
# Relazione progetto di Intelligenza Artificiale
# Anno Accademico 2022/2023

**Docenti**

Prof. Francesco Scarcello
Ing. Antonio Bono

**Candidati**

Le Pera Jacopo Alfonso MAT. 247521
Liparoti Silvio Emanuele MAT. 242663
Pisano Salvatore MAT. 247518

Sommario

# 1. Modelling

Il caso di studio del progetto è incentrato su un problema logistico. L'obiettivo è quello di modellare un sistema per la consegna di beni in situazioni di emergenza da parte di un agente robotico. I destinatari dei beni sono delle persone che occupano una posizione ben definita all'interno di una location dove l'agente può muoversi per effettuare le consegne. Come richiesto nell'assignment per la modellazione del problema di pianificazione è stato utilizzato il linguaggio PDDL nella versione 1.2.

## 1.1 Domain file

In questa sezione definiamo il dominio del problema specificando i tipi, i predicati e le azioni dello scenario. Di seguito riportiamo il codice PDDL del nostro dominio commentandolo opportunamente.

```
(define (domain logistics)
    (:requirements :adl :typing :universal-preconditions)
    (:types
        location locatable slot - object
        box carrier robot supplies person - locatable
    )
```

Per prima cosa sono stati definiti i tipi di riferimento:

1. **location, locatable e slot:** sono dei sotto-tipi di object (il tipo più generico del PDDL) e definiscono rispettivamente posizioni specifiche, oggetti posizionabili nello spazio d'azione e infine la capacità del carrello con il quale il robot effettua le consegne.
2. **box, carrier, robot, supplies e person :** sono dei sotto-tipi di locatable e definiscono oggetti che hanno la proprietà di essere posizionabili. A differenza delle box, del carrello, del robot e dei supplies, le persone non possono muoversi all'interno dello spazio d'azione.

Una volta definiti i tipi passiamo alla definizione dei predicati.

```
    (:predicates
       (at ?obj - locatable ?loc - location)
       (contains ?box - box ?sup - supplies)
       (is-empty ?box - box)
       (available ?sl - slot)
       (belongs ?sl - slot ?car - carrier)
       (on ?box - box ?car - carrier)
       (needs ?p - person ?sup - supplies)
       (has ?p - person ?sup - supplies)
    )
```

Dunque:

1. **at:** è il predicato che specifica la localizzazione di un oggetto di tipo locatable;
2. **contains:** predicato che ci dice se il bene (cibo, medicina o strumento) è contenuto nella scatola;
3. **is-empty:** è usato per verificare se la scatola è vuota;
4. **available:** per mantenere la capacità del carrello sono stati usati degli slot e il predicato available ci dice sostanzialmente se lo slot è occupato da una scatola o meno, dunque se tutti gli slot sono not available significa che il carrello è pieno;
5. **belongs:** predicato che definisce l'appartenenza di uno slot ad un carrello**;**
6. **on:** predicato che verifica se la scatola è sul carrello o meno;
7. **needs:** specifica il bene di cui la persona ha bisogno;
8. **has:** questo predicato infine, ci dice se la persona possiede il bene o in maniera equivalente che gli è stato già consegnato.

Per ultima istanza mancano le azioni che ci permettono di completare la definizione del dominio del nostro problema.

```
(:action fill-box
        :parameters ( ?b - box ?r - robot ?sup - supplies ?loc - location
        )
        :precondition (and
            (at ?r ?loc)
            (at ?c ?loc)
            (at ?b ?loc)
            (is-empty ?b)
        )
        :effect (and
            (not (is-empty ?b))
            (contains ?b ?sup)
        )
    )
```

L' azione fill-box ci permette di riempire una scatola di un contenuto. I parametri definiti sono: la box da riempire, il robot, il bene che devono rispettare la precondizione di trovarsi nella stessa location (altro parametro dell'azione). Come ultima precondizione deve valere che la scatola deve essere vuota al fine di essere riempita dall'agente. L'effetto dell'azione afferma che la scatola non è più vuota e che contiene il bene specificato.

```
(:action give-supplies
      :parameters (?b - box ?r - robot ?p - person ?sup - supplies ?loc -
      location
         )
         :precondition (and
             (needs ?p ?sup)
             (contains ?b ?sup)
             (at ?b ?loc)
             (at ?r ?loc)
             (at ?p ?loc)
         )
         :effect (and
             (not (contains ?b ?sup))
             (is-empty ?b)
             (has ?p ?sup)
         )
    )
```

L' azione give-supplies presenta pressoché gli stessi parametri dell'azione fill-box, ma aggiunge la persona a cui deve essere consegnato il supplies. Per quanto riguarda le precondizioni il robot, la box è la persona devono trovarsi alla location loc, la scatola deve contenere il bene ?*c* di cui la persona ha bisogno (*needs ?p ?c*). L'effetto dell'azione give-supplies prevede lo svuotamento della scatola verificando poi che sia vuota. Infine si verifica che la persona possiede il bene che era nella scatola.

```
    (:action load-box
      :parameters (?b - box ?r - robot ?loc - location ?car - carrier
?sl - slot
         )
         :precondition (and
             (at ?b ?loc)
             (at ?car ?loc)
             (at ?r ?loc)
             (belongs ?sl ?car)
             (available ?sl)
         )
         :effect (and
             (not (at ?b ?loc))
             (on ?b ?car)
             (not (available ?sl))
         )
    )
```

Questa azione ci permette di caricare la scatola sul carrello. I parametri definiti sono i seguenti: ?b - box ?r - robot ?loc - location ?car - carrier  ?sl – slot. La box, il carrello e il robot devono soddisfare il predicato at per la location ?loc. Successivamente viene verificato il binding tra il carello e lo slot il quale deve essere disponibile al caricamento. L'effetto dell'azione prevede che la scatola non deve essere più nella location poiché ora è sul carrello (che comunque è nella location) e successivamente viene specificato che lo slot non è più available.

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

L' azione move-robot è semplice e prevede lo spostamento del robot dalla location ?from alla location ?to. L'unica precondition è quella che il robot debba partire da ?from, mentre l'effetto causa lo spostamento alla location ?to.

```
    (:action move-robot-carrier
        :parameters (?r - robot ?car - carrier ?from - location ?to -
location
        )
        :precondition (and
            (at ?r ?from)
            (at ?car ?from)
        )
        :effect (and
            (not (at ?r ?from))
            (at ?r ?to)
            (not (at ?car ?from))
            (at ?car ?to)
        )
    )
```

Questa è l'azione che permette al robot di spostarsi insieme al carrello. I parametri dell'azione sono i medesimi dell'azione move-robot, ma in aggiunta abbiamo il carrello ?car. L'agente per spostarsi con il carrello deve trovarsi nella sua stessa location, ovvero ?from. L'effetto prevede che robot e carrello sono nella location ?to e non in ?from.

```
    (:action unload-box
```

```
        :parameters (?b - box ?r - robot ?loc - location ?sl - slot ?car
- carrier)
        :precondition (and
            (on ?b ?car)
            (at ?car ?loc)
            (at ?r ?loc)
            (belongs ?sl ?car)
            (not (available ?sl))
        )
        :effect (and
            (at ?b ?loc)
            (available ?sl)
            (not (on ?b ?car))
        )
    )

)
```

L’ ultima azione chiamata unload-box permette all’agente di scaricare la scatola dal carrello. I parametri definiti sono i seguenti: ?b - box ?r - robot ?loc - location ?sl - slot ?car – carrier. La scatola deve trovarsi sul carrello il quale si trova alla locazione ?loc così come il robot; deve anche valere il solito binding tra il carrello e lo slot che deve essere not available. L’effetto prevede che la scatola si trovi nella location poiché appena scaricata dal carrello: (at ?b ?loc) e (not (on ?b ?car). Infine abbiamo che lo slot ritorna disponibile.

# 2. Classical Planning

Dopo aver completato la definizione del file di dominio, il passo successivo nella pianificazione automatica coinvolge la creazione del file di problema definendo lo stato iniziale dell'ambiente in cui l'agente deve iniziare a pianificare e gli obiettivi che deve provare a raggiungere. Le istanze di problema in PDDL richieste dalla traccia erano tre ognuna con delle specifiche diverse.

## 2.1 Istanza 1

Di seguito riportiamo il codice della nostra prima istanza:

```
(define (problem instance1)
    (:domain logistics)
    (:objects
        depot loc1 loc2 - location
        b1 b2 b3 b4 b5 - box
        p1 p2 p3 - person
        r - robot
        car - carrier
        sl1 sl2 sl3 sl4 - slot
        drugs food tools - supplies
    )
```

La prima istanza prevede che ci siano 5 casse in una location che rappresenta il deposito; inizialmente anche il cibo, le medicine e gli strumenti si trovano qui. Abbiamo poi il robot agente, un carrello con quattro slot per il carico. In questa situazione, abbiamo tre persone (p1, p2, p3) che richiedono assistenza e vanno rispettati i seguenti vincoli:

1. p1 e p2 sono nella stessa location.
2. p1 ha bisogno di cibo e medicine, p2 di medicine e p3 di cibo.

L'obiettivo dell'agente robotico è quello di soddisfare le richieste delle persone.

Allo stato iniziale valgono le seguenti condizioni specificate nella sezione init:

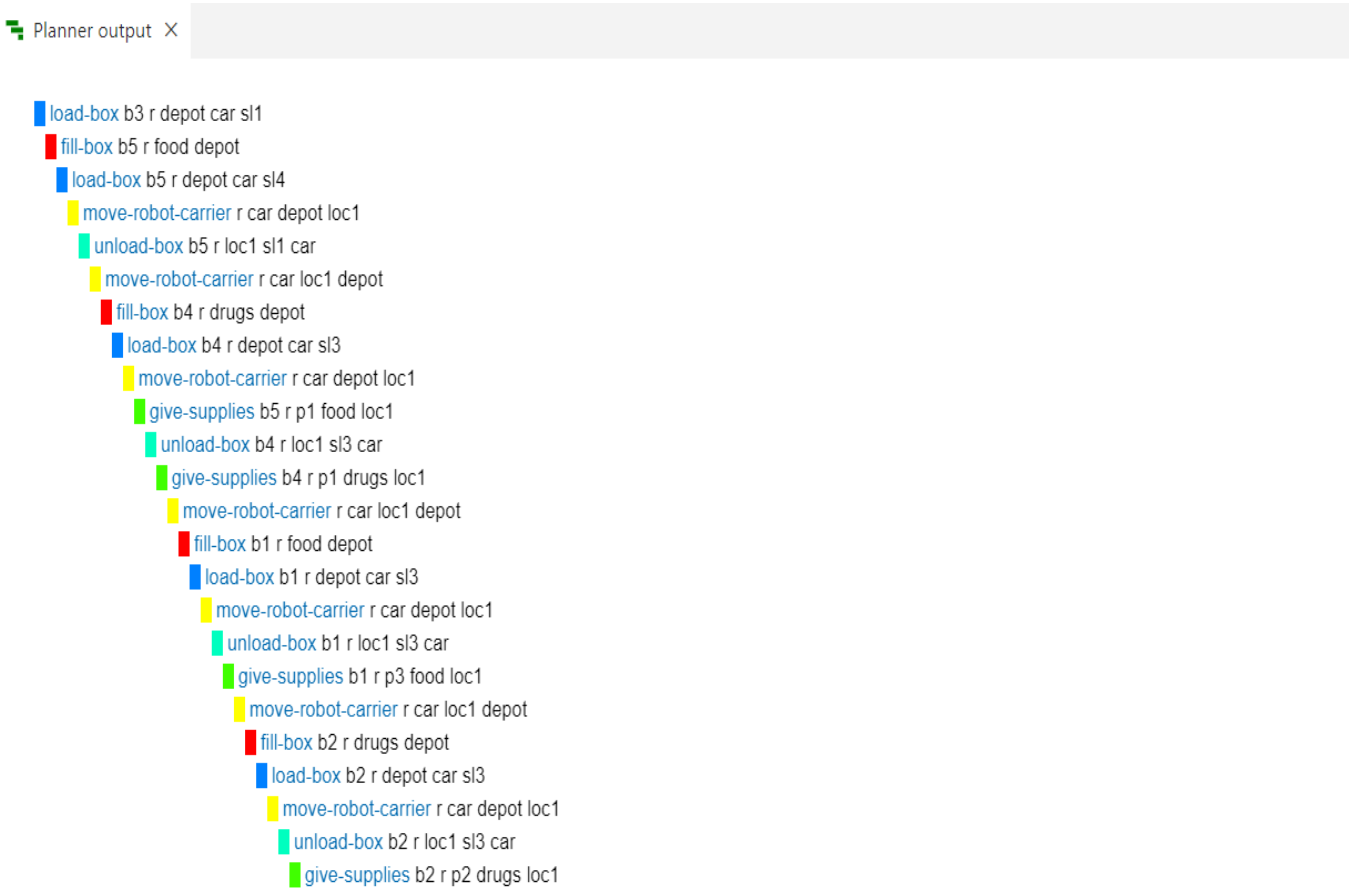
```
    (:init
        (at b1 depot)
        (at b2 depot)
        (at b3 depot)
        (at b4 depot)
        (at b5 depot)
        (at r depot)
        (at car depot)
        (at food depot)
        (at drugs depot)
        (at tools depot)
```

```
        (at p1 loc1)
        (at p2 loc1)
        (at p3 loc2)
        (needs p1 food)
        (needs p1 drugs)
        (needs p2 drugs)
        (needs p3 food)
        (is-empty b1)
        (is-empty b2)
        (is-empty b3)
        (is-empty b4)
        (is-empty b5)
        (available sl1)
        (available sl2)
        (available sl3)
        (available sl4)
        (belongs sl1 car)
        (belongs sl2 car)
        (belongs sl3 car)
        (belongs sl4 car)
    )

    (:goal
        (and
            (has p1 food)
            (has p1 drugs)
            (has p2 drugs)
            (has p3 food)
        )
    )
)
```

L'output mostra il piano ottenuto con il solver online e relativi dettagli.

```
Planning service: http://solver.planning.domains/solve
Domain: logistics, Problem: instance1
 --- OK.
 Match tree built with 119 nodes.

PDDL problem description loaded:
    Domain: LOGISTICS
    Problem: INSTANCE1
    #Actions: 119
    #Fluents: 51
Landmarks found: 4
Starting search with IW (time budget is 60 secs)...
rel_plan size: 9
#RP_fluents 13
```

```
Caption
{#goals, #UNnachieved,  #Achieved} -> IW(max_w)

{4/4/0}:IW(1) -> [2][3][4][5][6];; NOT I-REACHABLE ;;
Total time: -1.04308e-10
Nodes generated during search: 172
Nodes expanded during search: 172
IW search completed
Starting search with BFS(novel,land,h_add)...
Total time: 0.012
Nodes generated during search: 1467
Nodes expanded during search: 119
Plan found with cost: 24
BFS search completed
```



## 2.2 Istanza 2

La seconda istanza prevede che ci siano 3 scatole anziché 5, mentre il carrello possiede due slot per il carico. Abbiamo sei persone (p1, p2, p3, p4, p5, p6) che richiedono assistenza e vanno rispettati i seguenti vincoli:

1. p1 e p2 sono nella stessa location, mentre gli altri sono in posizioni diverse.
2. p1 ha bisogno di cibo e strumenti, p2 e p3 di medicine, p4 di medicine e cibo, p5 e p6 hanno bisogno di tutti i supplies.

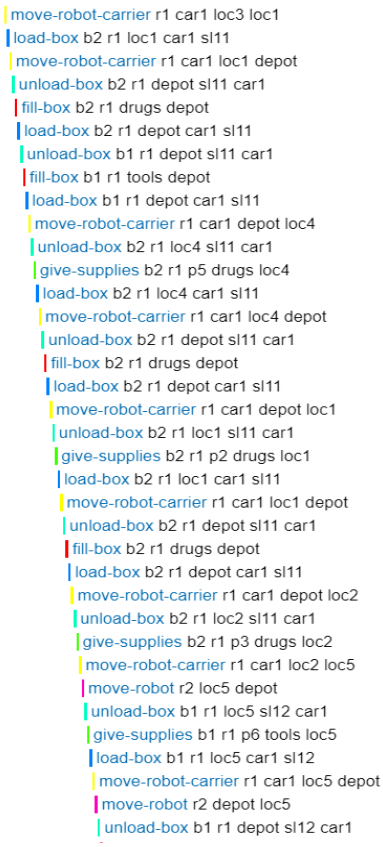
Il goal da soddisfare è il medesimo dell'istanza 1.

```
(define (problem instnce2)
    (:domain logistics)
    (:objects
        depot loc1 loc2 loc3 loc4 loc5 - location
        b1 b2 b3 - box
        p1 p2 p3 p4 p5 p6 - person
        r1 r2 - robot
        car1 car2 - carrier
        sl11 sl12 sl21 sl22 - slot
        drugs food tools - supplies
    )
    (:init
        (at b1 depot)
        (at b2 depot)
        (at b3 depot)
        (at r1 depot)
        (at r2 depot)
        (at car1 depot)
        (at car2 depot)
        (at food depot)
        (at drugs depot)
        (at tools depot)
        (at p1 loc1)
        (at p2 loc1)
        (at p3 loc2)
        (at p4 loc3)
        (at p5 loc4)
        (at p6 loc5)
        (needs p1 food)
        (needs p1 tools)
        (needs p2 drugs)
        (needs p3 drugs)
        (needs p4 food)
        (needs p4 drugs)
        (needs p5 food)
        (needs p5 drugs)
```

```
        (needs p5 tools)
        (needs p6 food)
        (needs p6 drugs)
        (needs p6 tools)
        (is-empty b1)
        (is-empty b2)
        (is-empty b3)
        (available sl11)
        (available sl12)
        (available sl21)
        (available sl22)
        (belongs sl11 car1)
        (belongs sl12 car1)
        (belongs sl21 car2)
        (belongs sl22 car2)
    )

    (:goal
        (and
            (has p1 food)
            (has p1 tools)
            (has p2 drugs)
            (has p3 drugs)
            (has p4 food)
            (has p4 drugs)
            (has p5 food)
            (has p5 drugs)
            (has p5 tools)
            (has p6 food)
            (has p6 drugs)
            (has p6 tools)
        )
    )
)
```

Planner output

```
load-box b1 r1 depot car1 sl11
move-robot r2 depot loc5
fill-box b3 r1 drugs depot
load-box b3 r1 depot car1 sl12
unload-box b1 r1 depot sl11 car1
fill-box b1 r1 food depot
fill-box b2 r1 tools depot
load-box b1 r1 depot car1 sl11
move-robot-carrier r1 car1 depot loc4
unload-box b1 r1 loc4 sl11 car1
give-supplies b1 r1 p5 food loc4
move-robot-carrier r1 car1 loc4 depot
load-box b2 r1 depot car1 sl11
move-robot-carrier r1 car1 depot loc1
unload-box b2 r1 loc1 sl11 car1
give-supplies b2 r1 p1 tools loc1
load-box b2 r1 loc1 car1 sl11
move-robot-carrier r1 car1 loc1 depot
unload-box b2 r1 depot sl11 car1
fill-box b2 r1 food depot
load-box b2 r1 depot car1 sl11
move-robot-carrier r1 car1 depot loc3
unload-box b2 r1 loc3 sl11 car1
give-supplies b2 r1 p4 food loc3
move-robot-carrier r1 car1 loc3 loc4
load-box b1 r1 loc4 car1 sl11
move-robot r2 loc5 depot
move-robot-carrier r1 car1 loc4 loc5
unload-box b3 r1 loc5 sl11 car1
give-supplies b3 r1 p6 drugs loc5
load-box b3 r1 loc5 car1 sl11
move-robot-carrier r1 car1 loc5 depot
move-robot r2 depot loc5
unload-box b3 r1 depot sl11 car1
fill-box b3 r1 drugs depot
load-box b3 r1 depot car1 sl11
unload-box b1 r1 depot sl11 car1
fill-box b1 r1 tools depot
load-box b1 r1 depot car1 sl11
move-robot-carrier r1 car1 depot loc4
unload-box b1 r1 loc4 sl11 car1
give-supplies b1 r1 p5 tools loc4
move-robot-carrier r1 car1 loc4 loc3
load-box b2 r1 loc3 car1 sl11
move-robot-carrier r1 car1 loc3 depot
unload-box b2 r1 depot sl11 car1
fill-box b2 r1 food depot
load-box b2 r1 depot car1 sl11
move-robot-carrier r1 car1 depot loc1
unload-box b2 r1 loc1 sl11 car1
give-supplies b2 r1 p1 food loc1
move-robot-carrier r1 car1 loc1 loc4
load-box b1 r1 loc4 car1 sl11
move-robot-carrier r1 car1 loc4 loc3
unload-box b3 r1 loc3 sl11 car1
give-supplies b3 r1 p4 drugs loc3
```




```
Planning service: http://solver.planning.domains/solve
Domain: logistics, Problem: instnce2
 --- OK.
 Match tree built with 558 nodes.

PDDL problem description loaded:
    Domain: LOGISTICS
    Problem: INSTNCE2
    #Actions: 558
    #Fluents: 80
Landmarks found: 12
Starting search with IW (time budget is 60 secs)...
rel_plan size: 31
#RP_fluents 39
Caption
{#goals, #UNnachieved,  #Achieved} -> IW(max_w)

{12/12/0}:IW(1) -> [2][3][4][5][6][7][8][9][10]rel_plan size: 29
#RP_fluents 41
```

```
{12/11/1}:IW(1) -> [2][3][4][5][6][7][8][9][10][11][12][13][14];; NOT I-REACHABLE
;;
Total time: 0.02
Nodes generated during search: 1426
Nodes expanded during search: 1374
IW search completed
Starting search with BFS(novel,land,h_add)...
Total time: 0.124
Nodes generated during search: 4981
Nodes expanded during search: 218
Plan found with cost: 97
BFS search completed
```

## 2.3 Istanza 3

La terza istanza è praticamente uguale alla seconda ma in questo caso abbiamo 4 boxes e otto persone (p1, p2, p3, p4, p5, p6, p7, p8) che richiedono assistenza e vanno rispettati i seguenti vincoli:

1. p1 e p2 sono nella stessa location, mentre gli altri sono in posizioni diverse.
2. p1 ha bisogno di cibo e strumenti, p2 e p3 di medicine, p4 di medicine e cibo, p5,p6,p7,p8 hanno bisogno di tutti i supplies.

Il goal da soddisfare è il medesimo.

```
(define (problem instance3)
    (:domain logistics)
    (:objects
        depot loc1 loc2 loc3 loc4 loc5 loc6 loc7 - location
        b1 b2 b3 b4 - box
        p1 p2 p3 p4 p5 p6 p7 p8 - person
        r1 r2 - robot
        car1 car2 - carrier
        sl11 sl12 sl21 sl22 - slot
        drugs food tools - supplies
    )
    (:init
        (at b1 depot)
        (at b2 depot)
        (at b3 depot)
        (at b4 depot)
        (at r1 depot)
        (at r2 depot)
        (at car1 depot)
        (at car2 depot)
        (at food depot)
        (at drugs depot)
        (at tools depot)
```

```
    (at p1 loc1)
    (at p2 loc1)
    (at p3 loc2)
    (at p4 loc3)
    (at p5 loc4)
    (at p6 loc5)
    (at p7 loc6)
    (at p8 loc7)
    (needs p1 food)
    (needs p1 tools)
    (needs p2 drugs)
    (needs p3 drugs)
    (needs p4 food)
    (needs p4 drugs)
    (needs p5 food)
    (needs p5 drugs)
    (needs p5 tools)
    (needs p6 food)
    (needs p6 drugs)
    (needs p6 tools)
    (needs p7 food)
    (needs p7 drugs)
    (needs p7 tools)
    (needs p8 food)
    (needs p8 drugs)
    (needs p8 tools)
    (is-empty b1)
    (is-empty b2)
    (is-empty b3)
    (is-empty b4)
    (available sl11)
    (available sl12)
    (available sl21)
    (available sl22)
    (belongs sl11 car1)
    (belongs sl12 car1)
    (belongs sl21 car2)
    (belongs sl22 car2)
)

(:goal
    (and
        (has p1 food)
        (has p1 tools)
        (has p2 drugs)
        (has p3 drugs)
        (has p4 food)
        (has p4 drugs)
        (has p5 food)
```

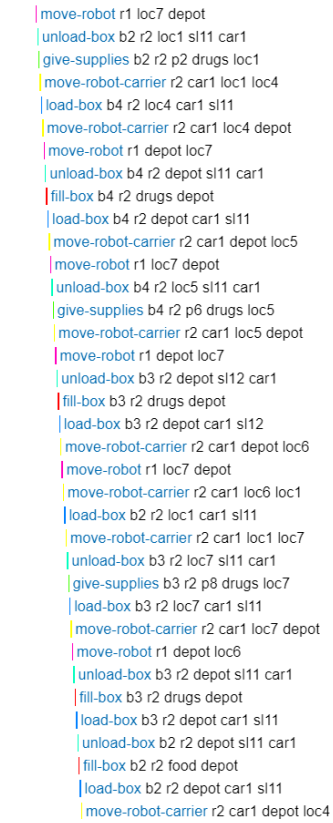
```
            (has p5 drugs)
            (has p5 tools)
            (has p6 food)
            (has p6 drugs)
            (has p6 tools)
            (has p7 food)
            (has p7 drugs)
            (has p7 tools)
            (has p8 food)
            (has p8 drugs)
            (has p8 tools)
        )
    )
)
```

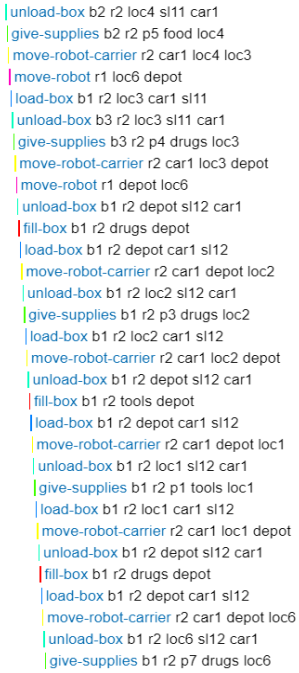
Planner output

```
load-box b3 r1 depot car1 sl11
fill-box b4 r1 drugs depot
load-box b4 r1 depot car1 sl12
move-robot r2 depot loc7
unload-box b3 r1 depot sl11 car1
fill-box b3 r1 food depot
fill-box b2 r1 tools depot
load-box b3 r1 depot car1 sl11
move-robot r2 loc7 loc6
move-robot-carrier r1 car1 depot loc7
move-robot r2 loc6 depot
load-box b2 r2 depot car2 sl22
move-robot-carrier r2 car2 depot loc6
move-robot-carrier r1 car1 loc7 loc5
unload-box b3 r1 loc5 sl11 car1
give-supplies b3 r1 p6 food loc5
move-robot-carrier r1 car1 loc5 depot
fill-box b1 r1 food depot
load-box b1 r1 depot car1 sl11
move-robot-carrier r1 car1 depot loc1
unload-box b1 r1 loc1 sl11 car1
give-supplies b1 r1 p1 food loc1
move-robot-carrier r2 car2 loc6 depot
load-box b1 r1 loc1 car1 sl11
move-robot-carrier r1 car1 loc1 loc7
move-robot-carrier r2 car2 depot loc6
move-robot-carrier r1 car1 loc7 depot
unload-box b1 r1 depot sl11 car1
fill-box b1 r1 food depot
load-box b1 r1 depot car1 sl11
move-robot-carrier r1 car1 depot loc7
move-robot-carrier r2 car2 loc6 loc4
move-robot-carrier r1 car1 loc7 loc6
unload-box b1 r1 loc6 sl11 car1
give-supplies b1 r1 p7 food loc6
move-robot-carrier r2 car2 loc4 depot
```

load-box b1 r1 loc6 car1 sl11
move-robot-carrier r1 car1 loc6 loc7
move-robot-carrier r2 car2 depot loc4
move-robot-carrier r1 car1 loc7 depot
unload-box b1 r1 depot sl11 car1
fill-box b1 r1 food depot
load-box b1 r1 depot car1 sl11
move-robot-carrier r1 car1 depot loc7
unload-box b1 r1 loc7 sl11 car1
give-supplies b1 r1 p8 food loc7
move-robot-carrier r2 car2 loc4 depot
load-box b1 r1 loc7 car1 sl11
move-robot-carrier r2 car2 depot loc4
move-robot-carrier r1 car1 loc7 depot
unload-box b1 r1 depot sl11 car1
fill-box b1 r1 food depot
load-box b1 r1 depot car1 sl11
move-robot-carrier r1 car1 depot loc3
unload-box b1 r1 loc3 sl11 car1
give-supplies b1 r1 p4 food loc3
move-robot-carrier r1 car1 loc3 loc4
move-robot r2 loc4 loc7
unload-box b2 r1 loc4 sl22 car2
give-supplies b2 r1 p5 tools loc4
move-robot r2 loc7 depot
load-box b2 r1 loc4 car1 sl11
move-robot-carrier r1 car1 loc4 depot
move-robot r1 depot loc7
unload-box b2 r2 depot sl11 car1
fill-box b2 r2 tools depot
load-box b2 r2 depot car1 sl11
move-robot-carrier r2 car1 depot loc6
move-robot r1 loc7 loc5
move-robot-carrier r2 car1 loc6 loc7
unload-box b2 r2 loc7 sl11 car1
give-supplies b2 r2 p8 tools loc7

move-robot r1 loc5 depot
load-box b2 r2 loc7 car1 sl11
move-robot-carrier r2 car1 loc7 depot
move-robot r1 depot loc6
unload-box b2 r2 depot sl11 car1
fill-box b2 r2 tools depot
load-box b2 r2 depot car1 sl11
move-robot-carrier r2 car1 depot loc5
unload-box b2 r2 loc5 sl11 car1
give-supplies b2 r2 p6 tools loc5
move-robot r1 loc6 depot
load-box b3 r2 loc5 car1 sl11
move-robot-carrier r2 car1 loc5 loc6
move-robot r1 depot loc4
move-robot-carrier r2 car1 loc6 depot
unload-box b3 r2 depot sl11 car1
fill-box b3 r2 tools depot
load-box b3 r2 depot car1 sl11
move-robot-carrier r2 car1 depot loc6
unload-box b3 r2 loc6 sl11 car1
give-supplies b3 r2 p7 tools loc6
move-robot-carrier r2 car1 loc6 loc5
load-box b2 r2 loc5 car1 sl11
move-robot r1 loc4 loc1
move-robot-carrier r2 car1 loc5 loc4
unload-box b4 r2 loc4 sl11 car1
give-supplies b4 r2 p5 drugs loc4
move-robot-carrier r2 car1 loc4 depot
unload-box b2 r2 depot sl12 car1
fill-box b2 r2 drugs depot
load-box b2 r2 depot car1 sl12
move-robot-carrier r2 car1 depot loc6
load-box b3 r2 loc6 car1 sl11
move-robot-carrier r2 car1 loc6 depot
move-robot r1 loc1 loc7
move-robot-carrier r2 car1 depot loc1




```
Planning service: http://solver.planning.domains/solve
Domain: logistics, Problem: instance3
 --- OK.
```

```
 Match tree built with 1016 nodes.
PDDL problem description loaded:
    Domain: LOGISTICS
    Problem: INSTANCE3
    #Actions: 1016
    #Fluents: 114
Landmarks found: 18
Starting search with IW (time budget is 60 secs)...
rel_plan size: 43
#RP_fluents 53
Caption
{#goals, #UNnachieved,  #Achieved} -> IW(max_w)
{18/18/0}:IW(1) -> [2][3][4][5][6][7][8][9][10][11][12]rel_plan size: 41
#RP_fluents 57
{18/17/1}:IW(1) -> [2][3][4][5][6][7][8][9][10][11][12][13][14][15][16][17][18];;
NOT I-REACHABLE ;;
Total time: 0.052
Nodes generated during search: 2821
Nodes expanded during search: 2743
IW search completed
Starting search with BFS(novel,land,h_add)...
Total time: 5.476
Nodes generated during search: 91985
Nodes expanded during search: 3136
Plan found with cost: 175
BFS search completed
Match tree built with 1016 nodes.
```

## 2.4 Struttura ed implementazione del planner

I risultati finora ottenuti sono stati possibili grazie alla risoluzione mediante solver online. In questa sezione, invece, forniremo una dettagliata descrizione del nostro planner, spiegando le sue caratteristiche chiave, l'approccio di pianificazione utilizzato e come è stato adattato per risolvere con successo il problema specifico affrontato nella pianificazione automatica e successivamente presenteremo i risultati. In questa fase di progetto è stata utilizzata la libreria PDDL4J che ci ha fornito una solida base per la gestione e la manipolazione dei modelli PDDL precedentemente introdotti. La nostra classe MyPlanner fa uso dell'algoritmo Iterative Deepening A* che combina le caratteristiche dell'algoritmo A* e della Ricerca a Profondità Iterativa (Iterative Deepening). IDA* inizia con una ricerca a profondità limitata e successivamente incrementa progressivamente il limite di profondità finché non trova una soluzione. Questo approccio consente di esplorare l'intero spazio di ricerca senza dover memorizzare l'intero albero, come avviene in A*. IDA* valuta la qualità dei nodi in base a una combinazione del costo del percorso finora e a una stima euristica del costo

rimanente per raggiungere l'obiettivo. L'algoritmo imposta un limite di profondità iniziale e inizia a esplorare il grafo di ricerca. Se il costo stimato supera il limite di profondità, l'algoritmo si interrompe e aumenta il limite di profondità per la prossima iterazione. IDA* esegue una ricerca ricorsiva, esplorando in profondità fino al limite corrente, quindi aumenta il limite e ripete il processo finché non trova una soluzione. In generale, IDA* è una scelta appropriata quando si desidera bilanciare l'efficienza nella gestione della memoria con la necessità di esplorare uno spazio di ricerca senza conoscere a priori la profondità massima del percorso ottimo. Ecco riportato il codice opportunamente commentato della nostra classe MyPlanner:

```
public class myPlanner extends AbstractPlanner {

    private static final Logger LOGGER =
LogManager.getLogger(myPlanner.class.getName());



//Costanti per il setting dell'euristica

    public static final String HEURISTIC_SETTING = "HEURISTIC";
    public static final StateHeuristic.Name DEFAULT_HEURISTIC =
StateHeuristic.Name.AJUSTED_SUM;
    public static final String WEIGHT_HEURISTIC_SETTING =
"WEIGHT_HEURISTIC";
    public static final double DEFAULT_WEIGHT_HEURISTIC = 1.0;
    private double heuristicWeight;
    private StateHeuristic.Name heuristic;

    public myPlanner() {
        this(myPlanner.getDefaultConfiguration());
    }
    public myPlanner(final PlannerConfiguration configuration) {
        super();
        this.setConfiguration(configuration);
    }
}

//metodi getter e setter

    @CommandLine.Option(names = {"-w",
        "--weight"}, defaultValue = "1.0", paramLabel = "<weight>",
description = "Set the weight of the heuristic (preset 1.0).")
    public void setHeuristicWeight(final double weight) {
        if (weight <= 0) {
            throw new IllegalArgumentException("Weight <= 0");
        }
        this.heuristicWeight = weight;
```

```
    }

    @CommandLine.Option(names = {"-e",
        "--heuristic"}, defaultValue = "FAST_FORWARD", description = "Set
the heuristic : AJUSTED_SUM, AJUSTED_SUM2, AJUSTED_SUM2M, COMBO, "
        + "MAX, FAST_FORWARD SET_LEVEL, SUM, SUM_MUTEX (preset:
FAST_FORWARD)")
    public void setHeuristic(StateHeuristic.Name heuristic) {
        this.heuristic = heuristic;
    }

    public final StateHeuristic.Name getHeuristic() {
        return this.heuristic;
    }

    public final double getHeuristicWeight() {
        return this.heuristicWeight;
    }
```

```
@Override
public Problem instantiate(DefaultParsedProblem problem) {
// Questo metodo restituisce il problema istanziato.
    final Problem pb = new DefaultProblem(problem);
    pb.instantiate();
    return pb;
}

@Override
public Plan solve(final Problem problem) {
// Questo metodo cerca di risolvere il problema di pianificazione
utilizzando un algoritmo specifico.
    Plan plan = null;

    try {
        final long begin = System.currentTimeMillis();
// Esegui un algoritmo di ricerca (nel nostro caso IDA*) per trovare una
soluzione al problema.
        plan = this.iterative_deepening_astar_search(problem);
        LOGGER.info("* Starting My IDA* search \n");
        final long end = System.currentTimeMillis();
// Se viene trovato un piano, aggiorna le statistiche del planner e
registra le informazioni sulla ricerca.
        if (plan != null) {
            LOGGER.info("* My IDA* search succeeded\n");
            this.getStatistics().setTimeToSearch(end - begin);
        } else {
```

```
            LOGGER.info("* My IDA* search failed\n");
//altrimenti registra il fallimento della ricerca.
        }

    } catch (ProblemNotSupportedException e) {
        LOGGER.error("not supported problem");
     // Registra un errore se il problema non è supportato.
        e.printStackTrace(); // Stampa lo stack trace dell'eccezione.
    } finally {
        // Restituisce il piano trovato o null se la ricerca fallisce.
        return plan;
    }
}
/*
Questi metodi gestiscono la configurazione dell'oggetto myPlanner,
inclusa la validità della configurazione, la configurazione predefinita,
il recupero della configurazione attuale e l'impostazione di una nuova
configurazione.
Controlla che il peso dell'euristica sia maggiore di zero e che il tipo
di euristica non sia nullo.
 */
public boolean hasValidConfiguration() {
        return super.hasValidConfiguration()
            && this.getHeuristicWeight() > 0.0
            && this.getHeuristic() != null;
    }

    public static PlannerConfiguration getDefaultConfiguration() {
        PlannerConfiguration config = Planner.getDefaultConfiguration();
        config.setProperty(myPlanner.HEURISTIC_SETTING,
myPlanner.DEFAULT_HEURISTIC.toString());
        config.setProperty(myPlanner.WEIGHT_HEURISTIC_SETTING,
            Double.toString(myPlanner.DEFAULT_WEIGHT_HEURISTIC));
        return config;
    }

    @Override
    public PlannerConfiguration getConfiguration() {
        final PlannerConfiguration config = super.getConfiguration();
        config.setProperty(myPlanner.HEURISTIC_SETTING,
this.getHeuristic().toString());
        config.setProperty(myPlanner.WEIGHT_HEURISTIC_SETTING,
Double.toString(this.getHeuristicWeight()));
        return config;
    }

    @Override
    public void setConfiguration(final PlannerConfiguration
configuration) {
```

```
        super.setConfiguration(configuration);
        if (configuration.getProperty(myPlanner.WEIGHT_HEURISTIC_SETTING)
== null) {
            this.setHeuristicWeight(myPlanner.DEFAULT_WEIGHT_HEURISTIC);
        } else {
            this.setHeuristicWeight(Double.parseDouble(configuration.getP
roperty(
                myPlanner.WEIGHT_HEURISTIC_SETTING)));
        }
        if (configuration.getProperty(myPlanner.HEURISTIC_SETTING) ==
null) {
            this.setHeuristic(myPlanner.DEFAULT_HEURISTIC);
        } else {
            this.setHeuristic(StateHeuristic.Name.valueOf(configuration.g
etProperty(
                myPlanner.HEURISTIC_SETTING)));
        }
    }

/*
Esegue una ricerca IDA* (Iterative Deepening A*) per risolvere il
problema di pianificazione specificato. Se il problema non è supportato,
viene generata un'eccezione. Restituisce il piano trovato o null se la
ricerca non ha avuto successo.
*/
public Plan iterative_deepening_astar_search(Problem problem) throws
ProblemNotSupportedException {
    if (!this.isSupported(problem)) {
        throw new ProblemNotSupportedException("Problem not supported");
    }

    Plan result = null;
    int max_depth = Integer.MAX_VALUE;

    for (int current_depth = 0; current_depth < max_depth;
current_depth++) {
        if (result == null) {
            result = depth_limited_search(problem, current_depth);
        }
    }

    return result;
}
/*
Esegue una ricerca a profondità limitata per risolvere il problema di
pianificazione specificato.
Questa ricerca utilizza l'euristica A* con una frontiera basata sulla
funzione f = w*h + g.
Restituisce il piano trovato o null se la ricerca non ha avuto successo.
```

```
*/
public Plan depth_limited_search(Problem problem, int depth) {
    final StateHeuristic heuristic =
StateHeuristic.getInstance(this.getHeuristic(), problem);
    final State init = new State(problem.getInitialState());
// Inizializza l'insieme dei nodi esplorati.
    final Set<Node> close = new HashSet<>();
/*
Inizializza la frontiera utilizzando una coda di priorità basata su
f = w*h + g. Il comparatore personalizzato viene utilizzato per
determinare l'ordine di estrazione degli oggetti Node dalla coda di
priorità. Si basa sul calcolo di f1 e f2 per i nodi n1 e n2,
rispettivamente. Il valore f1 è calcolato come il prodotto del peso
dell'euristica (weight) e l'euristica del nodo n1, sommato al costo
accumulato per raggiungere n1. In modo simile, il valore f2 è calcolato
per il nodo n2. Infine, il comparatore utilizza Double.compare per
confrontare f1 e f2 e stabilire l'ordine di estrazione dalla coda di
priorità.
*/

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

    // Crea il nodo radice con la stima dello stato iniziale.
    final Node root = new Node(init, null, -1, 0,
heuristic.estimate(init, problem.getGoal()));
    open.add(root);
    Plan plan = null;

    final int timeout = this.getTimeout() * 1000;
    long time = 0;

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

    // Se il goal è stato soddisfatto, calcola il piano e lo restituisce.
        if (current.satisfy(problem.getGoal())) {
            return this.extractPlan(current, problem);
        } else if (current.getDepth() > depth) {
            return null; // La profondità massima è stata raggiunta.
        } else {
```

```
            for (int i = 0; i < problem.getActions().size(); i++) {
                Action a = problem.getActions().get(i);

    // Verifica se l'azione i-esima è applicabile allo stato corrente.
                if (a.isApplicable(current)) {
                    Node next = new Node(current);
    // Applica l'azione e i suoi effetti condizionali.
                    final List<ConditionalEffect> effects =
a.getConditionalEffects();
                    for (ConditionalEffect ce : effects) {
                        if (current.satisfy(ce.getCondition())) {
                            next.apply(ce.getEffect());
                        }
                    }
    // Aggiorna il costo di g.
                    final double g = current.getCost() + 1;
    // Se il nodo non è stato esplorato precedentemente, aggiungilo alla
frontiera.
                    if (!close.contains(next)) {
                        next.setCost(g);
                        next.setParent(current);
                        next.setAction(i);
                        next.setHeuristic(heuristic.estimate(next,
problem.getGoal()));
                        open.add(next);
                    }
                }
            }
        }
    }
    return plan;
}

/*
 Estrae un piano a partire dal nodo finale della ricerca e dal problema
di pianificazione.
 Inizia dal nodo finale e risale la gerarchia dei nodi genitori per
ottenere la sequenza di azioni che compongono il piano.
*/

private Plan extractPlan(final Node node, final Problem problem) {
    Node n = node;
    final Plan plan = new SequentialPlan();
    // Risale la gerarchia dei nodi genitori a partire dal nodo finale.
    while (n.getAction() != -1) {
        final Action a = problem.getActions().get(n.getAction());
        plan.add(0, a);
        n = n.getParent();
    }
```

```
    return plan;
}


@Override
public boolean isSupported(Problem problem) {
        return
(problem.getRequirements().contains(RequireKey.ACTION_COSTS)
            || problem.getRequirements().contains(RequireKey.CONSTRAINTS)
            ||
problem.getRequirements().contains(RequireKey.CONTINOUS_EFFECTS)
            ||
problem.getRequirements().contains(RequireKey.DERIVED_PREDICATES)
            ||
problem.getRequirements().contains(RequireKey.DURATIVE_ACTIONS)
            ||
problem.getRequirements().contains(RequireKey.DURATION_INEQUALITIES)
            || problem.getRequirements().contains(RequireKey.FLUENTS)
            ||
problem.getRequirements().contains(RequireKey.GOAL_UTILITIES)
            ||
problem.getRequirements().contains(RequireKey.METHOD_CONSTRAINTS)
            ||
problem.getRequirements().contains(RequireKey.NUMERIC_FLUENTS)
            ||
problem.getRequirements().contains(RequireKey.OBJECT_FLUENTS)
            || problem.getRequirements().contains(RequireKey.PREFERENCES)
            ||
problem.getRequirements().contains(RequireKey.TIMED_INITIAL_LITERALS)
            || problem.getRequirements().contains(RequireKey.HIERARCHY))
            ? false
            : true;
}

public static void main(String[] args) {

    // Percorso alla directory dei benchmark
    final String myFiles = "src/main/java/fr/uga/pddl4j/AIproject/PDDL
FIles/";

    // Ottiene la configurazione predefinita dal planner
    PlannerConfiguration config = myPlanner.getDefaultConfiguration();

    // DOMINIO E PROBLEMA
    config.setProperty(myPlanner.DOMAIN_SETTING,
"C:/Users/user/Downloads/ProgettoIA/logistics.pddl");
    config.setProperty(myPlanner.PROBLEM_SETTING,
"C:/Users/user/Downloads/ProgettoIA/instance3.pddl");
```

```
    // Imposta il timeout allocato per la ricerca.
    config.setProperty(myPlanner.TIME_OUT_SETTING, 10000);
    // Imposta il livello di log
    config.setProperty(myPlanner.LOG_LEVEL_SETTING, LogLevel.INFO);
    // Imposta l'euristica utilizzata per la ricerca
    config.setProperty(myPlanner.HEURISTIC_SETTING,
StateHeuristic.Name.MY_HEURISTIC);
    // Imposta il peso dell'euristica
    config.setProperty(myPlanner.WEIGHT_HEURISTIC_SETTING, 25);

    // Crea un'istanza del planner myPlanner con la configurazione
specificata
    final myPlanner planner = new myPlanner(config);
    // Esegue il planner e stampa la soluzione
    try {
        planner.solve();
    } catch (InvalidConfigurationException e) {
        e.printStackTrace();
    }
}
```

Oltre alla classe MyPlanner ci siamo serviti della classe Node della libreria PDDL4J.

## 2.5 Implementazione dell'euristica

Il passo seguente che è stato affrontato è l'implementazione di un'euristica personalizzata per guidare il processo di ricerca verso una soluzione ottimale in modo più efficiente. L'euristica svolge un ruolo fondamentale nei sistemi di pianificazione intelligente, poiché consente all'agente di valutare la bontà delle possibili azioni in base a una stima del costo per raggiungere lo stato obiettivo desiderato. In questa sezione, descriveremo in dettaglio l'implementazione della nostra euristica, illustrando il suo funzionamento, le considerazioni di progettazione e le scelte effettuate durante lo sviluppo. Abbiamo dunque, sviluppato una classe personalizzata denominata MyHeuristic. Questa classe, che si basa ovviamente sulla libreria PDDL4J, estende la classe RelaxedGraphHeuristic ereditandone alcune funzionalità chiave. Questa estensione ci consente di implementare una variante della classica euristica Adjusted Sum e adattarla alle specifiche esigenze del nostro dominio di applicazione. Di seguito viene riportato il codice della nostra classe:

```
public class MyHeuristic extends RelaxedGraphHeuristic {

/*
Euristica utilizzata per la risoluzione. È una variante di Adjusted Sum.
*/
```

```
    public MyHeuristic(final Problem problem) {
        super(problem);
        super.setAdmissible(true);
    }
//Calcola una stima del costo per raggiungere il goal dallo stato
corrente.
    @Override
    public int estimate(final State state, final Condition goal) {
        super.setGoal(goal);
// Espande il grafo di pianificazione rilassato fino al livello
desiderato.
        final int level = super.expandRelaxedPlanningGraph(state);
// Calcola la stima del costo. Se il goal è raggiungibile, utilizza
l'Adjusted Sum,
// altrimenti restituisce il massimo valore possibile (costo infinito).
        return super.isGoalReachable() ? super.getPositiveSumValue() +
(level - super.getPositiveMaxValue()) : Integer.MAX_VALUE;
    }
    @Override
    public double estimate(final Node node, final Condition goal) {
        return estimate((State) node, goal);
    }

}
```

È importante evidenziare che abbiamo aggiunto alla classe RelaxedGraphHeuristics i metodi `getPositiveSumValue` e `getPositiveMaxValue` per adattare l'euristica Adjusted Sum alle nostre esigenze specifiche. Questa personalizzazione è stata necessaria per migliorare l'accuratezza delle stime del costo di pianificazione nel nostro contesto di applicazione. Inoltre, una volta scritti e testati i metodi, abbiamo proceduto a ricompilare la classe e ad esportare il file JAR risultante.
Successivamente, abbiamo importato questo JAR all'interno delle librerie del nostro progetto di pianificazione. Questa procedura di integrazione ci ha permesso di utilizzare le personalizzazioni apportate alla libreria nella nostra implementazione, garantendo così una pianificazione più precisa e adattata alle specifiche esigenze del nostro agente.
Il metodo, `getPositiveSumValue` ci fornisce una valutazione della complessità della pianificazione. La somma dei livelli delle proposizioni positive richieste per il goal ci indica quanta "fatica" l'agente dovrà fare per soddisfare tutti i requisiti. Questa informazione aiuta a guidare il processo decisionale durante la pianificazione, poiché sappiamo quanto sia costoso raggiungere il goal.

```
protected final int getPositiveSumValue() {
    int value = 0;
    BitVector pGoal = super.getGoal().getPositiveFluents();
```

```
    for(int g = pGoal.nextSetBit(0); g >= 0; g = pGoal.nextSetBit(g + 1))
{
        value += this.pPropLevel[g];
    }

    return value;
}
```

D'altra parte, `getPositiveMaxValue` identifica il livello massimo tra le proposizioni positive richieste. Questo valore riflette la massima profondità a cui una proposizione positiva richiesta si trova nel grafo di pianificazione rilassato. Un valore elevato suggerisce che alcune parti del goal richiedono più passaggi per essere soddisfatte, influenzando così la selezione delle azioni.

```
protected final int getPositiveMaxValue() {
    int max = Integer.MIN_VALUE;
    BitVector pGoal = super.getGoal().getPositiveFluents();

    for(int g = pGoal.nextSetBit(0); g >= 0; g = pGoal.nextSetBit(g+1)) {

           int gl = this.pPropLevel[g];
           if (gl > max) {
                max = gl;
           }
     }

    return max;
}
```

## 2.6 Risultati ottenuti

Di seguito riportiamo, i risultati ottenuti per l'istanza 1 grazie al nostro planner basato sulla logica Iterative Deepening A* e che, come euristica, usa una variante di Adjusted Sum.

### Risultati istanza 1:

problem instantiation done successfully (123 actions, 61 fluents)

* Starting My IDWA* search
* My A* search succeeded

found plan as follows:

00: (  fill-box b5 r food depot) [0]
01: (  load-box b5 r depot car sl1) [0]
02: (move-robot-carrier r car depot loc1) [0]
03: (  unload-box b5 r loc1 sl1 car) [0]
04: (move-robot-carrier r car loc1 depot) [0]
05: (  fill-box b1 r drugs depot) [0]
06: (  load-box b1 r depot car sl1) [0]
07: (move-robot-carrier r car depot loc1) [0]
08: (  unload-box b1 r loc1 sl1 car) [0]
09: (  give-supplies b1 r p1 drugs loc1) [0]
10: (move-robot-carrier r car loc1 depot) [0]
11: (  fill-box b2 r food depot) [0]
12: (  load-box b2 r depot car sl4) [0]
13: (move-robot-carrier r car depot loc1) [0]
14: (  give-supplies b5 r p1 food loc1) [0]
15: (  unload-box b2 r loc1 sl4 car) [0]
16: (  give-supplies b2 r p3 food loc1) [0]
17: (move-robot-carrier r car loc1 depot) [0]
18: (  fill-box b3 r drugs depot) [0]
19: (  load-box b3 r depot car sl1) [0]
20: (move-robot-carrier r car depot loc1) [0]
21: (  unload-box b3 r loc1 sl1 car) [0]
22: (  give-supplies b3 r p2 drugs loc1) [0]

time spent:  0,03 seconds parsing
             0,06 seconds encoding
             1,85 seconds searching
             1,94 seconds total time

memory used:  0,68 MBytes for problem representation
              0,00 MBytes for searching
              0,68 MBytes total

Process finished with exit code 0

## **Risultati istanza 2**:

parsing domain file "logistics.pddl" done successfully
parsing problem file "instance2.pddl" done successfully

problem instantiation done successfully (594 actions, 101 fluents)

* Starting My IDWA* search
* My A* search succeeded

found plan as follows:

00: (          fill-box b3 r2 tools depot) [0]
01: (       load-box b3 r1 depot car1 sl11) [0]
02: (move-robot-carrier r1 car1 depot loc1) [0]
03: (      unload-box b3 r1 loc1 sl11 car1) [0]
04: (    give-supplies b3 r1 p1 tools loc1) [0]
05: (          fill-box b1 r2 tools depot) [0]
06: (move-robot-carrier r1 car1 loc1 depot) [0]
07: (       load-box b1 r1 depot car1 sl11) [0]
08: (move-robot-carrier r1 car1 depot loc4) [0]
09: (      unload-box b1 r1 loc4 sl11 car1) [0]
10: (    give-supplies b1 r1 p5 tools loc4) [0]
11: (       load-box b1 r1 loc4 car1 sl11) [0]
12: (move-robot-carrier r1 car1 loc4 depot) [0]
13: (          fill-box b2 r1 drugs depot) [0]
14: (       load-box b2 r1 depot car2 sl21) [0]
15: (move-robot-carrier r1 car2 depot loc1) [0]
16: (      unload-box b2 r1 loc1 sl21 car2) [0]
17: (    give-supplies b2 r1 p2 drugs loc1) [0]
18: (     unload-box b1 r2 depot sl11 car1) [0]
19: (       load-box b2 r1 loc1 car2 sl21) [0]
20: (move-robot-carrier r1 car2 loc1 depot) [0]
21: (          fill-box b1 r1 drugs depot) [0]
22: (       load-box b1 r1 depot car1 sl11) [0]
23: (move-robot-carrier r1 car1 depot loc2) [0]
24: (      unload-box b1 r1 loc2 sl11 car1) [0]
25: (    give-supplies b1 r1 p3 drugs loc2) [0]
26: (       load-box b1 r1 loc2 car1 sl11) [0]
27: (move-robot-carrier r1 car1 loc2 depot) [0]
28: (     unload-box b1 r1 depot sl11 car1) [0]
29: (          fill-box b1 r1 drugs depot) [0]
30: (       load-box b1 r1 depot car1 sl11) [0]
31: (move-robot-carrier r1 car1 depot loc3) [0]
32: (      unload-box b1 r1 loc3 sl11 car1) [0]
33: (    give-supplies b1 r1 p4 drugs loc3) [0]
34: (       load-box b1 r1 loc3 car1 sl11) [0]
35: (move-robot-carrier r1 car1 loc3 depot) [0]
36: (     unload-box b2 r2 depot sl21 car2) [0]
37: (          fill-box b2 r1 drugs depot) [0]
38: (       load-box b2 r1 depot car2 sl21) [0]
39: (move-robot-carrier r1 car2 depot loc4) [0]
40: (      unload-box b2 r1 loc4 sl21 car2) [0]
41: (    give-supplies b2 r1 p5 drugs loc4) [0]
42: (       load-box b2 r1 loc4 car2 sl21) [0]
43: (move-robot-carrier r1 car2 loc4 depot) [0]
44: (     unload-box b1 r1 depot sl11 car1) [0]
45: (          fill-box b1 r1 drugs depot) [0]
46: (       load-box b1 r1 depot car1 sl11) [0]
47: (move-robot-carrier r1 car1 depot loc5) [0]
48: (      unload-box b1 r1 loc5 sl11 car1) [0]
49: (    give-supplies b1 r1 p6 drugs loc5) [0]
50: (       load-box b1 r1 loc5 car1 sl11) [0]

```
51: (move-robot-carrier r1 car1 loc5 depot) [0]
52: (     unload-box b1 r1 depot sl11 car1) [0]
53: (           fill-box b1 r1 food depot) [0]
54: (       load-box b1 r1 depot car1 sl11) [0]
55: (move-robot-carrier r1 car1 depot loc1) [0]
56: (      unload-box b1 r1 loc1 sl11 car1) [0]
57: (     give-supplies b1 r1 p1 food loc1) [0]
58: (        load-box b1 r1 loc1 car1 sl11) [0]
59: (move-robot-carrier r1 car1 loc1 depot) [0]
60: (     unload-box b1 r1 depot sl11 car1) [0]
61: (           fill-box b1 r1 food depot) [0]
62: (       load-box b1 r1 depot car1 sl11) [0]
63: (move-robot-carrier r1 car1 depot loc3) [0]
64: (      unload-box b1 r1 loc3 sl11 car1) [0]
65: (     give-supplies b1 r1 p4 food loc3) [0]
66: (        load-box b1 r1 loc3 car1 sl11) [0]
67: (move-robot-carrier r1 car1 loc3 depot) [0]
68: (     unload-box b1 r1 depot sl11 car1) [0]
69: (           fill-box b1 r1 food depot) [0]
70: (       load-box b1 r1 depot car1 sl11) [0]
71: (move-robot-carrier r1 car1 depot loc4) [0]
72: (      unload-box b1 r1 loc4 sl11 car1) [0]
73: (     give-supplies b1 r1 p5 food loc4) [0]
74: (        load-box b1 r1 loc4 car1 sl11) [0]
75: (move-robot-carrier r1 car1 loc4 depot) [0]
76: (     unload-box b1 r1 depot sl11 car1) [0]
77: (           fill-box b1 r1 food depot) [0]
78: (       load-box b1 r1 depot car1 sl11) [0]
79: (move-robot-carrier r1 car1 depot loc5) [0]
80: (      unload-box b1 r1 loc5 sl11 car1) [0]
81: (     give-supplies b1 r1 p6 food loc5) [0]
82: (     unload-box b2 r2 depot sl21 car2) [0]
83: (          fill-box b2 r2 tools depot) [0]
84: (       load-box b2 r2 depot car2 sl21) [0]
85: (move-robot-carrier r2 car2 depot loc5) [0]
86: (      unload-box b2 r1 loc5 sl21 car2) [0]
87: (    give-supplies b2 r1 p6 tools loc5) [0]

 time spent:       0,03 seconds parsing
                   0,09 seconds encoding
                  29,01 seconds searching
                  29,13 seconds total time

 memory used:      2,66 MBytes for problem representation
                   0,00 MBytes for searching
                   2,66 MBytes total

 Process finished with exit code 0
```

## **Risultati istanza 3**:

parsing domain file "logistics.pddl" done successfully
parsing problem file "instance3.pddl" done successfully

problem instantiation done successfully (1064 actions, 143 fluents)

* Starting My IDWA* search
* My A* search succeeded

found plan as follows:

000: (  fill-box b1 r1 food depot) [0]
001: (  load-box b1 r1 depot car2 sl22) [0]
002: (move-robot-carrier r1 car2 depot loc1) [0]
003: (  unload-box b1 r1 loc1 sl22 car2) [0]
004: (  give-supplies b1 r1 p1 food loc1) [0]
005: (  fill-box b3 r2 tools depot) [0]
006: (  load-box b3 r2 depot car1 sl11) [0]
007: (move-robot-carrier r1 car2 loc1 depot) [0]
008: (move-robot-carrier r1 car1 depot loc1) [0]
009: (  unload-box b3 r1 loc1 sl11 car1) [0]
010: (  give-supplies b3 r1 p1 tools loc1) [0]
011: (  fill-box b2 r2 drugs depot) [0]
012: (  load-box b2 r2 depot car2 sl21) [0]
013: (move-robot-carrier r1 car1 loc1 depot) [0]
014: (move-robot-carrier r1 car2 depot loc1) [0]
015: (  unload-box b2 r1 loc1 sl21 car2) [0]
016: (  give-supplies b2 r1 p2 drugs loc1) [0]
017: (  load-box b2 r1 loc1 car2 sl22) [0]
018: (move-robot-carrier r1 car2 loc1 depot) [0]
019: (  fill-box b4 r1 drugs depot) [0]
020: (  load-box b4 r1 depot car1 sl11) [0]
021: (move-robot-carrier r1 car1 depot loc2) [0]
022: (  unload-box b4 r1 loc2 sl11 car1) [0]
023: (  give-supplies b4 r1 p3 drugs loc2) [0]
024: (  load-box b4 r1 loc2 car1 sl12) [0]
025: (  unload-box b2 r2 depot sl22 car2) [0]
026: (move-robot-carrier r1 car1 loc2 depot) [0]
027: (  fill-box b2 r1 drugs depot) [0]
028: (  load-box b2 r1 depot car2 sl21) [0]
029: (move-robot-carrier r1 car2 depot loc3) [0]
030: (  unload-box b2 r1 loc3 sl21 car2) [0]
031: (  give-supplies b2 r1 p4 drugs loc3) [0]
032: (  unload-box b4 r2 depot sl12 car1) [0]
033: (  load-box b2 r1 loc3 car2 sl21) [0]
034: (move-robot-carrier r1 car2 loc3 depot) [0]
035: (  fill-box b4 r1 drugs depot) [0]
036: (  load-box b4 r1 depot car1 sl11) [0]
037: (move-robot-carrier r1 car1 depot loc4) [0]
038: (  unload-box b4 r1 loc4 sl11 car1) [0]
039: (  give-supplies b4 r1 p5 drugs loc4) [0]
040: (  load-box b4 r1 loc4 car1 sl11) [0]

041: (move-robot-carrier r1 car1 loc4 depot) [0]
042: ( unload-box b4 r2 depot sl11 car1) [0]
043: ( fill-box b4 r1 drugs depot) [0]
044: ( load-box b4 r1 depot car1 sl11) [0]
045: (move-robot-carrier r1 car1 depot loc5) [0]
046: ( unload-box b4 r1 loc5 sl11 car1) [0]
047: ( give-supplies b4 r1 p6 drugs loc5) [0]
048: ( unload-box b2 r2 depot sl21 car2) [0]
049: ( load-box b4 r1 loc5 car1 sl11) [0]
050: (move-robot-carrier r1 car1 loc5 depot) [0]
051: ( fill-box b2 r1 drugs depot) [0]
052: ( load-box b2 r1 depot car2 sl21) [0]
053: (move-robot-carrier r1 car2 depot loc6) [0]
054: ( unload-box b2 r1 loc6 sl21 car2) [0]
055: ( give-supplies b2 r1 p7 drugs loc6) [0]
056: ( load-box b2 r1 loc6 car2 sl21) [0]
057: (move-robot-carrier r1 car2 loc6 depot) [0]
058: ( unload-box b2 r1 depot sl21 car2) [0]
059: ( fill-box b2 r1 drugs depot) [0]
060: ( load-box b2 r1 depot car2 sl21) [0]
061: (move-robot-carrier r1 car2 depot loc7) [0]
062: ( unload-box b2 r1 loc7 sl21 car2) [0]
063: ( give-supplies b2 r1 p8 drugs loc7) [0]
064: ( load-box b2 r1 loc7 car2 sl21) [0]
065: (move-robot-carrier r1 car2 loc7 depot) [0]
066: ( unload-box b2 r1 depot sl21 car2) [0]
067: ( fill-box b2 r1 food depot) [0]
068: ( load-box b2 r1 depot car2 sl21) [0]
069: (move-robot-carrier r1 car2 depot loc3) [0]
070: ( unload-box b2 r1 loc3 sl21 car2) [0]
071: ( give-supplies b2 r1 p4 food loc3) [0]
072: ( load-box b2 r1 loc3 car2 sl21) [0]
073: (move-robot-carrier r1 car2 loc3 depot) [0]
074: ( unload-box b2 r1 depot sl21 car2) [0]
075: ( fill-box b2 r1 food depot) [0]
076: ( load-box b2 r1 depot car2 sl21) [0]
077: (move-robot-carrier r1 car2 depot loc4) [0]
078: ( unload-box b2 r1 loc4 sl21 car2) [0]
079: ( give-supplies b2 r1 p5 food loc4) [0]
080: ( load-box b2 r1 loc4 car2 sl22) [0]
081: ( unload-box b4 r2 depot sl11 car1) [0]
082: (move-robot-carrier r1 car2 loc4 depot) [0]
083: ( fill-box b4 r1 food depot) [0]
084: ( load-box b4 r1 depot car1 sl11) [0]
085: (move-robot-carrier r1 car1 depot loc5) [0]
086: ( unload-box b4 r1 loc5 sl11 car1) [0]
087: ( give-supplies b4 r1 p6 food loc5) [0]
088: ( load-box b4 r1 loc5 car1 sl11) [0]
089: (move-robot-carrier r1 car1 loc5 depot) [0]
090: ( unload-box b2 r1 depot sl22 car2) [0]
091: ( fill-box b2 r1 food depot) [0]

092: (  load-box b2 r1 depot car2 sl21) [0]
093: (move-robot-carrier r1 car2 depot loc6) [0]
094: (  unload-box b2 r1 loc6 sl21 car2) [0]
095: (  give-supplies b2 r1 p7 food loc6) [0]
096: (  load-box b2 r1 loc6 car2 sl21) [0]
097: (move-robot-carrier r1 car2 loc6 depot) [0]
098: (  unload-box b4 r2 depot sl11 car1) [0]
099: (  fill-box b4 r1 food depot) [0]
100: (  load-box b4 r1 depot car1 sl11) [0]
101: (move-robot-carrier r1 car1 depot loc7) [0]
102: (  unload-box b4 r1 loc7 sl11 car1) [0]
103: (  give-supplies b4 r1 p8 food loc7) [0]
104: (  unload-box b2 r2 depot sl21 car2) [0]
105: (  fill-box b2 r2 tools depot) [0]
106: (  load-box b2 r2 depot car2 sl21) [0]
107: (move-robot-carrier r2 car2 depot loc4) [0]
108: (  unload-box b2 r2 loc4 sl21 car2) [0]
109: (  load-box b4 r1 loc7 car1 sl11) [0]
110: (move-robot-carrier r1 car1 loc7 depot) [0]
111: (  give-supplies b2 r2 p5 tools loc4) [0]
112: (  unload-box b4 r1 depot sl11 car1) [0]
113: (  fill-box b4 r1 tools depot) [0]
114: (  load-box b4 r1 depot car1 sl11) [0]
115: (move-robot-carrier r1 car1 depot loc5) [0]
116: (  unload-box b4 r1 loc5 sl11 car1) [0]
117: (  load-box b2 r2 loc4 car2 sl21) [0]
118: (move-robot-carrier r2 car2 loc4 depot) [0]
119: (  give-supplies b4 r1 p6 tools loc5) [0]
120: (  unload-box b2 r2 depot sl21 car2) [0]
121: (  fill-box b2 r2 tools depot) [0]
122: (  load-box b2 r2 depot car2 sl21) [0]
123: (move-robot-carrier r2 car2 depot loc6) [0]
124: (  unload-box b2 r2 loc6 sl21 car2) [0]
125: (  give-supplies b2 r2 p7 tools loc6) [0]
126: (  load-box b2 r2 loc6 car2 sl21) [0]
127: (move-robot-carrier r2 car2 loc6 depot) [0]
128: (  unload-box b2 r2 depot sl21 car2) [0]
129: (  fill-box b2 r2 tools depot) [0]
130: (  load-box b2 r2 depot car2 sl21) [0]
131: (move-robot-carrier r2 car2 depot loc7) [0]
132: (  unload-box b2 r2 loc7 sl21 car2) [0]
133: (  give-supplies b2 r2 p8 tools loc7) [0]

time spent:  0,03 seconds parsing
  0,15 seconds encoding
  147,88 seconds searching
  148,06 seconds total time

memory used: 4,66 MBytes for problem representation
0,00 MBytes for searching
4,66 MBytes total

Process finished with exit code 0

# 3. Temporal Planning & Robotics

## 3.1 Temporal Planning

Si è richiesto di aggiornare il file del dominio precedentemente creato per includere la gestione del tempo utilizzando PDDL nella sua versione 2.1. Per soddisfare questa specifica, sono state introdotte le 'durative actions'. Di seguito è presentato il file del dominio modificato con commenti solo nelle sezioni che differiscono dalla versione originale discussa in precedenza. Inoltre, ai requirements precedentemente definiti è stato aggiunto il supporto per le 'durative actions' e i 'fluents' (variabili numeriche).

```
(define (domain logistics)
   (:requirements :adl :typing :universal-preconditions :durative-actions
:fluents)
```

Sono stati introdotti nuovi predicati nel dominio per gestire lo stato di occupazione del robot e lo stato di disponibilità degli spazi di carico. In particolare, è stato aggiunto il predicato ready per definire quando il robot è in uno stato in cui può effettuare azioni. Inoltre, è stato introdotto il predicato unavailable per indicare quando uno spazio di carico è già occupato e non può essere utilizzato. Questi cambiamenti sono stati apportati per migliorare la chiarezza della definizione del dominio e per evitare l'uso del costrutto not nelle condizioni, il che potrebbe causare problemi di compatibilità nell'esecuzione del solver.

```
(ready ?r - robot)
(unavailable ?sl - slot)
```

Sono state introdotte due nuove funzioni nel dominio: weight e total-weight. Queste funzioni sono state aggiunte per gestire il peso delle forniture e il peso totale del carrello nel problema. La funzione weight prende in input una fornitura e restituisce il suo peso, mentre la funzione total-weight restituisce il peso totale del carrello. L'aggiunta di queste funzioni consente di calcolare e gestire meglio il peso delle forniture nel contesto del problema.

```
(:functions
        (weight ?sup - supplies)
        (total-weight ?car - carrier)
)
```

I costrutti "at start" e "over all" nelle precondizioni delle azioni sono utilizzati per specificare quando una determinata condizione deve essere vera all'inizio della pianificazione ("at start") o in ogni passo dell'esecuzione ("over all").

I costrutti "at start" e "at end" negli "effect" delle azioni sono utilizzati per specificare quando una modifica di stato deve avvenire all'inizio dell'esecuzione dell'azione ("at start") o alla fine dell'esecuzione dell'azione ("at end").

Dopo aver chiarito il significato dei costrutti utilizzati per la modellazione delle durative actions, verrà presentato il codice che implementa tali azioni nel dominio PDDL 2.1.

```
(:durative-action fill-box
        :parameters (?b - box ?r - robot ?sup - supplies ?loc - location
        )
        :duration (= ?duration 1)
        :condition (and
            (at start (ready ?r))
            (over all (at ?r ?loc))
            (over all (at ?sup ?loc))
            (over all (at ?b ?loc))
            (at start (is-empty ?b))
        )
        :effect (and
            (at end (contains ?b ?sup))
            (at start (not(is-empty ?b)))
            (at start (not(ready ?r)))
            (at end (ready ?r))
        )
    )
    (:durative-action give-supplies
        :parameters (?b - box ?r - robot ?p - person ?sup - supplies ?loc
- location
        )
        :duration (= ?duration 1)
        :condition (and
            (at start (ready ?r))
            (over all (at ?r ?loc))
            (over all (at ?p ?loc))
            (over all (at ?b ?loc))
            (over all (needs ?p ?sup))
            (at start (contains ?b ?sup))
        )
        :effect (and
            (at start (not(ready ?r)))
            (at end (ready ?r))
            (at start (not(contains ?b ?sup)))
            (at end (is-empty ?b))
```

```
            (at end (has ?p ?sup))
        )
    )


    (:durative-action load-box
        :parameters (?b - box ?r - robot ?loc - location ?car - carrier
?sl - slot ?sup - supplies
        )
        :duration (= ?duration 1)
        :condition (and (at start (ready ?r))
            (over all (contains ?b ?sup))
            (over all (at ?r ?loc))
            (over all (at ?car ?loc))
            (at start (at ?b ?loc))
            (over all (belongs ?sl ?car))
            (at start (available ?sl))
        )
        :effect (and
            (at start (not(available ?sl)))
            (at start (unavailable ?sl))
            (at start (not(ready ?r)))
            (at end (ready ?r))
            (at start (not(at ?b ?loc)))
            (at end (on ?b ?car))
            (at end (increase
                    (total-weight ?car)
                    (weight ?sup)))
        )
    )

    (:durative-action move-robot
        :parameters (?r - robot ?from - location ?to - location
        )
        :duration (= ?duration 2)
        :condition (and
            (at start (ready ?r))
            (at start (at ?r ?from))
        )
        :effect (and
            (at start (not(ready ?r)))
            (at end (ready ?r))
            (at start (not(at ?r ?from)))
            (at end (at ?r ?to))
        )
    )

    (:durative-action move-robot-carrier
```

```
        :parameters (?r - robot ?car - carrier ?from - location ?to -
location
        )
        :duration (= ?duration (* 2 (total-weight ?car)))
        :condition (and
            (at start (ready ?r))
            (at start (at ?r ?from))
            (at start (at ?car ?from))
        )
        :effect (and
            (at start (not(ready ?r)))
            (at end (ready ?r))
            (at start (not(at ?r ?from)))
            (at end (at ?r ?to))
            (at start (not(at ?car ?from)))
            (at end (at ?car ?to))
        )
    )

    (:durative-action unload-box
        :parameters (?b - box ?r - robot ?loc - location ?sl - slot ?car
- carrier ?sup - supplies
        )
        :duration (= ?duration 1)
        :condition (and
            (over all (contains ?b ?sup))
            (at start (ready ?r))
            (over all (at ?r ?loc))
            (over all (at ?car ?loc))
            (over all (belongs ?sl ?car))
            (over all (unavailable ?sl))
            (at start (on ?b ?car))
        )
        :effect (and
            (at start (not(ready ?r)))
            (at end (ready ?r))
            (at end (at ?b ?loc))
            (at start (not (on ?b ?car)))
            (at end (not (unavailable ?sl)))
            (at end (available ?sl))
            (at end (decrease
                    (total-weight ?car)
                    (weight ?sup)))
        )
    )

)
```

A questo punto sono state apportate alcune modifiche all'istanza 1 del problema per riflettere accuratamente le nuove caratteristiche del dominio. L'unico cambiamento significativo nell'istanza 1 è l'introduzione del peso per il cibo, le medicine e gli strumenti. Questo influisce sulla pianificazione poiché ora il piano deve tener conto del peso degli oggetti sul carrello. Di conseguenza, il tempo necessario per spostare il carrello varierà in base al peso degli oggetti presenti su di esso. Questa modifica aggiunge un elemento di complessità alla pianificazione, poiché il planner deve considerare il peso come una variabile aggiuntiva nella stima dei costi delle azioni.

```
(:init
        …
        (= (weight drugs) 1)
        (= (weight food) 2)
        (= (weight tools) 3)
        (= (total-weight car) 1)
        (ready r)
)
```

Il peso totale del carrier è stato inizializzato a 1 per evitare che la durata temporale dell'azione di spostamento del robot e del carrier con il carico vuoto fosse calcolata come 0, poiché tale durata viene determinata moltiplicando il peso totale per una durata arbitraria selezionata.

### 3.1.1 Scelta del Planner per la Risoluzione del Problema

A questo punto, abbiamo affrontato la necessità di selezionare un planner dalla libreria PlanUtils che fosse in grado di risolvere il nostro problema. Abbiamo condotto una serie di test con l'obiettivo di individuare la soluzione migliore, tenendo conto della qualità del piano generato e dei tempi necessari per ottenere l'output desiderato. I test sono stati effettuati su POPF e LPG-td.

### 3.1.2 POPF

POPF (Partially Observable Probabilistic Planning): POPF è un planner che si concentra sulla pianificazione probabilistica in ambienti parzialmente osservabili. È stato progettato per gestire problemi PDDL (Planning Domain Definition Language) che coinvolgono incertezza e parzialità nelle informazioni disponibili. POPF utilizza un approccio di pianificazione basato su grafi e cerca di ottimizzare il tempo necessario per trovare una soluzione. Di seguito è riportato il benchmark dell'applicazione di POPF:

Number of literals: 59
Constructing lookup tables: [10%] [20%] [30%] [40%] [50%] [60%] [70%] [80%] [90%] [100%]
Post filtering unreachable actions: [10%] [20%] [30%] [40%] [50%] [60%] [70%] [80%] [90%] [100%]

[01;34mNo analytic limits found, not considering limit effects of goal-only operators[00m
All the ground actions in this problem are compression-safe
Initial heuristic = 9.000
b (8.000 | 4.002)b (7.000 | 8.005)b (6.000 | 9.006)b (5.000 | 11.007)b (4.000 | 12.008)b (3.000 | 20.014)b (2.000 | 21.015)b (1.000 | 29.021);;;; Solution Found
; States evaluated: 241
; Cost: 30.022
; Time 0.32
0.000: (fill-box b1 r food depot)  [1.000]
1.001: (load-box b1 r depot car sl1 food)  [1.000]
2.002: (move-robot-carrier r car depot loc1)  [2.000]
4.003: (unload-box b1 r loc1 sl1 car food)  [1.000]
5.004: (move-robot-carrier r car loc1 depot)  [2.000]
7.005: (fill-box b2 r drugs depot)  [1.000]
8.006: (load-box b2 r depot car sl1 drugs)  [1.000]
9.007: (move-robot-carrier r car depot loc1)  [2.000]
11.008: (unload-box b2 r loc1 sl1 car drugs)  [1.000]
12.009: (give-supplies b1 r p3 food loc1)  [1.000]
13.010: (move-robot-carrier r car loc1 depot)  [2.000]
15.011: (fill-box b3 r food depot)  [1.000]
16.012: (load-box b3 r depot car sl1 food)  [1.000]
17.013: (move-robot-carrier r car depot loc1)  [2.000]
19.014: (unload-box b3 r loc1 sl1 car food)  [1.000]
20.015: (give-supplies b3 r p1 food loc1)  [1.000]
21.016: (give-supplies b2 r p2 drugs loc1)  [1.000]
22.017: (move-robot-carrier r car loc1 depot)  [2.000]
24.018: (fill-box b4 r drugs depot)  [1.000]
25.019: (load-box b4 r depot car sl1 drugs)  [1.000]
26.020: (move-robot-carrier r car depot loc1)  [2.000]
28.021: (unload-box b4 r loc1 sl1 car drugs)  [1.000]
29.022: (give-supplies b4 r p1 drugs loc1)  [1.000]

Possiamo notare che il piano non è sicuramente un piano ottimale ma comunque accettabile per la risoluzione del problema, soprattutto osservando le notevoli tempistiche raggiunte.

### 3.1.3 LPG-td

LPG-td (LPG with Temporal Dependencies): LPG-td è una variante del planner LPG (LPG-td) che gestisce problemi di pianificazione temporale. È in grado di gestire durative actions, ovvero azioni che richiedono un certo periodo di tempo per essere eseguite. LPG-td utilizza una pianificazione basata su grafi e si sforza di produrre soluzioni ottimali o vicine all'ottimo in modo efficiente. Di seguito è riportato il benchmark dell'applicazione di LPG-td:

```
NUMERIC_THREATS_MODE: 0
; Command line: /home/aiguy/.planutils/packages/lpg-td/bin/lpg-td -o logistics.pddl -f instance1.pddl -v off
-noout -quality
Parsing domain file:  domain 'LOGISTICS' defined ... done.
Parsing problem file:  problem 'INSTANCE1' defined ... done.
…
   Time: (ACTION) [action Duration; action Cost]
0.0003: (FILL-BOX B4 R FOOD DEPOT) [D:1.0000; C:1.0000]
1.0005: (FILL-BOX B2 R FOOD DEPOT) [D:1.0000; C:1.0000]
2.0008: (LOAD-BOX B2 R DEPOT CAR SL3 FOOD) [D:1.0000; C:1.0000]
3.0010: (FILL-BOX B5 R DRUGS DEPOT) [D:1.0000; C:1.0000]
4.0012: (FILL-BOX B1 R DRUGS DEPOT) [D:1.0000; C:1.0000]
5.0015: (FILL-BOX B3 R DRUGS DEPOT) [D:1.0000; C:1.0000]
6.0017: (LOAD-BOX B4 R DEPOT CAR SL4 FOOD) [D:1.0000; C:1.0000]
7.0020: (LOAD-BOX B3 R DEPOT CAR SL1 DRUGS) [D:1.0000; C:1.0000]
8.0022: (LOAD-BOX B1 R DEPOT CAR SL2 DRUGS) [D:1.0000; C:1.0000]
9.0025: (MOVE-ROBOT-CARRIER R CAR DEPOT LOC1) [D:14.0000; C:1.0000]
23.0028: (UNLOAD-BOX B2 R LOC1 SL3 CAR FOOD) [D:1.0000; C:1.0000]
24.0030: (UNLOAD-BOX B4 R LOC1 SL4 CAR FOOD) [D:1.0000; C:1.0000]
25.0033: (GIVE-SUPPLIES B4 R P3 FOOD LOC1) [D:1.0000; C:1.0000]
26.0035: (GIVE-SUPPLIES B2 R P1 FOOD LOC1) [D:1.0000; C:1.0000]
27.0037: (UNLOAD-BOX B3 R LOC1 SL1 CAR DRUGS) [D:1.0000; C:1.0000]
28.0040: (GIVE-SUPPLIES B3 R P2 DRUGS LOC1) [D:1.0000; C:1.0000]
29.0042: (UNLOAD-BOX B1 R LOC1 SL2 CAR DRUGS) [D:1.0000; C:1.0000]
30.0045: (GIVE-SUPPLIES B1 R P1 DRUGS LOC1) [D:1.0000; C:1.0000]

Solution found:
Total time:      10.26
Search time:     0.41
Actions:         18
Execution cost:  18.00
Duration:        31.000
Plan quality:    18.000
Plan file:       plan_instance1.pddl_1.SOL
```

Anche in questo caso non abbiamo ottenuto un numero ottimo di azioni, ma abbiamo comunque un tempo ancora migliore e un output di qualità accettabile. Procediamo quindi con il piano appena ottenuto per la fase successiva.

## 3.2 Robotics Planning

Una volta affrontata la fase di Temporal Planning, possiamo simulare il comportamento di un agente robotico attraverso il framework Ros, tramite la libreria PlanSys2. Il piano generato è parzialmente compatibile con la semantica accettata: essa prevede infatti operazioni definite come:

<time>: (<action>) [<duration>]

**Fake actions:**

Dopo aver adattato il piano generato alla sintassi richiesta, possiamo ridefinire in C++ le azioni delle sezioni precedenti come fake actions. Questa ridefinizione permette al sistema di simulare i tempi che un agente robotico impiegherebbe nell'esecuzione delle azioni identificate dal planner. Durante la definizione della durata temporale per ciascuna azione, è stata prestata attenzione a garantire che queste durate siano coerenti con quelle specificate nel file di dominio di riferimento.

Essendo le diverse implementazioni molto simili, verrà presentato il codice relativo a una sola fake: in particolare, saranno dettagliati i segmenti di codice relativi all'azione relativa al riempimento della scatola (fill-box):

```
#include <memory>
#include <algorithm>
#include "plansys2_executor/ActionExecutorClient.hpp"
#include "rclcpp/rclcpp.hpp"
#include "rclcpp_action/rclcpp_action.hpp"
using namespace std::chrono_literals;

class FillBox : public plansys2::ActionExecutorClient
{
public:
    FillBox()
        : plansys2::ActionExecutorClient("fill_box", 200ms)
    {
        progress_ = 0.0;
    }

private:
    void do_work()
    {
        std::vector<std::string> arguments = get_arguments();

        if (progress_ < 1.0)
        {
            progress_ += 0.2;
            send_feedback(progress_, "Robot " + arguments[1] + " is filling box
" + arguments[0] + " with supplies " + arguments[2] + " at location " +
arguments[3] + "\n");
        }
        else
```

```
        {
            finish(true, 1.0, "Robot " + arguments[1] + " has successfully
filled box " + arguments[0] + " with supplies " + arguments[2] + " at location
" + arguments[3] + "\n");
            progress_ = 0.0;
            std::cout << std::endl;
        }

        std::cout << "Robot " + arguments[1] + " is filling box " +
arguments[0] + " with supplies " + arguments[2] + " at location " +
arguments[3] + "\n" << std::flush;
    }

    float progress_;
};

int main(int argc, char **argv)
{
    rclcpp::init(argc, argv);
    auto node = std::make_shared<FillBox>();
    node->set_parameter(rclcpp::Parameter("action_name", "fill_box"));
    node-
>trigger_transition(lifecycle_msgs::msg::Transition::TRANSITION_CONFIGURE);
    rclcpp::spin(node->get_node_base_interface());
    rclcpp::shutdown();
    return 0;
}
```

Sulla base di questo metodo sono stati poi realizzati in maniera analoga i metodi relativi alle altre azioni.

**Launch-file:**

È necessario ora definire il launch-file in Python che genera una configurazione di lancio per un sistema ROS 2 (Robot Operating System 2). In particolare, configura il lancio di vari nodi e specifica i parametri associati e il namespace.

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

def generate_launch_description():
    # Get the launch directory
    example_dir = get_package_share_directory("progetto_ia")
    namespace = LaunchConfiguration("namespace")

    declare_namespace_cmd = DeclareLaunchArgument(
        "namespace", default_value="", description="Namespace"
```

```
)
plansys2_cmd = IncludeLaunchDescription(
    PythonLaunchDescriptionSource(
        os.path.join(
            get_package_share_directory("plansys2_bringup"),
            "launch",
            "plansys2_bringup_launch_monolithic.py",
        )
    ),
    launch_arguments={
        "model_file": example_dir + "/pddl/logistics.pddl",
        "namespace": namespace,
    }.items(),
)
# Specifica le azioni
fillbox_cmd = Node(
    package='progetto_ia',
    executable='fill_box',
    name='fillbox',
    namespace=namespace,
    output='screen',
    parameters=[]
)
loadbox_cmd = Node(
    package='progetto_ia',
    executable='load_box',
    name='loadbox',
    namespace=namespace,
    output='screen',
    parameters=[]
)
moverobot_cmd = Node(
    package='progetto_ia',
    executable='move_robot',
    name='moverobot',
    namespace=namespace,
    output='screen',
    parameters=[]
)
moverobotcarrier_cmd = Node(
    package='progetto_ia',
    executable='move_robot_carrier',
    name='moverobotcarrier',
    namespace=namespace,
    output='screen',
    parameters=[]
)
givesupplies_cmd = Node(
    package='progetto_ia',
    executable='give_supplies',
    name='givesupplies',
    namespace=namespace,
    output='screen',
    parameters=[]
)
```

```
    unloadbox_cmd = Node(
        package='progetto_ia',
        executable='unload_box',
        name='unloadbox',
        namespace=namespace,
        output='screen',
        parameters=[]
    )

    ld = LaunchDescription()
    ld.add_action(declare_namespace_cmd)

    # Dichiarazione delle opzioni di lancio
    ld.add_action(plansys2_cmd)
    ld.add_action(fillbox_cmd)
    ld.add_action(loadbox_cmd)
    ld.add_action(moverobot_cmd)
    ld.add_action(moverobotcarrier_cmd)
    ld.add_action(givesupplies_cmd)
    ld.add_action(unloadbox_cmd)

    return ld
```

**Command file:**

Per comodità si definisce un file contenente i comandi da eseguire all'interno del terminale PlanSys2: questo file rappresenta una conversione del file di istanza da PDDL alla libreria, in modo da non dover manualmente inserire ogni oggetto definito dal problema.

```
set instance r robot
set instance car carrier
set instance b1 box
set instance b2 box
set instance b3 box
set instance b4 box
set instance b5 box
set instance depot location
set instance loc1 location
set instance sl1 slot
set instance sl2 slot
set instance sl3 slot
set instance sl4 slot
set instance drugs supplies
set instance food supplies
set instance tools supplies
set instance p1 person
set instance p2 person
set instance p3 person
set function (= (weight drugs) 1)
set function (= (weight food) 2)
set function (= (weight tools) 3)
set function (= (total-weight car) 1)
set predicate (ready r)
set predicate (at b1 depot)
```

```
set predicate (at b2 depot)
set predicate (at b3 depot)
set predicate (at b4 depot)
set predicate (at b5 depot)
set predicate (at r depot)
set predicate (at car depot)
set predicate (at food depot)
set predicate (at drugs depot)
set predicate (at tools depot)
set predicate (at p1 loc1)
set predicate (at p2 loc1)
set predicate (at p3 loc1)
set predicate (needs p1 food)
set predicate (needs p1 drugs)
set predicate (needs p2 drugs)
set predicate (needs p3 food)
set predicate (is-empty b1)
set predicate (is-empty b2)
set predicate (is-empty b3)
set predicate (is-empty b4)
set predicate (is-empty b5)
set predicate (available sl1)
set predicate (available sl2)
set predicate (available sl3)
set predicate (available sl4)
set predicate (belongs sl1 car)
set predicate (belongs sl2 car)
set predicate (belongs sl3 car)
set predicate (belongs sl4 car)
set goal (and (has p1 food) (has p1 drugs) (has p2 drugs) (has p3 food))
```

## CMakeList file:

Il file CMakeLists.txt è utilizzato nell'ambiente ROS2 per specificare le dipendenze e come i file sorgente C++ relativi alle fake actions devono essere compilati e distribuiti all'interno del sistema.

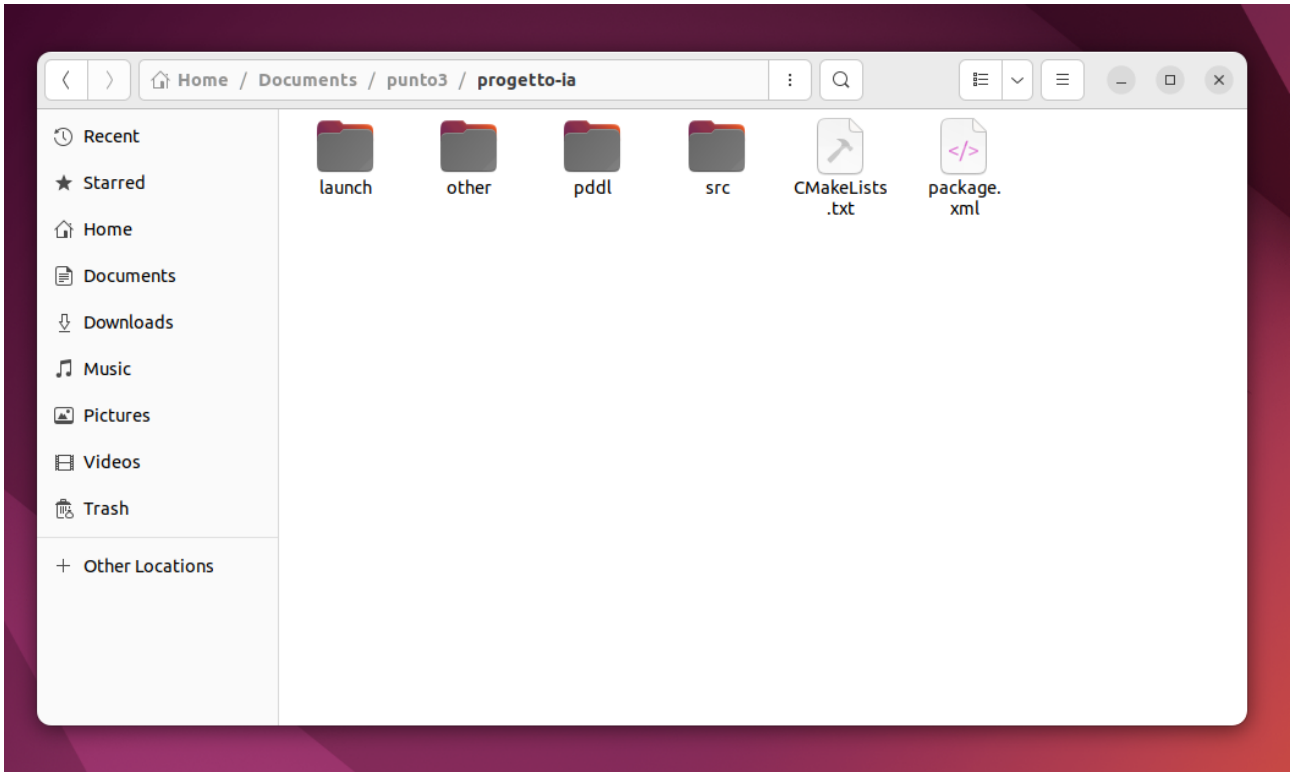
```
cmake_minimum_required(VERSION 3.5)
project(progetto_ia)

find_package(ament_cmake REQUIRED)
find_package(rclcpp REQUIRED)
find_package(rclcpp_action REQUIRED)
find_package(plansys2_msgs REQUIRED)
find_package(plansys2_executor REQUIRED)

set(CMAKE_CXX_STANDARD 17)
set(dependencies
  rclcpp
  rclcpp_action
  plansys2_msgs
  plansys2_executor
)
```

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

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

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

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

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

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

install(DIRECTORY launch pddl DESTINATION share/${PROJECT_NAME})
install(TARGETS
  fill_box
  give_supplies
  load_box
  move_robot
  move_robot_carrier
  unload_box
  ARCHIVE DESTINATION lib
  LIBRARY DESTINATION lib
  RUNTIME DESTINATION lib/${PROJECT_NAME}
)

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

ament_export_dependencies(${dependencies})
ament_package()
```

## 4. Deliverables

**Struttura del package:**

Una volta completata la creazione dei file di configurazione necessari all'esecuzione, si ottiene un package così strutturato:




- La cartella principale è stata nominata **progetto-ia**; è a sua volta contenuta in una cartella, che contiene anche i file pddl relativi alla sezione precedente (benchmark dei planner, dominio e istanza)
- Nella cartella **launch** vengono inseriti i file relativi all'avvio del programma, cioè file *commands* e file *plansys2_project_launch.py*
- Nella cartella **pddl** sono stati inseriti i file *logistics.pddl* e *instance1_solution.plan*, rispettivamente file di dominio e file contenente il piano, definiti nella sezione 3.1.
- Nella cartella **src** sono contenuti i file che descrivono le fake actions in C++
- Il file *CMakeLists.txt* è situato direttamente nella cartella di progetto
- Sono stati aggiunti in seguito i file contenenti i risultati (*output.txt*) e i file di info relativi all'installazione e alla build (*install_info.txt* e *build_info.txt*)

Nota: Il package contiene anche le cartelle di Build, Install e Log generate dopo l'esecuzione del processo