova ora presso la stazione di lavoro (ws).

- **empty\_box\_in\_location**: Il robot svuota una scatola nella location in cui si trova. Questa azione permette al robot di rimuovere il contenuto di una scatola e lasciarlo nella location specificata, rendendo la scatola vuota. La definizione dell'azione è analoga alla precedente e, per semplicità, non la si riporta.

- **fill\_box\_in\_workstation:** Il robot riempie una scatola vuota presso una stazione di lavoro, utilizzando il contenuto disponibile nella stessa stazione. Questa azione consente di riempire una scatola vuota prelevando il contenuto direttamente dalla stazione di lavoro.

```

(:action fill_box_in_workstation
  :parameters (?rob - robot ?box - box ?ws - workstation ?con
  - content ?carrier - carrier)
  :precondition (and
    (at_ws ?rob ?ws)
    (at_ws ?box ?ws)
    (at_ws ?con ?ws)
    (attached ?rob ?carrier)
    (is_empty ?box)
  )
  :effect (and
    (not (at_ws ?con ?ws))
    (filled ?box ?con)
    (not (is_empty ?box)))
)

```

– **precondition:**

- \* Il robot deve trovarsi presso la stazione di lavoro (ws).
- \* La scatola deve trovarsi presso la stazione di lavoro (ws).
- \* Il contenuto deve essere presente presso la stazione di lavoro (ws).
- \* La scatola deve essere vuota (is\_empty).
- \* Il carrier deve essere collegato al robot.

– **effects:**

- \* Il contenuto non si trova più presso la stazione di lavoro (ws).
- \* La scatola è ora riempita con il contenuto (filled).
- \* La scatola non è più vuota (not (is\_empty ?box)).

- **fill\_box\_in\_location:** Il robot riempie una scatola vuota nella location in cui si trova, utilizzando il contenuto disponibile nella stessa location. Questa azione consente di riempire una scatola vuota prelevando il contenuto direttamente dalla location specificata. La definizione dell'azione è analoga alla precedente e, per semplicità, non la si riporta.

- **fill\_box\_in\_warehouse**: Il robot riempie una scatola vuota nel magazzino centrale (warehouse) senza consumare alcun oggetto. Questa azione consente al robot di riempire una scatola vuota con un contenuto specifico all'interno del magazzino centrale.

```

(:action fill_box_in_warehouse
  :parameters (?rob - robot ?box - box ?con - content ?loc -
               location ?carrier - carrier)
  :precondition (and
    (is_warehouse ?loc)
    (at_loc ?rob ?loc)
    (at_loc ?box ?loc)
    (at_loc ?con ?loc)
    (is_empty ?box)
    (attached ?rob ?carrier)
    )
  :effect (and
    (filled ?box ?con)
    (not (is_empty ?box)))
  )
)

```

– **precondition:**

- \* La location deve essere il magazzino centrale (warehouse).
- \* Il robot deve trovarsi nel magazzino centrale (loc).
- \* La scatola deve trovarsi nel magazzino centrale (loc).
- \* Il contenuto deve essere presente nel magazzino centrale (loc).
- \* La scatola deve essere vuota (is\_empty).
- \* Il carrier deve essere collegato al robot.

– **effects:**

- \* La scatola è ora riempita con il contenuto (filled).
- \* La scatola non è più vuota (not (is\_empty ?box)).

#### 2.3.4 Problem - Versione semplificata

Proprio come nel task precedente si è deciso di testare quanto progettato dapprima su una istanza di problema più semplice, il cui **Problem.pddl** è il seguente:

```
(define (problem classical_planning_problem)
  (:domain industrial_manufacturing_service)

  (:objects
    robot1 - robot
    carrier1 - carrier
    warehouse location1 location2 - location
    workstation1 workstation2 workstation3 - workstation
    box1 box2 box3 - box
    bolt screw - content
    slot1 slot2 slot3 - slot
  )
  (:init
    ;il robot si trova nella warehouse
    (at_loc robot1 warehouse)

    ;il robot viene collegato ad un carrier
    (attached robot1 carrier1)

    ;si identifica la warehouse
    (is_warehouse warehouse)

    ;le box sono collocate nella warehouse
    (at_loc box1 warehouse)
    (at_loc box2 warehouse)
    (at_loc box3 warehouse)

    ;i materiali sono nella warehouse
    (at_loc bolt warehouse)
    (at_loc screw warehouse)

    ;le box sono vuote
    (is_empty box1)
    (is_empty box2)
    (is_empty box3)
  )
)
```

```

;si allocano le workstation nelle location 38
(at_loc workstation1 location1) 39
(at_loc workstation2 location2) 40
(at_loc workstation3 location2) 41
                                         42
                                         43
;si stabiliscono i collegamenti tra le varie location 44
(connected warehouse location1) 45
(connected location1 location2) 46
(connected location1 warehouse) 47
(connected location2 location1) 48
                                         49
;si imposta la capacita' del carrier 50
(holds carrier1 slot1) 51
(holds carrier1 slot2) 52
(holds carrier1 slot3) 53
                                         54
;il carrier e' vuoto per cui gli slot risultano tutti liberi 55
(free slot1) 56
(free slot2) 57
(free slot3) 58
)
                                         59
                                         60
(:goal 61
  (and 62
    ;almeno una workstation necessita di un materiale (bullone) 63
    (at_ws bolt workstation2) 64
                                         65
    ;almeno una workstation che non necessita di materiali 66
    (not (at_ws bolt workstation1)) 67
    (not (at_ws screw workstation1)) 68
                                         69
    ;almeno una workstation che necessita di pi   materiali ( 70
      bulloni e viti)
    (at_ws bolt workstation3) 71
    (at_ws screw workstation3) 72
  )
)
                                         73
                                         74
)
                                         75

```

Lo **stato iniziale** si presenta quindi con le seguenti caratteristiche:

- L'agente robot (robot1) si trova inizialmente nella warehouse ed è collegato a un carrier (carrier1);
- Le scatole (box1, box2, box3) sono tutte posizionate nella warehouse e sono vuote;
- I materiali, bulloni (bolt) e viti (screw), sono presenti inizialmente nella warehouse;
- La workstation1 è situata nella location1, mentre la workstation2 e la workstation3 si trovano nella location2;
- La rete di connessioni tra le varie location è tale per cui la warehouse è connessa alla location1, la location1 è connessa alla location2, ed entrambe sono collegate alla warehouse;
- Il carrier1 ha una capacità di tre slot (slot1, slot2, slot3), tutti inizialmente liberi.

Mentre il **goal** è il seguente:

- La workstation2 deve ricevere un bullone (bolt);
- La workstation1 non deve ricevere bulloni (bolt) né viti (screw);
- La workstation3 deve ricevere sia bulloni (bolt) che viti (screw).

### 2.3.5 Risultati - Versione semplificata

Seguendo lo stesso schema utilizzato nell'**Istanza 1** definita nel paragrafo 2.2 i risultati sono stati organizzati riportando prima i plan ottenuti tramite i planner build-in e successivamente le soluzioni conseguite con il planner personalizzato e le eventuali versioni dell'euristica personalizzata:

- Enforced Hill Climbing w/ euristica Fast Forward:

```

problem instantiation done successfully (190 actions, 63 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 bolt warehouse carrier) [0]
01: ( load_box_from_loc robot box1 warehouse carrier slot1) [0]
02: (fill_box_in_warehouse robot box2 screw warehouse carrier) [0]
03: ( load_box_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: ( emtv box in ws robot box2 workstation3 screw carrier) [0]
09: (put down box in ws robot box1 workstation3 carrier slot2) [0]
10: ( emtv 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_box_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: ( emtv box in ws robot box3 workstation2 bolt carrier) [0]

time spent:      0.07 seconds parsing
                  0.10 seconds encoding
                  1.81 seconds searching
                  1.97 seconds total time

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

```

Figura 4: *Risultati ottenuti con l'uso del planner EHC*

- A\* Search w/ euristica Adjusted Sum:

```

problem instantiation done successfully (190 actions, 63 fluents)

* Starting A* search with heuristic: AJUSTED_SUM
* Starting A* search
* A* search succeeded

found plan as follows:

00: ( fill_box_in_warehouse robot box2 bolt warehouse carrier) [0]
01: ( fill_box_in_warehouse robot box3 bolt warehouse carrier) [0]
02: ( load_box_from_loc robot box3 warehouse carrier slot2) [0]
03: (fill_box_in_warehouse robot box1 screw warehouse carrier) [0]
04: ( load_box_from_loc robot box2 warehouse carrier slot3) [0]
05: ( load_box_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:      0.11 seconds parsing
                  0.12 seconds encoding
                  3.82 seconds searching
                  4.05 seconds total time

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

```

Figura 5: Risultati ottenuti con l'uso del planner A\* ed euristica Adjusted Sum

- A\* Search w/ euristica Fast Forward:

```

problem instantiation done successfully (190 actions, 63 fluents)

* Starting ASTAR search with FAST_FORWARD heuristic
* ASTAR search succeeded

found plan as follows:

00: (fill_box_in_warehouse robot box1 screw warehouse carrier) [0]
01: ( fill_box_in_warehouse robot box2 bolt warehouse carrier) [0]
02: ( load_box_from_loc robot box2 warehouse carrier slot2) [0]
03: ( load_box_from_loc robot box1 warehouse carrier slot3) [0]
04: ( fill_box_in_warehouse robot box3 bolt warehouse carrier) [0]
05: ( load_box_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:      0.07 seconds parsing
                  0.13 seconds encoding
                  0.96 seconds searching
                  1.16 seconds total time

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

```

Figura 6: *Risultati ottenuti con l'uso del planner A\* ed euristica Fast Forward*

I risultati del planner personalizzato e le relative versioni della funzione euristica sono:

- Variante 1 riportata nel paragrafo 3.1.2:

```
problem instantiation done successfully (190 actions, 63 fluents)

* Starting A* Search with IndustrialProcessHeuristic
* A* search succeeded

found plan as follows:

00: (      fill_box_in_warehouse robot1 box3 screw warehouse carrier1) [0]
01: (      load_box_from_location robot1 box3 warehouse carrier1 slot2) [0]
02: (      fill_box_in_warehouse robot1 box1 bolt warehouse carrier1) [0]
03: (      fill_box_in_warehouse robot1 box2 bolt warehouse carrier1) [0]
04: (      load_box_from_location robot1 box2 warehouse carrier1 slot3) [0]
05: (      load_box_from_location robot1 box1 warehouse carrier1 slot1) [0]
06: (      move_to_location robot1 warehouse location1 carrier1) [0]
07: (      move_to_location robot1 location1 location2 carrier1) [0]
08: (      enter_workstation robot1 location2 workstation2 carrier1) [0]
09: (put_down_box_in_workstation robot1 box2 workstation2 carrier1 slot1) [0]
10: (      empty_box_in_workstation robot1 box2 workstation2 bolt carrier1) [0]
11: (      exit_workstation robot1 location2 workstation2 carrier1) [0]
12: (      enter_workstation robot1 location2 workstation3 carrier1) [0]
13: (put_down_box_in_workstation robot1 box1 workstation3 carrier1 slot3) [0]
14: (      empty_box_in_workstation robot1 box1 workstation3 bolt carrier1) [0]
15: (put_down_box_in_workstation robot1 box3 workstation3 carrier1 slot2) [0]
16: (      empty_box_in_workstation robot1 box3 workstation3 screw carrier1) [0]

time spent:      0.08 seconds parsing
                  0.27 seconds encoding
                  143.84 seconds searching
                  144.19 seconds total time

memory used:    0.78 MBytes for problem representation
                 961.15 MBytes for searching
                 961.93 MBytes total
```

Figura 7: Risultati ottenuti con il planner personalizzato e la prima variante dell'euristica

- Variante 2 riportata nel paragrafo 3.1.2:

```

problem instantiation done successfully (190 actions, 63 fluents)

* Starting A* Search with IndustrialProcessHeuristic
* A* search succeeded

found plan as follows:

00: (      fill_box_in_warehouse robot box3 screw warehouse carrier) [0]
01: (      load_box_from_location 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_box_from_location robot box2 warehouse carrier slot3) [0]
05: (      load_box_from_location robot box1 warehouse carrier slot1) [0]
06: (          move_to_location robot warehouse location1 carrier) [0]
07: (          move_to_location robot location1 location2 carrier) [0]
08: (          enter_workstation robot location2 workstation3 carrier) [0]
09: (put_down_box_in_workstation robot box2 workstation3 carrier slot1) [0]
10: (put_down_box_in_workstation robot box3 workstation3 carrier slot3) [0]
11: (empty_box_in_workstation robot box3 workstation3 screw carrier) [0]
12: (empty_box_in_workstation robot box2 workstation3 bolt carrier) [0]
13: (exit_workstation robot location2 workstation3 carrier) [0]
14: (enter_workstation robot location2 workstation2 carrier) [0]
15: (put_down_box_in_workstation robot box1 workstation2 carrier slot2) [0]
16: (empty_box_in_workstation robot box1 workstation2 bolt carrier) [0]

time spent:      0.07 seconds parsing
                  0.13 seconds encoding
                  211.73 seconds searching
                  211.92 seconds total time

memory used:      0.78 MBytes for problem representation
                  957.24 MBytes for searching
                  958.02 MBytes total

```

Figura 8: Risultati ottenuti con il planner personalizzato e la seconda variante dell'euristica

- Variante 3 vale a dire la versione definitivamente utilizzata:

```

problem instantiation done successfully (190 actions, 63 fluents)

* Starting A* Search with IndustrialProcessHeuristic
* A* search succeeded

found plan as follows:

00: (      fill_box_in_warehouse robot1 box3 screw warehouse carrier1) [0]
01: (      load_box_from_location robot1 box3 warehouse carrier1 slot2) [0]
02: (      fill_box_in_warehouse robot1 box1 bolt warehouse carrier1) [0]
03: (      fill_box_in_warehouse robot1 box2 bolt warehouse carrier1) [0]
04: (      load_box_from_location robot1 box2 warehouse carrier1 slot3) [0]
05: (      load_box_from_location robot1 box1 warehouse carrier1 slot1) [0]
06: (          move_to_location robot1 warehouse location1 carrier1) [0]
07: (          move_to_location robot1 location1 location2 carrier1) [0]
08: (          enter_workstation robot1 location2 workstation3 carrier1) [0]
09: (put_down_box_in_workstation robot1 box2 workstation3 carrier1 slot1) [0]
10: (put_down_box_in_workstation robot1 box3 workstation3 carrier1 slot3) [0]
11: (empty_box_in_workstation robot1 box3 workstation3 screw carrier1) [0]
12: (empty_box_in_workstation robot1 box2 workstation3 bolt carrier1) [0]
13: (exit_workstation robot1 location2 workstation3 carrier1) [0]
14: (enter_workstation robot1 location2 workstation2 carrier1) [0]
15: (put_down_box_in_workstation robot1 box1 workstation2 carrier1 slot2) [0]
16: (empty_box_in_workstation robot1 box1 workstation2 bolt carrier1) [0]

time spent:      0.05 seconds parsing
                  0.10 seconds encoding
                  139.64 seconds searching
                  139.79 seconds total time

memory used:    0.78 MBytes for problem representation
                 886.81 MBytes for searching
                 887.60 MBytes total

```

Figura 9: Risultati ottenuti con il planner personalizzato e la terza variante dell'euristica

In sintesi:

Planner	N° Azioni	Tempo totale	Spazio totale
EHC_FF_heuristic	190	1.97 sec	0.78 mbytes
A*_AS_heuristic	190	4.05 sec	7.86 mbytes
A*_FF_heuristic	190	1.16 sec	1.71 mbytes
-	-	-	-
A*_CustomHeuristic1	190	144.19 sec	961.93 mbytes
A*_CustomHeuristic2	190	211.92 sec	958.02 mbytes
A*_CustomHeuristic3	190	139.79 sec	887.60 mbytes

### 2.3.6 Problem - Versione complessa

In questa seconda versione del problema è stato aggiunto un **secondo robot** in supporto del primo, anch'esso **dotato di un carrier** ma con una capacità di trasporto minore (pari a 2) piuttosto che 3. Le altre caratteristiche, compreso il goal, sono rimaste invariate.

```
(define (problem classical_planning_problem) 1
  (:domain industrial_manufacturing_service) 2

  (:objects 3
    robot1 robot2 - robot
    carrier1 carrier2 - carrier
    warehouse location1 location2 - location
    workstation1 workstation2 workstation3 - workstation
    box1 box2 box3 - box
    bolt screw - content
    slot1 slot2 slot3 slot4 slot5 - slot
  ) 12

  (:init 13
    ;i robot si trovano nella warehouse
    (at_loc robot1 warehouse) 15
    (at_loc robot2 warehouse) 16
    ;ogni robot viene collegato ad un carrier
    (attached robot1 carrier1) 19
    (attached robot2 carrier2) 20
    ;is_warehouse warehouse
    (is_warehouse warehouse) 23
    ;at_loc box1 warehouse
    (at_loc box1 warehouse) 25
    (at_loc box2 warehouse) 26
    (at_loc box3 warehouse) 27
    ;at_loc bolt warehouse
    (at_loc bolt warehouse) 29
    (at_loc screw warehouse) 30
    ;is_empty box1
    (is_empty box1) 32
    (is_empty box2) 33
    (is_empty box3) 34
    ;at_loc workstation1 location1
    (at_loc workstation1 location1) 35
  ) 36
```

```

(at_loc workstation2 location2) 37
(at_loc workstation3 location2) 38
(at_loc workstation3 location2) 39

(at_connected warehouse location1) 40
(at_connected location1 location2) 41
(at_connected location1 warehouse) 42
(at_connected location2 location1) 43
(at_connected location2 location1) 44

; si imposta la capacita' di ciascun carrier 45
; il carrier1 ha capacita' 3 46
(holds carrier1 slot1) 47
(holds carrier1 slot2) 48
(holds carrier1 slot3) 49
; il carrier2 ha capacita' 2 50
(holds carrier2 slot4) 51
(holds carrier2 slot5) 52
(holds carrier2 slot5) 53

;i carrier sono vuoti:gli slot risultano tutti liberi 54
(free slot1) 55
(free slot2) 56
(free slot3) 57
(free slot4) 58
(free slot5) 59
)

(:goal 62
  (:and 63
    ;almeno una workstation necessita di un materiale 64
    (at_ws bolt workstation2) 65
    (at_ws bolt workstation2) 66

    ;almeno una workstation che non necessita di materiali 67
    (not (at_ws bolt workstation1)) 68
    (not (at_ws screw workstation1)) 69
    (not (at_ws screw workstation1)) 70

    ;almeno una workstation che necessita di pi materiali 71
    (at_ws bolt workstation3) 72
    (at_ws screw workstation3) 73
  )
)
)

```

Per semplicità di scrittura si è deciso di non riportare la descrizione del file in quanto avente

una struttura molto simile a quello mostrato nel paragrafo 2.3.4.

### 2.3.7 Risultati - Versione complessa

Come nel caso precedente i risultati sono i seguenti:

- Enforced Hill Climbing w/ euristica Fast Forward:

```
problem instantiation done successfully (342 actions, 77 fluents)

* Starting ENFORCED_HILL_CLIMBING search with FAST_FORWARD heuristic
* ENFORCED_HILL_CLIMBING search succeeded

found plan as follows:

00: (      fill_box_in_warehouse robot1 box1 bolt warehouse carrier1) [0]
01: (      load_box_from_location robot1 box1 warehouse carrier1 slot1) [0]
02: (      fill_box_in_warehouse robot1 box2 screw warehouse carrier1) [0]
03: (      load_box_from_location robot1 box2 warehouse carrier1 slot2) [0]
04: (          move_to_location robot1 warehouse location1 carrier1) [0]
05: (          move_to_location robot1 location1 location2 carrier1) [0]
06: (          enter_workstation robot1 location2 workstation3 carrier1) [0]
07: (put_down_box_in_workstation robot1 box2 workstation3 carrier1 slot1) [0]
08: (empty_box_in_workstation robot1 box2 workstation3 screw carrier1) [0]
09: (      fill_box_in_warehouse robot2 box3 bolt warehouse carrier2) [0]
10: (      load_box_from_location robot2 box3 warehouse carrier2 slot4) [0]
11: (          move_to_location robot2 warehouse location1 carrier2) [0]
12: (          move_to_location robot2 location1 location2 carrier2) [0]
13: (          enter_workstation robot2 location2 workstation2 carrier2) [0]
14: (put_down_box_in_workstation robot1 box1 workstation3 carrier1 slot2) [0]
15: (put_down_box_in_workstation robot2 box3 workstation2 carrier2 slot4) [0]
16: (empty_box_in_workstation robot1 box1 workstation3 bolt carrier1) [0]
17: (empty_box_in_workstation robot2 box3 workstation2 bolt carrier2) [0]

time spent:      0.05 seconds parsing
                0.14 seconds encoding
                1.43 seconds searching
                1.62 seconds total time

memory used:    1.26 MBytes for problem representation
                0.00 MBytes for searching
                1.26 MBytes total
```

Figura 10: *Risultati ottenuti con il planner Enforced Hill Climbing*

- A\* Search w/ euristica Adjusted Sum:

```

problem instantiation done successfully (342 actions, 77 fluents)

* Starting A* search with heuristic: AJUSTED_SUM
* Starting A* search
* A* search succeeded

found plan as follows:

00: (      fill_box_in_warehouse robot1 box1 bolt warehouse carrier1) [0]
01: (      fill_box_in_warehouse robot1 box2 screw warehouse carrier1) [0]
02: (      load_box_from_location robot1 box2 warehouse carrier1 slot1) [0]
03: (      load_box_from_location robot1 box1 warehouse carrier1 slot2) [0]
04: (          move_to_location robot1 warehouse location1 carrier1) [0]
05: (          move_to_location robot1 location1 location2 carrier1) [0]
06: (          enter_workstation robot1 location2 workstation3 carrier1) [0]
07: (put_down_box_in_workstation robot1 box1 workstation3 carrier1 slot1) [0]
08: (put_down_box_in_workstation robot1 box2 workstation3 carrier1 slot2) [0]
09: (    empty_box_in_workstation robot1 box1 workstation3 bolt carrier1) [0]
10: (    empty_box_in_workstation robot1 box2 workstation3 screw carrier1) [0]
11: (      fill_box_in_warehouse robot2 box3 bolt warehouse carrier2) [0]
12: (      load_box_from_location robot2 box3 warehouse carrier2 slot4) [0]
13: (          move_to_location robot2 warehouse location1 carrier2) [0]
14: (          move_to_location robot2 location1 location2 carrier2) [0]
15: (          enter_workstation robot2 location2 workstation2 carrier2) [0]
16: (put_down_box_in_workstation robot2 box3 workstation2 carrier2 slot4) [0]
17: (    empty_box_in_workstation robot2 box3 workstation2 bolt carrier2) [0]

time spent:      0.05 seconds parsing
                0.13 seconds encoding
                2.18 seconds searching
                2.36 seconds total time

memory used:    1.26 MBytes for problem representation
                6.69 MBytes for searching
                7.95 MBytes total

```

Figura 11: Risultati ottenuti con il planner A\* ed euristica Adjusted Sum

- A\* Search w/ euristica Fast Forward:

```

problem instantiation done successfully (342 actions, 77 fluents)

* Starting ASTAR search with FAST_FORWARD heuristic
* ASTAR search succeeded

found plan as follows:

00: (      fill_box_in_warehouse robot1 box1 bolt warehouse carrier1) [0]
01: (      load_box_from_location robot1 box1 warehouse carrier1 slot1) [0]
02: (      fill_box_in_warehouse robot1 box2 screw warehouse carrier1) [0]
03: (      load_box_from_location robot1 box2 warehouse carrier1 slot2) [0]
04: (      fill_box_in_warehouse robot1 box3 bolt warehouse carrier1) [0]
05: (      load_box_from_location robot1 box3 warehouse carrier1 slot3) [0]
06: (      move_to_location robot1 warehouse location1 carrier1) [0]
07: (      move_to_location robot1 location1 location2 carrier1) [0]
08: (      enter_workstation robot1 location2 workstation2 carrier1) [0]
09: (put_down_box_in_workstation robot1 box3 workstation2 carrier1 slot2) [0]
10: (empty_box_in_workstation robot1 box3 workstation2 bolt carrier1) [0]
11: (exit_workstation robot1 location2 workstation2 carrier1) [0]
12: (enter_workstation robot1 location2 workstation3 carrier1) [0]
13: (put_down_box_in_workstation robot1 box2 workstation3 carrier1 slot3) [0]
14: (empty_box_in_workstation robot1 box2 workstation3 screw carrier1) [0]
15: (put_down_box_in_workstation robot1 box1 workstation3 carrier1 slot1) [0]
16: (empty_box_in_workstation robot1 box1 workstation3 bolt carrier1) [0]

time spent:      0.09 seconds parsing
                  0.45 seconds encoding
                  1.81 seconds searching
                  2.35 seconds total time

memory used:    1.26 MBytes for problem representation
                  2.80 MBytes for searching
                  4.05 MBytes total

```

Figura 12: Risultati ottenuti con il planner A\* ed euristica Fast Forward

L'implementazione dell'algoritmo A\* con l'euristica personalizzata su questa istanza di problema causa un elevato consumo di memoria, generando un errore dovuto all'esaurimento delle risorse. Per gestire questo problema, si è introdotto un **cutoff** che limita la profondità della ricerca, consentendo all'algoritmo di interrompere l'esplorazione ad una specifica profondità. In questo modo, si evita l'errore di memoria esaurita, bilanciando l'efficacia dell'euristica con le risorse disponibili. Di seguito viene riportata la modifica all'algoritmo.

L'intestazione del metodo A\* cambia come segue:

```
public Plan customAstar(Problem problem, int maxDepth) throws  
    ProblemNotSupportedException
```

Si osservi come uno dei parametri passati all'algoritmo sia un valore intero, **maxDepth**, utilizzato per limitare la profondità della ricerca (cutoff).

L'unica altra modifica rilevante, rispetto al metodo riportato nella sezione precedente, è la seguente, inserita all'interno del ciclo while che regola il processo di ricerca:

```
// We start the search  
while (!open.isEmpty() && plan==null && time < timeout) {  
    //We pop the first node in the pending list open  
    final Node current = open.poll();  
    close.add(current);  
  
    // Check if the depth of the current node exceeds the maximum  
    // allowed depth  
    if (current.getDepth() > maxDepth) {  
        continue; // Skip expanding this node and move to the next  
    }  
    ...  
}
```

L'istruzione fornita controlla se la profondità del nodo corrente ha superato una profondità massima predefinita (**maxDepth**) e, se lo ha fatto, salta l'espansione di quel nodo e passa al successivo.

All'interno del metodo **solve** del planner personalizzato sarà necessario impostare la profondità massima. Di seguito un esempio:

```
int depth = 90;  
final Plan plan = this.customAstar(problem, depth);
```

Per determinare il valore ottimale di profondità rispetto alla quale proseguire la ricerca, sono stati condotti diversi esperimenti, testando vari valori di profondità. Questi tentativi hanno permesso di individuare un compromesso efficace tra l'accuratezza della soluzione e il consumo di risorse computazionali, evitando sia un'esplorazione troppo limitata che un eccessivo utilizzo di memoria e tempo di calcolo.

- **A\* Search con Cutoff = 5 w/ euristica personalizzata:** si ricade in una situazione particolare rispetto alle precedenti. **In questo caso settando il Cutoff della ricerca a 5 il planner non è riuscito a trovare un plan per la risoluzione del problema:**

```
problem instantiation done successfully (342 actions, 77 fluents)

* Starting A* Search with IndustrialProcessHeuristic
* A* search failed

no plan found

time spent:      0.15 seconds parsing
                 0.70 seconds encoding
                 0.00 seconds searching
                 0.84 seconds total time

memory used:     1.26 MBytes for problem representation
                 0.00 MBytes for searching
                 1.26 MBytes total
```

Figura 13: Fallimento del planner A\* con cutoff = 5

- A\* Search con Cutoff = 20 w/ euristica personalizzata:

```

problem instantiation done successfully (342 actions, 77 fluents)

* Starting A* Search with IndustrialProcessHeuristic
* A* search succeeded

found plan as follows:

00: (      fill_box_in_warehouse robot1 box2 bolt warehouse carrier1) [0]
01: (      load_box_from_location robot1 box2 warehouse carrier1 slot1) [0]
02: (      fill_box_in_warehouse robot1 box1 screw warehouse carrier1) [0]
03: (      load_box_from_location robot1 box1 warehouse carrier1 slot2) [0]
04: (      fill_box_in_warehouse robot1 box3 bolt warehouse carrier1) [0]
05: (      load_box_from_location robot1 box3 warehouse carrier1 slot3) [0]
06: (      move_to_location robot1 warehouse location1 carrier1) [0]
07: (      move_to_location robot1 location1 location2 carrier1) [0]
08: (      enter_workstation robot1 location2 workstation2 carrier1) [0]
09: (put_down_box_in_workstation robot1 box3 workstation2 carrier1 slot2) [0]
10: (empty_box_in_workstation robot1 box3 workstation2 bolt carrier1) [0]
11: (exit_workstation robot1 location2 workstation2 carrier1) [0]
12: (enter_workstation robot1 location2 workstation3 carrier1) [0]
13: (put_down_box_in_workstation robot1 box1 workstation3 carrier1 slot3) [0]
14: (empty_box_in_workstation robot1 box1 workstation3 screw carrier1) [0]
15: (put_down_box_in_workstation robot1 box2 workstation3 carrier1 slot1) [0]
16: (empty_box_in_workstation robot1 box2 workstation3 bolt carrier1) [0]

time spent:    0.08 seconds parsing
               0.16 seconds encoding
               4.02 seconds searching
               4.26 seconds total time

memory used:   1.26 MBytes for problem representation
               56.90 MBytes for searching
               58.15 MBytes total

```

Figura 14: Risultato ottenuto con il planner A\*, euristica personalizzata e cutoff = 20

Per completezza sono stati calcolati i plan utilizzando altri valori per il cutoff quali 60 e 90 ma si è deciso di non riportarli in quanto privi di differenze; si farà unicamente riferimento alle informazioni più importanti nella sintesi finale.

Sintetizzando come fatto fin'ora dunque:

Planner	N° Azioni	Tempo totale	Spazio totale
EHC_FF_heuristic	342	1.62 sec	1.26 mbytes
A*_AS_heuristic	342	2.36 sec	7.95 mbytes
A*_FF_heuristic	342	2.35 sec	4.05 mbytes
-	-	-	-
A*_CustomHeuristic_cutoff=20	342	4.26 sec	58.15 mbytes
A*_CustomHeuristic_cutoff=60	342	4.51 sec	9.83 mbytes
A*_CustomHeuristic_cutoff=90	342	7.67 sec	36.76 mbytes

### 3 Temporal Planning & Robotics

#### 3.1 Temporal Planning

In questa penultima sezione viene descritto come si è modificato il dominio presentato in precedenza per consentire la generazione di una sequenza temporale di azioni con durate specifiche. Un problema di **temporal planning** richiede di trovare una serie di azioni che non solo siano eseguibili da un punto di vista causale (come nella pianificazione classica), ma che rispettino anche un insieme di vincoli temporali relativi alle durate delle azioni lungo una linea temporale senza limiti.

Una **durative action** è una particolare azione che richiede una certa quantità di tempo per essere completata. Questa durata può essere espressa come un valore fisso o come una variabile dipendente da altri fattori. Analogamente alle azioni tradizionali, è possibile specificare degli effetti e delle condizioni, ma nella pianificazione temporale queste ultime vengono definite attraverso il termine “**condition**” al posto di “**precondition**”.

Questa variazione semantica riflette il fatto che una condizione di una **durative action** può essere valida non solo all'inizio, ma anche alla fine o durante tutta la durata dell'azione stessa. In questo ambito l'uso del costrutto "**at start**" seguito da un predicato implica che la condizione debba essere soddisfatta prima che l'azione abbia inizio, mentre il costrutto "**over all**" indica che la condizione deve mantenersi vera dall'inizio alla fine dell'azione. Questi costrutti possono essere utilizzati anche nella definizione degli effetti di un'azione.

Sono riportati, a questo punto, i cambiamenti fatti al **Domain.pddl** definito nel task precedente in modo tale da rispecchiare quanto appena descritto. Per quanto riguarda l'intestazione:

```
(define (domain industrial_manufacturing_service_durative_actions) 1
      (:requirements :strips :typing :durative-actions :equality :numeric 2
                     -fluents :negative-preconditions)
      (:types 3
              location box content robot workstation carrier
      ) 4
      5
      6
      7)
```

Per quanto invece concerne la **modellazione della capacità totale** e tenere traccia del **carico del carrier** sono state utilizzate due funzioni. Le funzioni in PDDL sono utilizzate per modellare i fluent. Un **fluent** è una variabile che rappresenta una proprietà numerica o una quantità che può cambiare valore durante l'esecuzione di un piano. A differenza dei predicati, che esprimono condizioni booleane (vero o falso), i fluenti hanno valori numerici e vengono usati per rappresentare grandezze che possono variare.

```
(:predicates
  (at_loc ?obj - (either robot workstation box content) ?loc -
            location)
  (is_empty ?box - box)
  (filled ?box - box ?cont - content)
  (at_ws ?obj - (either robot box content) ?ws - workstation)
  (carrying ?carrier - carrier ?box - box)
  (connected ?loc1 ?loc2 - location)
  (is_warehouse ?loc - location)
  (joined ?robot - robot ?carrier - carrier)
)
(:functions
  (capacity ?carrier - carrier)
  (load ?carrier - carrier)
)
```

Nel paragrafo precedente 2.3.2 lo stesso concetto è stato progettato utilizzando i predicati **carrying**, **free**, **attached**, **holds**, come di seguito:

```
(:predicates
  (at_loc ?obj - (either robot workstation box content) ?loc - 1
         location)
  (is_empty ?box - box) 2
  (filled ?box - box ?cont - content) 3
  (at_ws ?obj - (either robot box content) ?ws - workstation) 4
  (connected ?loc1 ?loc2 - location) 5
  (is_warehouse ?loc - location) 6
;nuovi predici per la seconda istanza 7
  (carrying ?carrier - carrier ?box - box) 8
  (free ?slot - slot) 9
  (attached ?rob - robot ?carrier - carrier) 10
  (holds ?carrier - carrier ?slot - slot) 11
)
12
13
```

A proposito delle **azioni** invece, queste hanno subito leggere modifiche per potersi adattare al contesto d'uso delle **durative-actions**. Nel paragrafo precedente 2.3.3 seguivano la forma:

```
(:action move_to_location
  :parameters (?rob - robot ?from ?to - location ?carrier - carrier
  )
  :precondition (and
    (at_loc ?rob ?from)
    (connected ?from ?to)
    (attached ?rob ?carrier)
    (not (is_warehouse ?to)))
  :effect (and
    (not (at_loc ?rob ?from))
    (at_loc ?rob ?to))
  )
) 11
```

Mentre adesso avranno una struttura del tipo:

```
(:durative-action move_to_location
  :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)))
  )
) 12
```

In cui è possibile notare la definizione di nuovi costrutti:

- **Conditions** piuttosto che **Preconditions**;
- **at\_start**: seguito da un predicato, specifica che la condizione espressa deve essere valida prima dell'avvenire dell'azione;
- **at\_end**: seguito da un predicato, specifica che la condizione espressa sarà valida una volta terminata l'azione;
- **over all**: seguito da un predicato, specifica che la condizione espressa deve essere valida prima dell'avvenire dell'azione e per tutta la sua durata.

In una durative action è, inoltre, necessario specificare la durata dell'azione e le condizioni affinché si verifichino i singoli predicati.

### 3.1.1 Problem

Come di consueto in questa sezione si tratterà la risoluzione di un'istanza del problema relativo al dominio fin'ora trattato.

La definizione è la stessa vista nel paragrafo 2.3.4 tranne per la seguente differenza:

```
;----- PRIMA -----
(holds carrier1 slot1)          1
(holds carrier1 slot2)          2
(holds carrier1 slot3)          3
(holds carrier1 slot3)          4
                               5
(free slot1)                  6
(free slot2)                  7
(free slot3)                  8
                               9

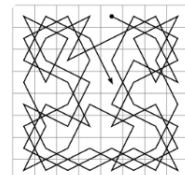
;----- DOPO -----
(= (capacity carrier) 2)        10
(= (load carrier) 0)            11
                               12
```

Si può notare come la capacità iniziale del carrier viene modellata attraverso la funzione **capacity** mentre la quantità di carico viene modellata dalla funzione **load**.

### 3.1.2 Risultati

Per la risoluzione dell'istanza si è fatto uso del framework **planutils**

## AI-Planning/ planutils



General library for setting up linux-based environments for developing, running, and evaluating planners.

20  
Contributors

9  
Used by

3  
Discussions

103  
Stars

30  
Forks



Come planner risolutore si è scelto di utilizzare **LPG-td** poiché in grado di supportare l'utilizzo delle **durative-actions** nonchè di molte delle caratteristiche avanzate di ADL (Action Description Language, estensione del PDDL) inclusi effetti condizionali, precondizioni negative, quantificatori, e connettivi logici complessi, da noi utilizzati nella definizione del domain file. Questo permette in realtà due diverse metodologie di ricerca:

- **speed:** consente di ottenere un plan nel minor tempo possibile. Il comando d'esecuzione è riportato di seguito:

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

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_BOX_FROM_LOCATION ROBOT BOX3 WAREHOUSE CARRIER) [D:1.50; C:1.00]
5.0000: (MOVE_TO_LOCATION ROBOT WAREHOUSE LOCATION1 CARRIER) [D:3.00; C:1.00]
8.0000: (MOVE_TO_LOCATION ROBOT LOCATION1 LOCATION2 CARRIER) [D:3.00; C:1.00]
11.0000: (ENTER_WORKSTATION ROBOT LOCATION2 WORKSTATION2 CARRIER) [D:0.00; C:1.00]
11.0000: (PUT_DOWN_BOX_IN_WORKSTATION ROBOT BOX3 WORKSTATION2 CARRIER) [D:1.50; C:1.00]
12.5000: (EMPTY_BOX_IN_WORKSTATION ROBOT BOX3 WORKSTATION2 BOLT CARRIER) [D:3.50; C:1.00]
14.5000: (LOAD_BOX_FROM_WORKSTATION ROBOT BOX3 WORKSTATION2 CARRIER) [D:1.50; C:1.00]
16.0000: (EXIT_WORKSTATION ROBOT LOCATION2 WORKSTATION2 CARRIER) [D:0.00; C:1.00]
16.0000: (MOVE_TO_LOCATION ROBOT LOCATION2 LOCATION1 CARRIER) [D:3.00; C:1.00]
19.0000: (MOVE_TO_WAREHOUSE ROBOT LOCATION1 WAREHOUSE CARRIER) [D:3.00; C:1.00]
22.0000: (PUT_DOWN_BOX_IN_LOCATION ROBOT BOX3 WAREHOUSE CARRIER) [D:1.50; C:1.00]
22.0000: (FILL_BOX_IN_WAREHOUSE ROBOT BOX2 BOLT WAREHOUSE CARRIER) [D:3.50; C:1.00]
23.5000: (FILL_BOX_IN_WAREHOUSE ROBOT BOX3 SCREW WAREHOUSE CARRIER) [D:3.50; C:1.00]
25.5000: (LOAD_BOX_FROM_LOCATION ROBOT BOX2 WAREHOUSE CARRIER) [D:1.50; C:1.00]
27.0000: (LOAD_BOX_FROM_LOCATION ROBOT BOX3 WAREHOUSE CARRIER) [D:1.50; C:1.00]
28.5000: (MOVE_TO_LOCATION ROBOT WAREHOUSE LOCATION1 CARRIER) [D:3.00; C:1.00]
31.5000: (MOVE_TO_LOCATION ROBOT LOCATION1 LOCATION2 CARRIER) [D:3.00; C:1.00]
34.5000: (ENTER_WORKSTATION ROBOT LOCATION2 WORKSTATION3 CARRIER) [D:0.00; C:1.00]
34.5000: (PUT_DOWN_BOX_IN_WORKSTATION ROBOT BOX3 WORKSTATION3 CARRIER) [D:1.50; C:1.00]
34.5000: (PUT_DOWN_BOX_IN_WORKSTATION ROBOT BOX2 WORKSTATION3 CARRIER) [D:1.50; C:1.00]
36.0000: (EMPTY_BOX_IN_WORKSTATION ROBOT BOX2 WORKSTATION3 BOLT CARRIER) [D:3.50; C:1.00]
36.0000: (EMPTY_BOX_IN_WORKSTATION ROBOT BOX3 WORKSTATION3 SCREW CARRIER) [D:3.50; C:1.00]
```

Solution number: 3  
Total time: 9.77  
Search time: 9.76  
Actions: 23  
Duration: 39.500  
Plan quality: 23.000  
Total Num Flips: 14353

Figura 15: Risultati ottenuti con il planner LPG-td utilizzando l'opzione *speedy*

- **quality**: in questo caso la risoluzione del problema avviene cercando di generare un plan tramite l'uso di un quantitativo di risorse computazionali superiore, si punta ad un miglioramento della soluzione. Il comando per eseguirlo è:

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

```
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_BOX_FROM_LOCATION ROBOT BOX3 WAREHOUSE CARRIER) [D:1.50; C:1.00]
5.0000: (MOVE_TO_LOCATION ROBOT WAREHOUSE LOCATION1 CARRIER) [D:3.00; C:1.00]
8.0000: (MOVE_TO_LOCATION ROBOT LOCATION1 LOCATION2 CARRIER) [D:3.00; C:1.00]
11.0000: (ENTER_WORKSTATION ROBOT LOCATION2 WORKSTATION2 CARRIER) [D:0.00; C:1.00]
11.0000: (PUT_DOWN_BOX_IN_WORKSTATION ROBOT BOX3 WORKSTATION2 CARRIER) [D:1.50; C:1.00]
12.5000: (EMPTY_BOX_IN_WORKSTATION ROBOT BOX3 WORKSTATION2 BOLT CARRIER) [D:3.50; C:1.00]
14.5000: (LOAD_BOX_FROM_WORKSTATION ROBOT BOX3 WORKSTATION2 CARRIER) [D:1.50; C:1.00]
16.0000: (EXIT_WORKSTATION ROBOT LOCATION2 WORKSTATION2 CARRIER) [D:0.00; C:1.00]
16.0000: (MOVE_TO_LOCATION ROBOT LOCATION2 LOCATION1 CARRIER) [D:3.00; C:1.00]
19.0000: (MOVE_TO_WAREHOUSE ROBOT LOCATION1 WAREHOUSE CARRIER) [D:3.00; C:1.00]
22.0000: (PUT_DOWN_BOX_IN_LOCATION ROBOT BOX3 WAREHOUSE CARRIER) [D:1.50; C:1.00]
22.0000: (FILL_BOX_IN_WAREHOUSE ROBOT BOX2 BOLT WAREHOUSE CARRIER) [D:3.50; C:1.00]
23.5000: (FILL_BOX_IN_WAREHOUSE ROBOT BOX3 SCREW WAREHOUSE CARRIER) [D:3.50; C:1.00]
25.5000: (LOAD_BOX_FROM_LOCATION ROBOT BOX2 WAREHOUSE CARRIER) [D:1.50; C:1.00]
27.0000: (LOAD_BOX_FROM_LOCATION ROBOT BOX3 WAREHOUSE CARRIER) [D:1.50; C:1.00]
28.5000: (MOVE_TO_LOCATION ROBOT WAREHOUSE LOCATION1 CARRIER) [D:3.00; C:1.00]
31.5000: (MOVE_TO_LOCATION ROBOT LOCATION1 LOCATION2 CARRIER) [D:3.00; C:1.00]
34.5000: (ENTER_WORKSTATION ROBOT LOCATION2 WORKSTATION3 CARRIER) [D:0.00; C:1.00]
34.5000: (PUT_DOWN_BOX_IN_WORKSTATION ROBOT BOX3 WORKSTATION3 CARRIER) [D:1.50; C:1.00]
34.5000: (PUT_DOWN_BOX_IN_WORKSTATION ROBOT BOX2 WORKSTATION3 CARRIER) [D:1.50; C:1.00]
36.0000: (EMPTY_BOX_IN_WORKSTATION ROBOT BOX2 WORKSTATION3 BOLT CARRIER) [D:3.50; C:1.00]
36.0000: (EMPTY_BOX_IN_WORKSTATION ROBOT BOX3 WORKSTATION3 SCREW CARRIER) [D:3.50; C:1.00]

Solution number: 3
Total time:      9.77
Search time:     9.76
Actions:        23
Duration:       39.500
Plan quality:   23.000
Total Num Flips: 14353
```

Figura 16: Risultati ottenuti con il planner LPG-td utilizzando l'opzione quality

## 3.2 Robotic Planning

L'ultima sezione dell'elaborato è incentrata sulla pianificazione robotica, disciplina che si occupa della progettazione e dello sviluppo di algoritmi e strategie che consentono ai robot autonomi di pianificare e programmare le proprie azioni al fine di raggiungere obiettivi specifici in ambienti dinamici e incerti.

Per pianificare le proprie attività, il robot deve disporre di una rappresentazione accurata dell'ambiente in cui opera e deve essere in grado di elaborare un piano che lo conduca al raggiungimento degli obiettivi prefissati.

In relazione al secondo punto richiesto del terzo task è necessario implementare il problema descritto nella sezione precedente utilizzando ROS2 Planning System (**Plansys2**), una piattaforma open source basata su **ROS** (Robot Operating System 2).



**Plansys2** fornisce strumenti per la pianificazione, l'esecuzione e la gestione delle attività dei robot autonomi, supportando anche la pianificazione temporale. All'interno di Plansys2 sono presenti le cosiddette “**fake actions**”: azioni simulate e rappresentate in modo da permettere al sistema di pianificazione di gestire scenari specifici o di modellare comportamenti desiderati che non necessariamente corrispondono alle azioni reali eseguite dal robot o dall'agente. Queste azioni sono utilizzate per modellare situazioni particolari, come ad esempio il fatto che il robot "rimanga in attesa" in una determinata posizione per un periodo di tempo specifico, e possono essere impiegate per gestire le tempistiche, includendo pause e/o ritardi nell'operatività.

Le **fake actions** appena accennate sono state implementate utilizzando C++ con l'intento di riprodurre le **durative actions** definite nel dominio visto nel paragrafo 2.3. Per non appesantire troppo l'elaborato si è deciso di riportare il codice di una sola **fake actions**:

```
#include <memory>                                     1
#include <algorithm>                                    2
#include <vector>                                       3
#include <string>                                       4
                                                       5
#include "plansys2_executor/ActionExecutorClient.hpp"    6
#include "rclcpp/rclcpp.hpp"                                7
```

```

#include "rclcpp_action/rclcpp_action.hpp"                                8
                                                               9
using namespace std::chrono_literals;                                     10
                                                               11
class EmptyBoxInLocation : public plansys2::ActionExecutorClient          12
{
public:
    EmptyBoxInLocation()                                                 13
        : plansys2::ActionExecutorClient("empty_box_in_location", 250ms) 14
    {
        progress_ = 0.0;                                                 15
    }

    /** Metodo che viene eseguito periodicamente per simulare il      16
        progresso dell'azione: aggiorna lo stato dell'azione e invia     17
        feedback. **/                                                 18

private:
    void do_work()                                                       19
    { std :: vector <std :: string > arguments = get_arguments();           20
        if (progress_ < 1.0) {
            /* Se progress_ e' inferiore a 1.0, viene incrementata di 0.2 e   21
               si invia un feedback con la percentuale attuale e una
               descrizione dell'operazione */
            progress_ += 0.2;
            send_feedback(progress_, "Robot " + arguments [0] + " is           22
                           emptying " + arguments [1] + " in " + arguments [3] + ", the 23
                           box contained some " + arguments [2] + " carring a " +
                           arguments [4]);
        } else {                                                       24
            /* Se progress_ raggiunge 1.0 l'azione e' completata, viene       25
               inviata la notifica di completamento tramite finish() e la
               variabile viene resettata a 0.0. */
            finish(true, progress_, "Robot " + arguments [0] + " is           26
                           emptying " + arguments [1] + " in " + arguments [3] + ", the 27
                           box contained some " + arguments [2] + " carring a " +
                           arguments [4]);
        }
        progress_ = 0.0;                                                 28
        std::cout << std::endl;                                         29
    }
}

```

```

    std::cout << "\r\e[K" << std::flush;
    std::cout << "Robot " + arguments[0] + " is emptying " +
        arguments[1] + " in " + arguments[3] + ", the box
            contained some "+
        arguments[2] + " carrying a "+
        arguments[4] + " . . . [ " << std::min(100.0, progress_ *
            100.0) << "%" ] " <<
    std::flush;
}

/* Variabile privata che tiene traccia dello stato di avanzamento
dell'azione. Viene aggiornata ad ogni ciclo finche' l'azione non
e' completa. */
float progress_;
};

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

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

    rclcpp::spin(node->get_node_base_interface());

    rclcpp::shutdown();

    return 0;
}

```

La classe EmptyBoxInLocation eredita da plansys2::ActionExecutorClient, che è utilizzata per eseguire un'azione specifica nel contesto di un sistema di pianificazione in ROS 2.

Questo task particolare ha richiesto poi la definizione di un file Python di configurazione, definito **plansys2\_msl.py**, che ha il compito di comportarsi come un'interfaccia tra il sistema di pianificazione di alto livello e le componenti di controllo e gestione delle azioni del robot. È riportato di seguito:

```
import os 1
from ament_index_python.packages import get_package_share_directory 2
from launch import LaunchDescription 3
from launch.actions import DeclareLaunchArgument, 4
    IncludeLaunchDescription 5
from launch.launch_description_sources import 6
    PythonLaunchDescriptionSource 7
from launch.substitutions import LaunchConfiguration 8
from launch_ros.actions import Node 9
10
11
def generate_launch_description(): 12
    # Get the launch directory 13
    example_dir = get_package_share_directory(' 14
        industrial_manufacturing_service_robots') 15
    namespace = LaunchConfiguration('namespace') 16
17
    declare_namespace_cmd = DeclareLaunchArgument( 18
        'namespace', 19
        default_value='', 20
        description='Namespace') 21
22
    plansys2_cmd = IncludeLaunchDescription( 23
        PythonLaunchDescriptionSource(os.path.join( 24
            get_package_share_directory('plansys2_bringup'), 25
            'launch', 26
            'plansys2_bringup_launch_monolithic.py')), 27
        launch_arguments={ 28
            'model_file': example_dir + '/pddl/domain.pddl', 29
            'namespace': namespace 30
        }.items()) 31
32
    # Specify the actions 32
    empty_box_in_location_cmd = Node( 33
        package='industrial_manufacturing_service_robots', 34
```

```

executable='empty_box_in_location_action_node',
          35
name='empty_box_in_loc_action_node',
          36
namespace=namespace,
          37
output='screen',
          38
parameters=[])

          39
          40

empty_box_in_workstation_cmd = Node(
    package='industrial_manufacturing_service_robots',
          41
    executable='empty_box_in_workstation_action_node',
          42
    name='empty_box_in_workstation_action_node',
          43
    namespace=namespace,
          44
    output='screen',
          45
    parameters=[])

          46
          47
          48

enter_workstation_cmd = Node(
    package='industrial_manufacturing_service_robots',
          49
    executable='enter_workstation_action_node',
          50
    name='enter_workstation_action_node',
          51
    namespace=namespace,
          52
    output='screen',
          53
    parameters[]) # Create the launch description and populate
          54
          55
          56

exit_workstation_cmd = Node(
    package='industrial_manufacturing_service_robots',
          57
    executable='exit_workstation_action_node',
          58
    name='exit_workstation_action_node',
          59
    namespace=namespace,
          60
    output='screen',
          61
    parameters[])

          62
          63
          64

fill_box_in_location_cmd = Node(
    package='industrial_manufacturing_service_robots',
          65
    executable='fill_box_in_location_action_node',
          66
    name='fill_box_in_location_action_node',
          67
    namespace=namespace,
          68
    output='screen',
          69
    parameters[])

          70
          71
          72

fill_box_in_warehouse_cmd = Node(
    package='industrial_manufacturing_service_robots',
          73
    executable='fill_box_in_warehouse_action_node',
          74
    name='fill_box_in_warehouse_action_node',
          75
    namespace=namespace,
    output='screen',
    parameters[])

```

```

    name='fill_box_in_warehouse_action_node',
    namespace=namespace,
    output='screen',
    parameters=[])

fill_box_in_workstation_cmd = Node(
    package='industrial_manufacturing_service_robots',
    executable='fill_box_in_workstation_action_node',
    name='fill_box_in_workstation_action_node',
    namespace=namespace,
    output='screen',
    parameters=[])

load_box_from_location_cmd = Node(
    package='industrial_manufacturing_service_robots',
    executable='load_box_from_location_action_node',
    name='load_box_from_location_action_node',
    namespace=namespace,
    output='screen',
    parameters=[])

load_box_from_workstation_cmd = Node(
    package='industrial_manufacturing_service_robots',
    executable='load_box_from_workstation_action_node',
    name='load_box_from_workstation_action_node',
    namespace=namespace,
    output='screen',
    parameters=[])

move_to_location_cmd = Node(
    package='industrial_manufacturing_service_robots',
    executable='move_to_location_action_node',
    name='move_to_location_action_node',
    namespace=namespace,
    output='screen',
    parameters=[])

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

```

```

    name='move_to_warehouse_action_node',
    namespace=namespace,
    output='screen',
    parameters=[])

117
118
119
120
121
122

put_down_box_in_location_cmd = Node(
    package='industrial_manufacturing_service_robots',
    executable='put_down_box_in_location_action_node',
    name='put_down_box_in_location_action_node',
    namespace=namespace,
    output='screen',
    parameters=[])

123
124
125
126
127
128
129
130
131

put_down_box_in_workstation_cmd = Node(
    package='industrial_manufacturing_service_robots',
    executable='put_down_box_in_workstation_action_node',
    name='put_down_box_in_workstation_action_node',
    namespace=namespace,
    output='screen',
    parameters=[])

132
133
134
135
136
137
138
139

ld = LaunchDescription()

140
141

ld.add_action(declare_namespace_cmd)

142
143

# Declare the launch options
ld.add_action(plansys2_cmd)

144
145
146

ld.add_action(empty_box_in_location_cmd)
ld.add_action(empty_box_in_workstation_cmd)
ld.add_action(enter_workstation_cmd)
ld.add_action(exit_workstation_cmd)
ld.add_action(fill_box_in_location_cmd)
ld.add_action(fill_box_in_warehouse_cmd)
ld.add_action(fill_box_in_workstation_cmd)
ld.add_action(load_box_from_location_cmd)
ld.add_action(load_box_from_workstation_cmd)
ld.add_action(move_to_location_cmd)
ld.add_action(move_to_warehouse_cmd)

147
148
149
150
151
152
153
154
155
156
157

```

ld.add_action(put_down_box_in_location_cmd)	158
ld.add_action(put_down_box_in_workstation_cmd)	159
	160
return ld	161

Il launch file in ROS2 serve a configurare e avviare uno o più nodi ROS e altre risorse associate (come parametri, argomenti, o log) in modo coordinato. Nello specifico **Node** è una classe fornita dalla libreria **launch\_ros** di ROS2 ed è utilizzata per definire un nodo che sarà avviato quando si lancia il file. Questo nodo esegue un processo ROS2 specifico. In particolare, nei parametri è necessario specificare il package ROS 2 da cui verrà lanciato l'eseguibile e l'eseguibile che verrà mandato in esecuzione all'interno di quel package.

Per ottenere il risultato desiderato si ha la necessità di gestire due terminali, e dare infine in input uno dei plan precedentemente generati nel paragrafo 3.1.2:

- Nel primo terminale eseguiremo **PlanSys2** lanciando la seguente sequenza di comandi:

1. `colcon build --symlink-install`: utilizzato per compilare i file C++;
2. `source install/setup.bash`.
3. `ros2 launch industrial_manufacturing_service_robots plansys2_msl.py`

In particolare l'ultimo comando avvia uno o più nodi ROS2 definiti nel file di lancio **plansys2\_msl.py** all'interno del pacchetto **industrial\_manufacturing\_service\_robots**.

```
aiguy@vm-ubu22: ~/Desktop/ProgettoIA_Curcio_Gullone/task3_temporal_planning_robots/instance_3_2/industrial_manufacturing... aiguy@vm-ubu22: ~/Desktop/ProgettoIA_Curcio_Gullone/task3_temporal_planning_robots/instance_3_2/industrial_manufacturing...
[plansys2_node-1] [INFO] [1724862724.379224476] [executor]: Action FILL_BOX_IN_WAREHOUSE timeout percentage -1.000000
[plansys2_node-1] [INFO] [1724862724.755841595] [executor]: Action LOAD_BOX_FROM_LOCATION timeout percentage -1.000000
[plansys2_node-1] [INFO] [1724862725.446636867] [executor]: Action MOVE_TO_LOCATION timeout percentage -1.000000
[plansys2_node-1] [INFO] [1724862725.682271152] [executor]: Action MOVE_TO_LOCATION timeout percentage -1.000000
[plansys2_node-1] [INFO] [1724862725.754699231] [executor]: Action ENTER_WORKSTATION timeout percentage -1.000000
[plansys2_node-1] [INFO] [1724862725.943879538] [executor]: Action PUT_DOWN_BOX_IN_WORKSTATION timeout percentage -1.000000
[plansys2_node-1] [INFO] [1724862726.044493998] [executor]: Action EMPTY_BOX_IN_WORKSTATION timeout percentage -1.000000
[plansys2_node-1] [INFO] [1724862726.175040069] [executor]: Action LOAD_BOX_FROM_WORKSTATION timeout percentage -1.000000
[plansys2_node-1] [INFO] [1724862726.414747054] [executor]: Action MOVE_TO_WORKSTATION timeout percentage -1.000000
[plansys2_node-1] [INFO] [1724862726.619479253] [executor]: Action MOVE_TO_LOCATION timeout percentage -1.000000
[plansys2_node-1] [INFO] [1724862726.917366214] [executor]: Action MOVE_TO_WAREHOUSE timeout percentage -1.000000
[plansys2_node-1] [INFO] [1724862726.973662144] [executor]: Action PUT_DOWN_BOX_IN_LOCATION timeout percentage -1.000000
[plansys2_node-1] [INFO] [1724862727.108316656] [executor]: Action FILL_BOX_IN_WAREHOUSE timeout percentage -1.000000
[plansys2_node-1] [INFO] [1724862727.682857777] [executor]: Action LOAD_BOX_FROM_LOCATION timeout percentage -1.000000
[plansys2_node-1] [INFO] [1724862727.923593583] [executor]: Action LOAD_BOX_IN_WAREHOUSE timeout percentage -1.000000
[plansys2_node-1] [INFO] [1724862728.173015560] [executor]: Action MOVE_TO_LOCATION timeout percentage -1.000000
[plansys2_node-1] [INFO] [1724862728.423283590] [executor]: Action MOVE_TO_LOCATION timeout percentage -1.000000
[plansys2_node-1] [INFO] [1724862728.703648557] [executor]: Action MOVE_TO_WORKSTATION timeout percentage -1.000000
[plansys2_node-1] [INFO] [1724862728.814025322] [executor]: Action PUT_DOWN_BOX_IN_WORKSTATION timeout percentage -1.000000
[plansys2_node-1] [INFO] [1724862728.941939115] [executor]: Action LOAD_BOX_IN_WORKSTATION timeout percentage -1.000000
[plansys2_node-1] [INFO] [1724862729.159806695] [executor]: Action EMPTY_BOX_IN_WORKSTATION timeout percentage -1.000000
[plansys2_node-1] [INFO] [1724862729.269369501] [executor]: Action EMPTY_BOX_IN_WORKSTATION timeout percentage -1.000000
[plansys2_node-1] [WARN] [1724862729.403617613] [rcl.logging_rosout]: Publisher already registered for provided node name. If this is due to multiple nodes with the same name then all logs for that logger name will go out over the existing publisher. As soon as any node with that name is destroyed it will unregister the publisher, preventing any further logs for that name from being published on the rosout topic.
[plansys2_node-1] [WARN] [1724862730.655354980] [rcl.logging_rosout]: Publisher already registered for provided node name. If this is due to multiple nodes with the same name then all logs for that logger name will go out over the existing publisher. As soon as any node with that name is destroyed it will unregister the publisher, preventing any further logs for that name from being published on the rosout topic.
Robot ROBOT is moving from WAREHOUSE to LOCATION1 carrying a CARRIER . . . [ 100% ]
[plansys2_node-1] [WARN] [1724862730.856548555] [LifecyclePublisher]: Trying to publish message on the topic '/actions_hub', but the publisher is not activated
Robot ROBOT is loading BOX1 in WAREHOUSE carrying a CARRIER . . . [ 100% ]
[plansys2_node-1] [WARN] [1724862730.906614866] [LifecyclePublisher]: Trying to publish message on the topic '/actions_hub', but the publisher is not activated
Robot ROBOT moving from WORKSTATION1 carrying a CARRIER . . . [ 100% ]
[plansys2_node-1] [WARN] [1724862740.836676172] [LifecyclePublisher]: Trying to publish message on the topic '/actions_hub', but the publisher is not activated
Robot ROBOT moving from LOCATION1 to LOCATION2 carrying a CARRIER . . . [ 100% ]
Robot ROBOT start using WORKSTATION2 in LOCATION2 carrying a CARRIER . . . [ 100% ]
Robot ROBOT is deploying BOX1 in WORKSTATION2 carrying a CARRIER . . . [ 100% ]
Robot ROBOT is emptying BOX1 in WORKSTATION2 carrying a CARRIER . . . [ 100% ]
Robot ROBOT stop using WORKSTATION2 in LOCATION2 carrying a CARRIER . . . [ 100% ]
Robot ROBOT moving from LOCATION2 to LOCATION1 carrying a CARRIER . . . [ 100% ]
Robot ROBOT moving from LOCATION1 to WAREHOUSE carrying a CARRIER . . . [ 100% ]
Robot ROBOT is deploying BOX1 in WAREHOUSE carrying a CARRIER . . . [ 100% ]
Robot ROBOT is filling BOX1 in WAREHOUSE with some SCREW carrying a CARRIER . . . [ 100% ]
Robot ROBOT is filling BOX1 in WAREHOUSE with some SCREW carrying a CARRIER . . . [ 100% ]
Robot ROBOT is loading BOX2 in WAREHOUSE carrying a CARRIER . . . [ 100% ]
Robot ROBOT is emptying BOX2 in WAREHOUSE carrying a CARRIER . . . [ 100% ]
Robot ROBOT moving from WAREHOUSE to LOCATION1 carrying a CARRIER . . . [ 100% ]
Robot ROBOT moving from LOCATION1 to LOCATION2 carrying a CARRIER . . . [ 100% ]
Robot ROBOT start using WORKSTATION3 in LOCATION2 carrying a CARRIER . . . [ 100% ]
[plansys2_node-1] [WARN] [1724862767.958171123] [executor]: No action performer for (PUT_DOWN_BOX_IN_WORKSTATION ROBOT BOX1 WORKSTATION3 CARRIER). retrying
Robot ROBOT is deploying BOX2 in WORKSTATION3 carrying a CARRIER . . . [ 100% ]
[plansys2_node-1] [WARN] [1724862768.961388196] [executor]: No action performer for (PUT_DOWN_BOX_IN_WORKSTATION ROBOT BOX1 WORKSTATION3 CARRIER). retrying
Robot ROBOT is emptying BOX2 in WORKSTATION3, the box contained some BOLT carrying a CARRIER . . . [ 100% ]
Robot ROBOT is deploying BOX1 in WORKSTATION3 carrying a CARRIER . . . [ 100% ]
Robot ROBOT is emptying BOX1 in WORKSTATION3, the box contained some SCREW carrying a CARRIER . . . [ 100% ]
[plansys2_node-1] [INFO] [1724862772.54545669] [executor]: Plan Succeeded
```

Figura 17: Contenuto del primo terminale al termine dell'esecuzione del plan

- Nel secondo terminale invece eseguiremo il **terminale di Plansys** con i comandi:

1. ros2 run plansys2\_terminal plansys2\_terminal
2. source ./launch/commands
3. run plan-file <percorso assoluto del file plan.txt>

```

aiguy@vm-ubu22: ~/Desktop/ProgettoIA_Curcio_Gullone/task3_temporal_planning_robots/instance_3_2/industrial_manufacturing_service_robots
aiguy@vm-ubu22:~/Desktop/ProgettoIA_Curcio_Gullone/task3_temporal_planning_robots/instance_3_2/industrial_manufacturing_service_robots$ ros2 run plansys2_terminal plansys2_terminal
[INFO] [1724862662.617196088] [terminal]: No problem file specified.
ROS2 Planning System console. Type "quit" to finish
> source ./launch/commands
done
done
> run plan-file /home/aiguy/Desktop/ProgettoIA_Curcio_Gullone/task3_temporal_planning_robots/instance_3_2/industrial_manufacturing_service_robots/pddl/plan.txt
The plan read from "/home/aiguy/Desktop/ProgettoIA_Curcio_Gullone/task3_temporal_planning_robots/instance_3_2/industrial_manufacturing_service_robots/pddl/plan.txt" is
0: (FILL_BOX_IN_WAREHOUSE ROBOT BOX1 BOLT WAREHOUSE CARRIER) [3.5]
3.5: (LOAD_BOX_FROM_LOCATION ROBOT BOX1 WAREHOUSE CARRIER) [1.5]
5: (MOVE_TO_LOCATION ROBOT WAREHOUSE LOCATION1 CARRIER) [3]
8: (MOVE_TO_LOCATION ROBOT LOCATION1 LOCATION2 CARRIER) [3]
11: (ENTER_WORKSTATION ROBOT LOCATION1 WORKSTATION2 CARRIER) [0]
11: (PUT_DOWN_BOX_IN_WORKSTATION ROBOT BOX1 WORKSTATION2 CARRIER) [1.5]
12.5: (EMPTY_BOX_IN_WORKSTATION ROBOT BOX1 WORKSTATION2 CARRIER) [3.5]
14.5: (LOAD_BOX_FROM_LOCATION ROBOT BOX1 WORKSTATION2 CARRIER) [1.5]
16: (MOVE_TO_LOCATION ROBOT LOCATION2 LOCATION1 CARRIER) [3]
19: (MOVE_TO_WAREHOUSE ROBOT LOCATION1 WAREHOUSE CARRIER) [3]
22: (PUT_DOWN_BOX_IN_LOCATION ROBOT BOX1 WAREHOUSE CARRIER) [1.5]
23.5: (FILL_BOX_IN_WAREHOUSE ROBOT BOX1 SCREW WAREHOUSE CARRIER) [3.5]
27: (LOAD_BOX_FROM_LOCATION ROBOT BOX1 WAREHOUSE CARRIER) [1.5]
22: (FILL_BOX_IN_WAREHOUSE ROBOT BOX2 BOLT WAREHOUSE CARRIER) [3.5]
28.5: (LOAD_BOX_FROM_LOCATION ROBOT BOX2 WAREHOUSE CARRIER) [1.5]
30: (MOVE_TO_LOCATION ROBOT WAREHOUSE LOCATION1 CARRIER) [3]
33: (MOVE_TO_LOCATION ROBOT LOCATION1 LOCATION2 CARRIER) [3]
36: (ENTER_WORKSTATION ROBOT LOCATION1 WORKSTATION3 CARRIER) [0]
36: (PUT_DOWN_BOX_IN_WORKSTATION ROBOT BOX2 WORKSTATION3 CARRIER) [1.5]
36: (EMPTY_BOX_IN_WORKSTATION ROBOT BOX2 WORKSTATION3 CARRIER) [3.5]
37.5: (LOAD_BOX_FROM_LOCATION ROBOT BOX2 WORKSTATION3 CARRIER) [1.5]
37.5: (EMPTY_BOX_IN_WORKSTATION ROBOT BOX2 WORKSTATION3 BOLT CARRIER) [3.5]
[INFO] [1724862774.624360303] [executor_client]: Plan Succeeded
Successful finished
>

```

Figura 18: *Contenuto del secondo terminale al termine dell'esecuzione del plan*

## 4 Organizzazione dell'archivio di lavoro

Il contenuto del progetto è organizzato come segue:

- Cartella "task1\_modelling": contiene due directory, una dedicata ai file pddl che contiene il **Domain.pddl** e il **Problem.pddl** relativi al dominio descritto e l'altra che contiene i risultati dell'esecuzione dei planner di PDDL4J sulle istanze di problema;

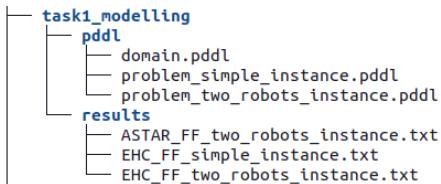


Figura 19: *Contenuto della prima directory di progetto*

- Cartella "task\_classical\_planning": in cui sono presenti le seguenti sottocartelle
  - **src**: contenente i file java utilizzati per l'implementazione del planner personalizzato e dell'euristica;
  - **pddl**: contenente i file **Domain.pddl** e **Problem.pddl** che hanno permesso di modellare i domini e i problemi delle istanze richieste;
  - **results**: contenente i risultati ottenuti;

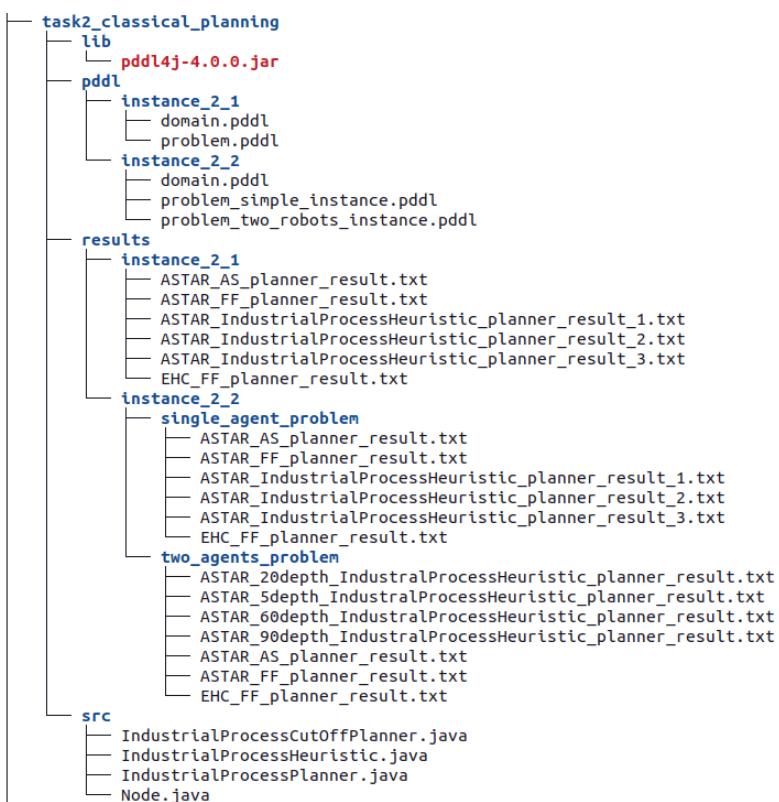


Figura 20: *Contenuto della seconda directory di progetto*

- Cartella "task3\_temporal\_planning\_robotics": in cui sono presenti le sottocartelle
  - **instance\_3\_1**: contenente come prima i file **Domain.pddl** e **Problem.pddl**;
  - **instance\_3\_2**: contenente i file di configurazione e implementazione per tutto il necessario dei nodi per il **task 3**;

```

└── task3_temporal_planning_robotics
    ├── instance_3_1
    │   ├── domain.pddl
    │   ├── problem.pddl
    │   ├── problem_resolution_lpg-td_quality.txt
    │   └── problem_resolution_lpg-td_speedy.txt
    ├── instance_3_2
    │   ├── industrial_manufacturing_service_robots
    │   │   ├── CMakeLists.txt
    │   │   ├── launch
    │   │   │   ├── commands
    │   │   │   └── plansys2_msl.py
    │   │   ├── package.xml
    │   │   └── pddl
    │   │       ├── domain.pddl
    │   │       ├── plan.txt
    │   │       └── problem.pddl
    │   └── src
    │       ├── empty_box_in_location_action_node.cpp
    │       ├── empty_box_in_workstation_action_node.cpp
    │       ├── enter_workstation_action_node.cpp
    │       ├── exit_workstation_action_node.cpp
    │       ├── fill_box_in_location_action_node.cpp
    │       ├── fill_box_in_warehouse_action_node.cpp
    │       ├── fill_box_in_workstation_action_node.cpp
    │       ├── load_box_from_location_action_node.cpp
    │       ├── load_box_from_workstation_action_node.cpp
    │       ├── move_to_location_action_node.cpp
    │       ├── move_to_warehouse_action_node.cpp
    │       ├── put_down_box_in_location_action_node.cpp
    │       └── put_down_box_in_workstation_action_node.cpp

```

Figura 21: *Contenuto della terza directory di progetto*