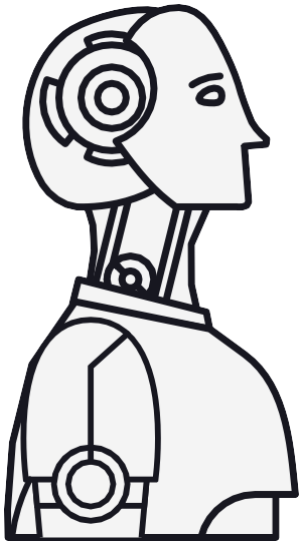
UNIVERSITÀ DELLA CALABRIA
DIPARTIMENTO DI **INGEGNERIA INFORMATICA, MODELLISTICA, ELETTRONICA E SISTEMISTICA**
DIMES

A.A. 2023/2024

Planning logistico per servizi di emergenza

Progetto d'esame di Intelligenza Artificiale
Corso di laurea magistrale in ingegneria informatica



Studentessa Gaia Assunta Bertolino, mat. 242590

1. Introduzione

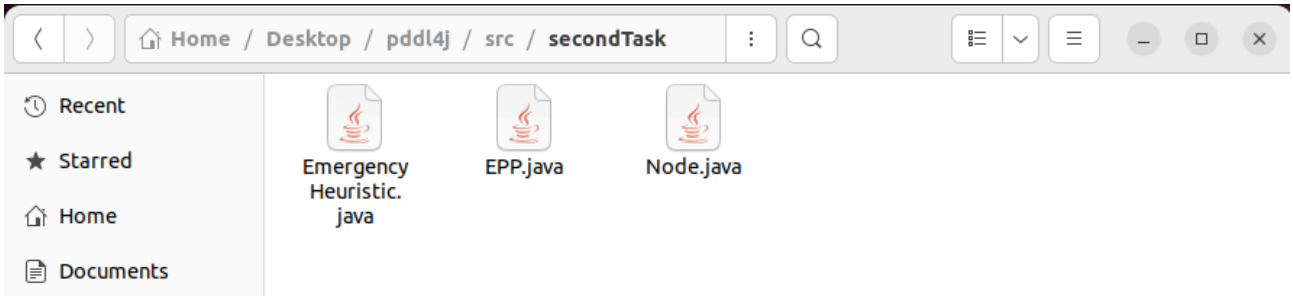
1.1 Obiettivo

Lo scopo del progetto è implementare un sistema di intelligenza artificiale per la modellazione, il planning classico, il planning temporale e il planning robotico di un sistema basato su agenti autonomi. In particolare, lo scenario rappresenta un problema di organizzazione logistica in cui un numero di persone ferite devono essere raggiunte da servizi di emergenza, i quali possono trasportare diversi oggetti.

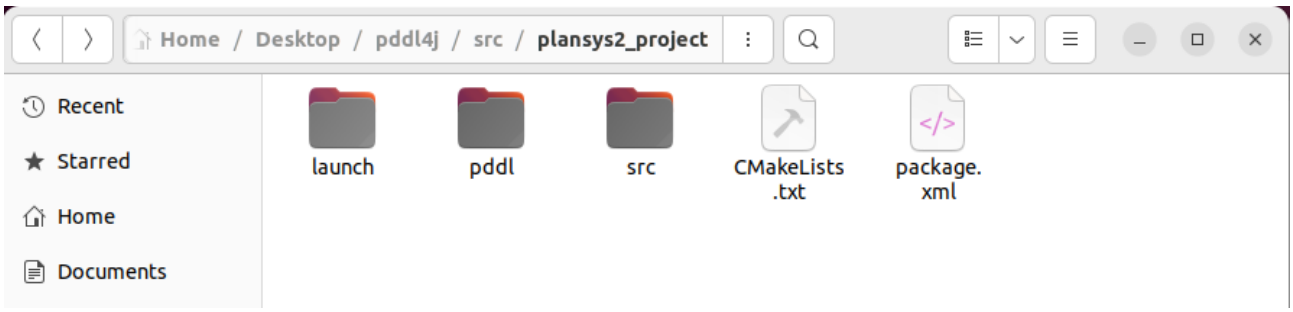
1.2 Directory

I file allegati sono organizzati nella seguente struttura:

- Cartella secondTask
  - la sottocartella classes contente le classi di implementazione del planner e della funzione euristica custom
  - la sottocartella pddl contente i file relativi alla formulazione del dominio e dei problemi del secondo task



- Cartella plansys2_project
  - il file CMakeLists.txt e il file package.xml che riporta le dipendenze
  - la sottocartella src riportante i file C++ che realizzano le fake actions
  - la sottocartella launch che contiene il file commands e il file di lancio plansys2_project_launch.py
  - la sottocartella pddl che contiene il dominio e il problema con azioni durative e il plan ottenuto dall'applicazione di LPG-TD



2. Modelling

2.1 Specifiche

Lo scopo della modellazione è implementare, in linguaggio apposito, quelle che sono le specifiche del problema. Nel caso in oggetto si è ricordo a PDDL, un linguaggio standard per definire problemi di pianificazione nell'intelligenza artificiale.

In particolare, per un problema è necessario redigere due file:

- Domain file, il quale specifica il dominio. Contiene dunque le informazioni sul mondo e descrive in che modo può cambiare uno stato
- Problem file, il quale specifica il problema. Contiene dunque gli oggetti coinvolti, lo stato iniziale e l'obiettivo da raggiungere.

Lo scenario presentato prevede che vengano modellate le seguenti condizioni:

- presenza di agenti robotici in grado di consegnare dei contenitori contenenti gli oggetti richiesti a dei feriti, i quali non possono muoversi dalla loro locazione
- ogni contenitore è in una locazione iniziale quale un deposito e può essere riempita con contenuto fra medicine, strumenti o cibo (se vuoto).
- una persona ferita ha o non ha ciascuno dei possibili oggetti, per cui bisogna mantenerne traccia in maniera distinta
- in una stessa locazione possono trovarsi più persone per cui è necessario scindere la locazione da chi vi si trova e mantenerne traccia
- è necessario per ogni agente tenere traccia dei contenitori in ciascun mezzo, sia della quantità che della tipologia
- non sono presenti strade o rotte ma gli agenti possono muoversi autonomamente tra le locazioni
- gli agenti possono fare uso di mezzi di trasporto la cui capacità è limitata ma relativa al singolo problema

Per ogni agente robotico sono necessarie le seguenti specifiche:

- può svuotare e riempire i contenitori, caricare o scaricare da un mezzo e spostarsi con o senza mezzo
- può riempire un contenitore vuoto se agente e box sono nella stessa locazione
- può caricare su un mezzo uno o più contenitori o scaricarli solo se agente, strumento ausiliario e contenitore si trovano nella stessa location.

2.2 Implementazione in PDDL

L'implementazione del dominio in linguaggio PDDL prevede la definizione di tipi, predicati e azioni.

2.2.1 Tipi

Per rappresentare gli oggetti descritti, si è ricorso all'implementazione dei seguenti tipi:

```
(:types agent box location person content veichle place)
```

- agent: oggetto che modella il robot
- box: oggetto che modella il contenitore
- location: oggetto che modella la posizione in cui possono trovarsi gli oggetti del dominio
- person: oggetto che modella una persona ferita
- content: oggetto che modella il contenuto del box
- veichle: oggetto che modella il mezzo di trasporto
- place: oggetto che modella lo spazio su un veichle nel quale è possibile caricare un box

2.2.2 Predicati

I predicati sono utilizzati per esprimere proprietà di interesse degli oggetti, le quali possono essere vere o false. Per esprimere le specifiche ricercate, si è ricorso all'implementazione dei seguenti predicati:

```
(:predicates (filled ?c - content ?b - box)
             (empty ?b - box)
             (hasContent ?p - person ?c - content)
             (veichlePlace ?p - place ?v - veichle)
             (placeAvailable ?p - place)
             (placeOccupied ?p - place)
             (boxLoaded ?b - box ?v - veichle)
             (at ?x - (either agent person box content veichle) ?l - location)
             (needContent ?c - content ?p - person)
             (noContent ?p - person ?c1 - content ?c2 - content )
             (gotContent ?p - person ?c1 - content ?c2 - content ))
```

- filled: vero se la box contiene il contenuto specificato
- empty: vero se la box è vuota
- hasContent: vero se la persona ha ricevuto il contenuto specificato
- veichlePlace: vero se lo slot appartiene al mezzo
- placeOccupied: vero se lo slot è occupato
- placeAvailable: vero se lo slot è disponibile
- boxLoaded: vero se la box è presente sul mezzo specificato
- at: vero se l'agente, la persona, la box, il contenuto o il mezzo si trova nella location specificata
- needContent: vero se la persona ha bisogno del contenuto
- noContent: vero se la persona non ha ricevuto nessuno dei due due contenuti
- gotContent: vero se la persona ha ricevuto almeno uno dei due contenuti

L'uso di predicati complementari quali placeOccupied/placeAvailable e noContent/gotContent si è reso necessario a causa dell'impossibilità di poter ricorrere all'uso del not nelle precondizioni delle azioni, di seguito illustrate, nell'ambiente utilizzato.

2.2.3 Azioni

Un'azione definisce una trasformazione da uno stato ad un altro. La definizione di un'azione è suddivisa in tre sezioni distinte:

- In *parameters* si definiscono gli oggetti coinvolti nell'azione
- In *precondition* si esprimono le condizioni che devono essere soddisfatte affinché l'azione venga applicata. Esse sono espresse attraverso la congiunzione o la disgiunzione di predicati
- La terza sezione è una scelta tra *effect* e *expansion*. In particolare, nella maggior parte dei pianificatori viene utilizzato *effect*, il cui scopo è di definire gli esiti dell'azione, attraverso l'uso di espressioni logiche e predicati.

Nell'ambito del progetto, si è ricorso all'implementazione delle seguenti azioni:

- fill: azione di riempimento di una scatola con del contenuto da parte di un agente.
  Le precondition verificano rispettivamente che
  - il contenuto si trovi nella posizione indicata
  - la scatola si trovi nella posizione indicata
  - l'agente si trovi nella posizione indicata
  - la scatola sia vuota

  Come effetto si avrà che
  - la scatola non sarà più vuota
  - la scatola sarà riempita con lo specifico contenuto

```
(:action fill
    :parameters (?l - location ?c - content ?b - box ?a - agent)
    :precondition ( and (at ?c ?l)
                        (at ?b ?l)
                        (at ?a ?l)
                        (empty ?b))
    :effect ( and  (filled ?c ?b)
                   (not(empty ?b))))
```

- charge: azione di carico della scatola box sul veicolo veichle.
  Le precondition verificano rispettivamente che
  - il mezzo abbia spazio per la scatola da caricare
  - lo spazio libero faccia riferimento al mezzo in oggetto
  - la scatola si trovi nella posizione indicata
  - l'agente si trovi nella posizione indicata
  - il mezzo si trovi nella posizione indicata

  Come effetto si avrà che
  - la scatola è stata caricata sul mezzo
  - la scatola non si trova più nella sua location originaria
  - il posto sul mezzo è stato riempito e non è più disponibile

```
(:action charge
     :parameters(?l - location ?b - box ?p - place
                 ?v - veichle ?a - agent ?c - content)
```

```
:precondition(and (placeAvailable ?p)
                  (veichlePlace ?p ?v)
                  (at ?b ?l)
                  (at ?a ?l)
                  (at ?v ?l)
                  (filled ?c ?b))
:effect(and(not(placeAvailable ?p))
        (placeOccupied ?p)
        (not (at ?b ?l))
        (boxLoaded ?b ?v)))
```

- discharge: azione di scarico del contenuto di una scatola dal mezzo. Tale azione è possibile solo nel caso in cui la box sia vuota.
  Le precondition verificano rispettivamente che
  - lo slot da cui scaricare fosse effettivamente occupato da una box
  - l'agente si trovi nella posizione indicata
  - il mezzo si trovi nella posizione indicata
  - la scatola da scaricare si trovi sul mezzo
  - lo slot da cui scaricare appartenga al mezzo
  - la scatola sia vuota

  Come effetto si avrà che
  - lo spazio da cui si è scaricato diviene libero
  - la scatola non si trova più sul mezzo
  - la scatola si trova nella locazione di scarico

```
(:action discharge
     :parameters (?l - location ?b - box ?p - place
                 ?v - veichle ?a - agent ?c - content)
     :precondition (and (placeOccupied ?p)
                    (at ?a ?l)
                    (at ?v ?l)
                    (boxLoaded ?b ?v)
                    (veichlePlace ?p ?v)
                    (empty ?b)
     )
     :effect ( and (placeAvailable ?p)
                   (not (placeOccupied ?p))
                   (at ?b ?l)
                   (not(boxLoaded ?b ?v))))
```

- moveAgent: azione di movimento di un agente senza mezzo verso una posizione.
  La precondition verifica che
  - l'agente si trovi nella posizione di partenza specificata

  Come effetto si avrà che
  - l'agente non si trova più nella posizione iniziale
  - l'agente si trova nella posizione di arrivo

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

- moveVeichle: azione di movimento di un mezzo da parte di un agente verso la posizione specificata.
  Le precondition verificano rispettivamente che
  - l'agente si trovi nella posizione indicata
  - il mezzo si trovi nella posizione indicata

  Come effetti si avrà che
  - l'agente non si trova più nella posizione iniziale
  - l'agente si trova nella posizione di arrivo
  - il mezzo non si trova più nella posizione iniziale
  - il mezzo si trova nella posizione di arrivo

```
(:action moveVeichle
    :parameters(?a - agent ?from ?to - location ?v - veichle)
    :precondition(and(at ?a ?from)
                     (at ?v ?from))
    :effect(and (not(at ?a ?from))
                (at ?a ?to)
                (not(at ?v ?from))
                (at ?v ?to)))
```

- satisfied: questo predicato è introdotto per rispettare il vincolo or richiesto per la risoluzione della seconda e terza istanza della sezione *classical planning*. Si tratta dunque di un'azione fittizia.
  Le precondition verificano rispettivamente che
  - la persona ha ricevuto uno dei due contenuti richiesti
  - la persona non abbia già ricevuto in precedenza uno dei due contenuti richiesti

  Come effetti si avrà che
  - la persona avrà ricevuto uno dei due contenuti richiesti

```
(:action satisfiedWithOne
    :parameters(?c1 - content ?c2 - content ?p - person)
    :precondition(and
                    (hasContent ?p ?c1)
                    (noContent ?p ?c1 ?c2))
    :effect(and
                (gotContent ?p ?c1 ?c2)
                (not (noContent ?p ?c1 ?c2))
                (not (noContent ?p ?c2 ?c1))
                (gotContent ?p ?c2 ?c1)))
```

3. Classical Planning

La pianificazione automatica è una branca che riguarda la realizzazione di strategie o sequenze di azioni per affrontare la risoluzione dei problemi. Per affrontare questa parte, si è ricorso alla modellazione dei problemi e alla successiva costruzione di una funzione euristica ad hoc per il dominio in oggetto in modo da ottenere un plan d'azione in tempi ragionevoli.

3.1 Modellazione dei problemi

La traccia prevede di affrontare tre casistiche differenti con complessità crescente; ciascuna di esse è descritta da una condizione iniziale e un obiettivo (goal):

Primo problema:

- Condizione iniziale:
  - vi sono cinque scatole localizzate in un deposito in cui si trovano anche tutti i contenuti necessari a soddisfare le necessità dei feriti
  - vi sono tre persone di cui p1 e p2 localizzate nella stessa posizione, con p1 che necessità di cibo e medicine, p2 di medicine e p3 di cibo.
  - è presente un solo agente e un solo mezzo che si trovano al deposito
  - un carrello può portare fino a quattro scatole
- Obiettivo: tutti i feriti devono ottenere ciò che gli serve
- Remider: non importa quale oggetto ricevano ma solo la categoria in cui esso ricade

```
(define (problem emergencyDelivery)
    ( :domain emergency )
    ( :objects
        dep loc1 loc2 - location
        b1 b2 b3 b4 b5 - box
        p1 p2 p3 - person
        a - agent
        veic - veichle
        pl1 pl2 pl3 pl4 - place
        drugs food tools - content)

    ( :init
        ( at a dep )
        ( at veic dep )

        ( at b1 dep )
        ( at b2 dep )
        ( at b3 dep )
        ( at b4 dep )
        ( at b5 dep )
```

```
        ( at food dep )
        ( at tools dep )
        ( at drugs dep )

        ( at p1 loc1 )
        ( at p2 loc1)
        ( at p3 loc2 )

        ( needContent food p1)
        ( needContent drugs p1)
        ( needContent drugs p2)
        ( needContent food p3)

        (empty b1)
        (empty b2)
        (empty b3)
        (empty b4)
        (empty b5)

        ( placeAvailable pl1 )
        ( placeAvailable pl2 )
        ( placeAvailable pl3 )
        ( placeAvailable pl4 )
        ( veichlePlace pl1 veic )
        ( veichlePlace pl2 veic )
        ( veichlePlace pl3 veic )
        ( veichlePlace pl4 veic )
    )
    ( :goal
        (and
            ( hasContent p1 food )
            ( hasContent p1 drugs )
            ( hasContent p2 drugs )
            ( hasContent p3 food )))
```

Secondo problema:

- Condizione iniziale:
  - sono presenti due agenti, due mezzi e tre scatole
  - ciascun mezzo può trasportare due scatole
  - sono presenti sei persone di cui p1 ha bisogno di cibo o strumenti, p2 e p3 di medicine, p4 di di medicine e cibo, p5 e p6 di tutto
- Obiettivo: tutti i feriti devono ottenere ciò che gli serve

Il secondo problema, così come il terzo, prevederebbe l'utilizzo di una clausola or per la persona p1. Tuttavia, come anticipato nella discussione del dominio, si è ricorso ad un'azione fittizia e a dei predicati ah hoc per gestire tale casistica.

```
( define (problem emergencyDelivery)
    ( :domain emergency )
    ( :objects
        b1 b2 b3 - box
        dep loc1 loc2 loc3 loc4 loc5 - location
        p1 p2 p3 p4 p5 p6 - person
        a1 a2 - agent
        drugs food tools - content
        veic1 veic2 - veichle
        pl1veic1 pl2veic1 pl1veic2 pl2veic2 - place)

    ( :init
        ( at a1 dep )
        ( at a2 dep )

        ( at b1 dep )
        ( at b2 dep )
        ( at b3 dep )

        ( at veic1 dep )
        ( at veic2 dep )

        ( at food dep )
        ( at drugs dep )
        ( at tools dep )

        ( at p1 loc1 )
        ( at p2 loc1 )
        ( at p3 loc2 )
        ( at p4 loc3 )
        ( at p5 loc4 )
        ( at p6 loc5 )

        (empty b1)
        (empty b2)
        (empty b3)

        ( needContent food p1)
        ( needContent tools p1)
        (noContent p1 food tools)
        (noContent p1 tools food)
        ( needContent drugs p2)
        ( needContent drugs p3)
        ( needContent food p4)
        ( needContent drugs p4)
        ( needContent food p5)
        ( needContent drugs p5)
        ( needContent tools p5)
```

```
        ( needContent food p6)
        ( needContent drugs p6)
        ( needContent tools p6)

        ( placeAvailable pl1veic1 )
        ( placeAvailable pl2veic1 )
        ( placeAvailable pl1veic2 )
        ( placeAvailable pl2veic2 )

        ( veichlePlace pl1veic1 veic1 )
        ( veichlePlace pl2veic1 veic1 )
        ( veichlePlace pl1veic2 veic2 )
        ( veichlePlace pl2veic2 veic2 )
    )
    ( :goal
        (and
            ( gotContent p1 food tools)
            ( gotContent p1 tools food)
            ( hasContent p2 drugs )
            ( hasContent p3 drugs )
            ( hasContent p4 drugs )
            ( hasContent p4 food )
            ( hasContent p5 food )
            ( hasContent p5 drugs )
            ( hasContent p5 tools )
            ( hasContent p6 food )
            ( hasContent p6 tools )
            ( hasContent p6 drugs ))))
```

Terzo problema:

- Condizione iniziale:
  - sono presenti quattro scatole
  - sono presenti 8 persone di cui p1 ha bisogno di cibo o strumenti, p2 e p3 di medicine, p4 di medicine e cibo, p5, p6, p7 e p8 hanno bisogno di tutto
- Obiettivo: tutti i feriti devono ottenere ciò che gli serve

```
( define (problem emergencyDelivery )
    ( :domain emergency )
    ( :objects
        b1 b2 b3 b4 - box
        p1 p2 p3 p4 p5 p6 p7 p8 - person
        a1 a2 - agent
        veic1 veic2 - veichle
        dep loc1 loc2 loc3 loc4 loc5 loc6 loc7 - location
        pl1veic1 pl2veic1 pl1veic2 pl2veic2 - place
        drugs food tools - content)
    ( :init
        ( at a1 dep )
```

```
( at a2 dep )

( at b1 dep )
( at b2 dep )
( at b3 dep )
( at b4 dep )

( at food dep )
( at drugs dep )
( at tools dep )

( at veic1 dep )
( at veic2 dep )

( at p1 loc1 )
( at p2 loc1 )
( at p3 loc2 )
( at p4 loc3 )
( at p5 loc4 )
( at p6 loc5 )
( at p7 loc6 )
( at p8 loc7 )

(empty b1)
(empty b2)
(empty b3)
(empty b4)

( needContent food p1 )
( needContent tools p1 )

( needContent drugs p2 )

( needContent drugs p3 )

( needContent food p4 )
( needContent drugs p4 )

( needContent food p5 )
( needContent drugs p5 )
( needContent tools p5 )

( needContent food p6 )
( needContent drugs p6 )
( needContent tools p6 )

( needContent food p7 )
( needContent drugs p7 )
```

```
    ( needContent tools p7 )

    ( needContent food p8 )
    ( needContent drugs p8 )
    ( needContent tools p8 )


    ( veichlePlace pl1veic1 veic1 )
    ( veichlePlace pl2veic1 veic1 )
    ( veichlePlace pl1veic2 veic2 )
    ( veichlePlace pl2veic2 veic2 )

    ( placeAvailable pl1veic1 )
    ( placeAvailable pl2veic1 )
    ( placeAvailable pl1veic2 )
    ( placeAvailable pl2veic2 )

    (noContent p1 food tools)
    (noContent p1 tools food))
( :goal
    (and
        ( gotContent p1 food tools)
        ( gotContent p1 tools food)
        ( hasContent p2 drugs )

        ( hasContent p3 drugs )

        ( hasContent p4 food )
        ( hasContent p4 drugs )

        ( hasContent p5 drugs )
        ( hasContent p5 food )
        ( hasContent p5 tools )

        ( hasContent p6 food )
        ( hasContent p6 drugs )
        ( hasContent p6 tools )

        ( hasContent p7 food )
        ( hasContent p7 drugs )
        ( hasContent p7 tools )

        ( hasContent p8 drugs )
        ( hasContent p8 food )
        ( hasContent p8 tools ))))
```

3.2 Planner custom

Per l'individuazione di un piano d'azione in grado di risolvere i problemi visti, è necessario l'utilizzo di un planner ovvero di un pianificatore delle azioni che portano da uno stato iniziale ad un goal. Inoltre, i problemi discussi risultano essere difficilmente risolvibili in poco tempo e con poche risorse senza ricorrere all'utilizzo di un algoritmo informato, anche noto come algoritmo di ricerca guidata, il quale utilizza informazioni aggiuntive sul problema per guidare la ricerca verso una soluzione più efficiente ed efficace. Queste informazioni possono essere fornite sotto forma di euristiche o di conoscenza specifica del dominio del problema. Gli algoritmi informati esplorano lo spazio di ricerca in modo più mirato, selezionando le azioni o i nodi da esaminare in base a tali informazioni, riducendo così il numero di ricerche inutili e migliorando le prestazioni complessive dell'algoritmo. Un'euristica è considerata ammissibile quando fornisce una stima uguale o inferiore al costo effettivo per raggiungere la destinazione. In altre parole, un'euristica ammissibile non sovrastima mai il costo rimanente. Questa proprietà è fondamentale per garantire l'individuazione della soluzione ottimale.

Per realizzare il planner di tipo custom, si è ricorso alla personalizzazione di una classe Java di base chiamata EPP che estende l'interfaccia Abstract Planner. La classe è stata ottenuta ricorrendo all'uso della libreria PDDL4J e realizza l'override dei metodi instantiate, il quale si occupa di stanziare il problema, solve, il quale si occupa di applicare l'algoritmo di risoluzione del problema invocando un metodo custom customAstar, e isSupported, il quale verifica che il problema rispetti i requirements necessari a poterlo elaborare. In particolare, il metodo customAstar utilizza la funzione euristica custom nell'algoritmo informato A*; infatti, l'algoritmo A* utilizza due valutazioni per selezionare il prossimo nodo da esplorare: la valutazione del costo reale g(n) e la valutazione euristica h(n). A ogni iterazione, l'algoritmo seleziona il nodo con il costo complessivo più basso (la somma di g(n) e h(n)) tra tutti i nodi raggiungibili e continua a esplorare i nodi fino a quando non raggiunge il nodo di destinazione o esaurisce tutti i nodi raggiungibili. Quando raggiunge il nodo di destinazione, l'algoritmo ha trovato un percorso ottimale.

EPP.java

```
@CommandLine.Command(name = "EPP",
    version = "EPP 1.0",
    description = "Solves a specified planning problem using A* search strategy.
The heuristic function is a custom.",
    sortOptions = false,
    mixinStandardHelpOptions = true,
    headerHeading = "Usage:%n",
    synopsisHeading = "%n",
    descriptionHeading = "%nDescription:%n%n",
    parameterListHeading = "%nParameters:%n",
    optionListHeading = "%nOptions:%n")
```

```
public class EPP extends AbstractPlanner {
    //The class logger
    private static final Logger LOGGER = LogManager.getLogger(EPP.class.getName());

    /**
     * The HEURISTIC property used for planner configuration.
     */
   public static final String HEURISTIC_SETTING = "HEURISTIC";

   /**
    * The default value of the HEURISTIC property used for planner configuration.
    */
   public static final StateHeuristic.Name DEFAULT_HEURISTIC =
StateHeuristic.Name.FAST_FORWARD;

   /**
    * The WEIGHT_HEURISTIC property used for planner configuration.
    */
   public static final String WEIGHT_HEURISTIC_SETTING = "WEIGHT_HEURISTIC";

   /**
    * The default value of the WEIGHT_HEURISTIC property used for planner
configuration.
    */
   public static final double DEFAULT_WEIGHT_HEURISTIC = 1.0;

   /**
    * The weight of the heuristic.
    */
   private double heuristicWeight;

   /**
    * The name of the heuristic used by the planner.
    */
   private StateHeuristic.Name heuristic;

   /**
    * Creates a new A* search planner with the default configuration.
    */
   public EPP() {
       this(EPP.getDefaultConfiguration());
   }

        /**
      * Creates a new A* search planner with a specified configuration.
      *
      * @param configuration the configuration of the planner.
      */
```

```
      public EPP(final PlannerConfiguration configuration) {
        super();
        this.setConfiguration(configuration);
    }

    /**
      * Set the name of heuristic used by the planner to the solve a planning
problem.
      *
      * @param heuristic the name of the heuristic.
      */
      @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;
    }

    /**
      * Returns the name of the heuristic used by the planner to solve a planning
problem.
      *
      * @return the name of the heuristic used by the planner to solve a planning
problem.
      */
      public final StateHeuristic.Name getHeuristic() {
        return this.heuristic;
    }

    /**
     * Returns the weight of the heuristic.
     *
     * @return the weight of the heuristic.
     */
    public final double getHeuristicWeight() {
        return this.heuristicWeight;
    }

     /**
      * Checks the planner configuration and returns if the configuration is valid.
      * A configuration is valid if (1) the domain and the problem files exist and
      * can be read, (2) the timeout is greater than 0, (3) the weight of the
      * heuristic is greater than 0 and (4) the heuristic is a not null.
      *
      * @return <code>true</code> if the configuration is valid <code>false</code>
otherwise.
```

```
     */
     public boolean hasValidConfiguration() {
       return super.hasValidConfiguration()
               && this.getHeuristicWeight() > 0.0
               && this.getHeuristic() != null;
   }

   /**
     * Sets the configuration of the planner. If a planner setting is not defined
in
     * the specified configuration, the setting is initialized with its default
value.
     *
     * @param configuration the configuration to set.
     */
     @Override
     public void setConfiguration(final PlannerConfiguration configuration) {
         super.setConfiguration(configuration);
         if (configuration.getProperty(EPP.WEIGHT_HEURISTIC_SETTING) == null) {
             this.setHeuristicWeight(EPP.DEFAULT_WEIGHT_HEURISTIC);
         } else {
             this.setHeuristicWeight(Double.parseDouble(configuration.getProperty(
                     EPP.WEIGHT_HEURISTIC_SETTING)));
         }
         if (configuration.getProperty(EPP.HEURISTIC_SETTING) == null) {
             this.setHeuristic(EPP.DEFAULT_HEURISTIC);
         } else {

this.setHeuristic(StateHeuristic.Name.valueOf(configuration.getProperty(
                     EPP.HEURISTIC_SETTING)));
         }
     }

     /**
      * This method return the default arguments of the planner.
      *
      * @return the default arguments of the planner.
      * @see PlannerConfiguration
      */
     public static PlannerConfiguration getDefaultConfiguration() {
         PlannerConfiguration config = Planner.getDefaultConfiguration();
         config.setProperty(EPP.HEURISTIC_SETTING,
EPP.DEFAULT_HEURISTIC.toString());
         config.setProperty(EPP.WEIGHT_HEURISTIC_SETTING,
                 Double.toString(EPP.DEFAULT_WEIGHT_HEURISTIC));
         return config;
     }
```

```
    /**
     * Sets the weight of the heuristic.
     *
     * @param weight the weight of the heuristic. The weight must be greater than
0.
     * @throws IllegalArgumentException if the weight is strictly less than 0.
     */
    @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;
    }

    //Instantiates the planning problem from a parsed problem.
    @Override
    public Problem instantiate(DefaultParsedProblem problem) {
        final Problem pb = new DefaultProblem(problem);
        pb.instantiate();
        return pb;
    }
    /**
      * Returns the configuration of the planner.
      *
      * @return the configuration of the planner.
      */
      @Override
      public PlannerConfiguration getConfiguration() {
          final PlannerConfiguration config = super.getConfiguration();
          config.setProperty(EPP.HEURISTIC_SETTING,
this.getHeuristic().toString());
          config.setProperty(EPP.WEIGHT_HEURISTIC_SETTING,
Double.toString(this.getHeuristicWeight()));
          return config;
      }
    //Search a solution plan to a specified domain and problem using A*.
    //@param problem the problem to solve.
    //@return the plan found or null if no plan was found.
    @Override
    public Plan solve(final Problem problem) throws ProblemNotSupportedException {
        LOGGER.info("* Starting search \n");
        Plan plan = null;

        try {
            final long start = System.currentTimeMillis();
```

```
            plan = this.customAstar(problem);
            final long end = System.currentTimeMillis();
            if (plan != null) {
                LOGGER.info("* Search succeeded\n");
                //this.getStatistics().setTimeToSearch(end-start);
            } else {
                LOGGER.info("* Search failed\n");
            }
            this.getStatistics().setTimeToSearch(end-start);
        } catch (ProblemNotSupportedException e) {
            LOGGER.error("Not supported problem");
            e.printStackTrace();
        }
        finally {
            // Return the plan found or null if the search fails.
            return plan;
        }
    }

    /**
      * Search a solution plan for a planning problem using an A* search strategy.
      *
      * @param problem the problem to solve.
      * @return a plan solution for the problem or null if there is no solution
      * @throws ProblemNotSupportedException if the problem to solve is not
supported by the planner.
      */
    public Plan customAstar( Problem problem) throws ProblemNotSupportedException {
        //Requirements
        if (!this.isSupported(problem)) { throw new
ProblemNotSupportedException("Cannot solve the problem"); }

        //Euristic
        final EmergencyHeuristic heuristic = new EmergencyHeuristic(problem);

        //Initial state
        final State init = new State(problem.getInitialState());
        //Nodes list
        final Set<Node> exploredNodes = new HashSet<>();
        //Priority queue of visited node
        final double weight = this.getHeuristicWeight();
        final PriorityQueue<Node> toExplore = new PriorityQueue<>(100, new
Comparator<Node>() {
            public int compare(Node n1, Node n2) {
                double w1 = weight*n1.getHeuristic() + n1.getCost();
                double w2 = weight*n2.getHeuristic() + n2.getCost();
                return Double.compare(w1,w2);
            }
```

```
        });

        //Root
        final Node root = new Node(init, null, -1, 0,
(problem.getGoal().getNegativeFluents().stream().count() +
problem.getGoal().getPositiveFluents().stream().count())*2);
        toExplore.add(root);
        Plan plan = null;

        final int timeout = 600 * 1000;
        long time = 0;
        final long begin = System.currentTimeMillis();
        //Search beginning
        while (!toExplore.isEmpty() && plan == null && time < timeout) {

            final Node current = toExplore.poll();
            exploredNodes.add(current);

            // Check if goal is reached
            if(current.satisfy(problem.getGoal())) {
                long m1 = Runtime.getRuntime().totalMemory();
                long m2 = Runtime.getRuntime().freeMemory();
                this.getStatistics().setMemoryUsedToSearch(m1 - m2);
                return this.returnPlan(problem, current);
            }
            else {
                //Apply actions which belong to the node
                for(int i=0; i<problem.getActions().size(); i++) {
                    Action a = problem.getActions().get(i);
                    // Check if action can be applied to the current node
                    if(a.isApplicable(current)) {
                        Node next = new Node(current);
                        // Apply effects
                        final List<ConditionalEffect> effects =
a.getConditionalEffects();
                        for (ConditionalEffect condEffect : effects) {
                            if(current.satisfy(condEffect.getCondition())) {
                                next.apply(condEffect.getEffect());
                            }
                        }
                        //Setting of informations for the child node
                        final double c = current.getCost() +1;
                        if (!exploredNodes.contains(next)) {
                            next.setCost(c);
                            next.setParent(current);
                            next.setAction(i);
                            next.setHeuristic(heuristic.estimate(next,
problem.getGoal(), a, current.getHeuristic()));
```

```
                        toExplore.add(next);
                    }
                }
            }
        }
        // Compute the searching time
        time = System.currentTimeMillis() - begin;
    }
    return plan;
}

private Plan returnPlan(final Problem problem, final Node node) {
    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;
}

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

//Returns if a specified problem is supported by the planner. Just ADL problem
can be solved by this planner.
@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;
    }
}
```

Per modellare gli stati sottoforma di nodi, è stata inoltre implementata una classe java Node.java di seguito riportata

Node.java

```
//This class implements a node of the tree search.

public final class Node extends State {

//The parent node of this node.
    private Node parent;

//The action apply to reach this node.
    private int action;

//The cost to reach this node from the root node.
    private double cost;

//The estimated distance to the goal from this node.
    private double heuristic;

//The depth of the node.
    private int depth;

//Creates a new node from a specified state.
    public Node(State state) {
        super(state);
    }

//Creates a new node with a specified state, parent node, operator,cost and
heuristic value.
    public Node(State state, Node parent, int action, double cost, double
heuristic) {
        super(state);
        this.parent = parent;
        this.action = action;
        this.cost = cost;
        this.heuristic = heuristic;
        this.depth = -1;
    }
```

```
//Creates a new node with a specified state, parent node, operator, cost,depth and
heuristic value.
    public Node(State state, Node parent, int action, double cost, int depth,
double heuristic) {
        super(state);
        this.parent = parent;
        this.action = action;
        this.cost = cost;
        this.depth = depth;
        this.heuristic = heuristic;
    }

//Returns the action applied to reach the node.
    public final int getAction() {
        return this.action;
    }

//Sets the action applied to reach the node.
    public final void setAction(final int action) {
        this.action = action;
    }

//Returns the parent node of the node.
    public final Node getParent() {
        return parent;
    }

//Sets the parent node of the node.
    public final void setParent(Node parent) {
        this.parent = parent;
    }

//Returns the cost to reach the node from the root node.
    public final double getCost() {
        return cost;
    }

//Sets the cost needed to reach the node from the root node.
    public final void setCost(double cost) {
        this.cost = cost;
    }

//Returns the estimated distance to the goal from the node.
    public final double getHeuristic() {
        return heuristic;
    }
```

```
//Sets the estimated distance to the goal from the node.
    public final void setHeuristic(double estimates) {
        this.heuristic = estimates;
    }

//Returns the depth of this node.
    public int getDepth() {
        return this.depth;
    }

//Set the depth of this node.
    public void setDepth(final int depth) {
        this.depth = depth;
    }

//Returns the value of the heuristic function, i.e.,<code>this.node.getCost() +
this.node.getHeuristic()</code>.
    public final double getValueF(double weight) {
        return weight * this.heuristic + this.cost;
    }
}
```

Il planner fa uso di una funzione euristica realizzata ad hoc per il dominio in oggetto riportata nella classe EmergencyHeuristic.java. In particolare, il valore dell'euristica proposta è data da due quantità:

- la distanza dall'obiettivo misurata come il numero di vincoli del goal non ancora soddisfatti
- delle penalità associate a ciascuna azione affinché vengano promosse attività che avvicinano al goal e scartartate le altre

Partendo dal valore dell'euristica dello stato corrente, si calcola il valore di uno stato futuro in base all'azione che viene compiuta:

- per una azione di consegna si sottrae 1
- per una azione di carico o di riempimento di una box si sottrae 0.5
- per una azione di movimento con mezzo si somma 0.2
- per una azione di movimento senza mezzo si somma 0.1

Ne consegue che per le azioni di movimento il valore dell'euristica aumenta e ciò porta a scartare tali diramazioni; nel caso di una consegna, di carico o di riempimento invece l'euristica diminuisce, portando a scegliere tali azioni più di altre.
Il valore dell'euristica del nodo iniziale può essere definito almeno pari a n azioni di consegna, riempimento e caricamento, per cui pari a n*1 + n*0.5 + n*0.5 = 2*n.
Per come è strutturata, l'euristica risulta essere dunque ammissibile in quanto non si tengono in considerazione le azioni di scaricamento e di movimento, realizzando dunque un costo sicuramente minore dell'originale. Tuttavia, considerando che un algoritmo informativo è strutturato come f(n) = g(n) + w * h(n), ciò che rende inammissibile l'euristica e dunque rende potenzialmente subottimali le soluzioni individuate è l'applicazione del peso w alla funzione euristica maggiore di 2. Tuttavia, nel caso in esame è oggetto d'interesse

l'individuazione di una soluzione che sia un compromesso fra ottimalità e tempi di esecuzione della ricerca per cui l'applicazione di pesi importanti risulta essere la scelta effettuata.

EmergencyHeuristic.java

```
package fr.uga.pddl4j.examples.asp;

import fr.uga.pddl4j.parser.NamedTypedList;
import fr.uga.pddl4j.parser.Symbol;
import fr.uga.pddl4j.parser.SymbolType;
import fr.uga.pddl4j.parser.TypedSymbol;
import fr.uga.pddl4j.planners.statespace.search.Node;
import fr.uga.pddl4j.problem.*;
import fr.uga.pddl4j.problem.operator.Condition;
import fr.uga.pddl4j.util.BitVector;
import fr.uga.pddl4j.problem.operator.Action;
import java.util.*;
import java.util.stream.Collectors;
import fr.uga.pddl4j.heuristics.state.RelaxedGraphHeuristic;
import fr.uga.pddl4j.heuristics.state.StateHeuristic;


public final class EmergencyHeuristic extends RelaxedGraphHeuristic {

    public EmergencyHeuristic(Problem problem) {
        super(problem);
        super.setAdmissible(true);
    }

    public int estimate(State state, Condition goal) {
        System.out.println("Wrong Estimating");
        super.setGoal(goal);
        super.expandRelaxedPlanningGraph(state);
        return super.isGoalReachable() ? 0 : Integer.MAX_VALUE;
    }

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

    public double estimate(State state, Condition goal, Action a, double
oldHeuristicValue) {

        super.setGoal(goal);
        this.expandRelaxedPlanningGraph(state);

        double esteem = oldHeuristicValue;
```

```
        if(a.getName().equals("giveContent"))
            esteem--;
        else if (a.getName().equals("fill") || a.getName().equals("charge"))
            esteem-=0.5;
        else if (a.getName().equals("moveVeichle"))
            esteem += 0.1;
        else if (a.getName().equals("moveAgent"))
            esteem += 0.2;

         return super.isGoalReachable() ? esteem : Integer.MAX_VALUE;
    }
}
```

3.3 Analisi prima istanza

Nel caso della prima istanza, un buon compromesso è risultato essere un peso pari a 5. L'algoritmo ha così generato un plan di 26 azioni in poco più di 4 secondi.

```
* Starting search
* Search succeeded

found plan as follows:

00: (                fill dep drugs b1 a) [0]
01: (                fill dep drugs b2 a) [0]
02: (                fill dep drugs b3 a) [0]
03: (                fill dep drugs b4 a) [0]
04: (                fill dep drugs b5 a) [0]
05: (     charge dep b1 pl1 veic a drugs) [0]
06: (     charge dep b4 pl4 veic a drugs) [0]
07: (     charge dep b3 pl2 veic a drugs) [0]
08: (     charge dep b5 pl3 veic a drugs) [0]
09: (        moveveichle a dep loc1 veic) [0]
10: (givecontent loc1 drugs b1 a p2 veic) [0]
11: (givecontent loc1 drugs b5 a p1 veic) [0]
12: (        moveveichle a loc1 dep veic) [0]
13: (   discharge dep b1 pl2 veic a food) [0]
14: (                 fill dep food b1 a) [0]
15: (  discharge dep b5 pl4 veic a tools) [0]
16: (                fill dep drugs b5 a) [0]
17: (      charge dep b1 pl2 veic a food) [0]
18: (     charge dep b5 pl4 veic a drugs) [0]
19: (        moveveichle a dep loc2 veic) [0]
20: ( givecontent loc2 food b1 a p3 veic) [0]
21: (        moveveichle a loc2 dep veic) [0]
22: (  discharge dep b1 pl4 veic a drugs) [0]
23: (                 fill dep food b1 a) [0]
24: (      charge dep b1 pl4 veic a food) [0]
```

```
25: (        moveveichle a dep loc1 veic) [0]
26: ( givecontent loc1 food b1 a p1 veic) [0]

time spent:       0.03 seconds parsing
                  0.14 seconds encoding
                  4.07 seconds searching
                  4.25 seconds total time

memory used:      1.77 MBytes for problem representation
                 52.93 MBytes for searching
                 54.69  MBytes total
```

3.3 Analisi seconda istanza

Nel caso del secondo problema, un buon compromesso è risultato essere l'euristica con peso pari a 10, grazie al quale l'algoritmo ha richiesto un tempo di esecuzione di 2 minuti e mezzo, per un totale di 68 azioni.

```
* Starting search
* Search succeeded

found plan as follows:

00: (                   fill dep drugs b1 a1) [0]
01: (                   fill dep drugs b2 a1) [0]
02: (                   fill dep drugs b3 a1) [0]
03: (   charge dep b1 pl1veic1 veic1 a1 drugs) [0]
04: (   charge dep b2 pl2veic1 veic1 a1 drugs) [0]
05: (   charge dep b3 pl1veic2 veic2 a1 drugs) [0]
06: (           moveveichle a2 dep loc4 veic1) [0]
07: (   givecontent loc4 drugs b1 a2 p5 veic1) [0]
08: (           moveveichle a2 loc4 dep veic1) [0]
09: (discharge dep b1 pl2veic1 veic1 a2 tools) [0]
10: (                   fill dep drugs b1 a1) [0]
11: (   charge dep b1 pl2veic1 veic1 a2 drugs) [0]
12: (           moveveichle a1 dep loc5 veic1) [0]
13: (   givecontent loc5 drugs b1 a1 p6 veic1) [0]
14: (           moveveichle a1 loc5 dep veic1) [0]
15: (discharge dep b1 pl1veic1 veic1 a1 drugs) [0]
16: (                    fill dep food b1 a1) [0]
17: (    charge dep b1 pl1veic1 veic1 a2 food) [0]
18: (           moveveichle a1 dep loc1 veic1) [0]
19: (   givecontent loc1 drugs b2 a1 p2 veic1) [0]
20: (           moveveichle a1 loc1 dep veic1) [0]
21: (discharge dep b2 pl1veic1 veic1 a1 drugs) [0]
22: (                   fill dep drugs b2 a1) [0]
```

```
23: (   charge dep b2 pl1veic1 veic1 a2 drugs) [0]
24: (          moveveichle a2 dep loc3 veic2) [0]
25: (   givecontent loc3 drugs b3 a2 p4 veic2) [0]
26: (          moveveichle a2 loc3 dep veic2) [0]
27: (discharge dep b3 pl1veic2 veic2 a2 tools) [0]
28: (                   fill dep drugs b3 a1) [0]
29: (   charge dep b3 pl2veic2 veic2 a1 drugs) [0]
30: (          moveveichle a1 dep loc2 veic1) [0]
31: (   givecontent loc2 drugs b2 a1 p3 veic1) [0]
32: (          moveveichle a1 loc2 dep veic1) [0]
33: (discharge dep b2 pl1veic1 veic1 a1 drugs) [0]
34: (                   fill dep drugs b2 a1) [0]
35: (   charge dep b2 pl1veic1 veic1 a1 drugs) [0]
36: (          moveveichle a2 dep loc5 veic1) [0]
37: (    givecontent loc5 food b1 a2 p6 veic1) [0]
38: (          moveveichle a2 loc5 dep veic1) [0]
39: (discharge dep b1 pl2veic1 veic1 a2 tools) [0]
40: (                    fill dep food b1 a1) [0]
41: (    charge dep b1 pl2veic1 veic1 a1 food) [0]
42: (          moveveichle a2 dep loc1 veic1) [0]
43: (    givecontent loc1 food b1 a2 p1 veic1) [0]
44: (          moveveichle a2 loc1 dep veic1) [0]
45: (discharge dep b1 pl1veic1 veic1 a1 drugs) [0]
46: (                    fill dep food b1 a1) [0]
47: (    charge dep b1 pl1veic1 veic1 a1 food) [0]
48: (          moveveichle a1 dep loc3 veic1) [0]
49: (    givecontent loc3 food b1 a1 p4 veic1) [0]
50: (          moveveichle a1 loc3 dep veic1) [0]
51: (discharge dep b1 pl1veic1 veic1 a1 drugs) [0]
52: (                    fill dep food b1 a1) [0]
53: (    charge dep b1 pl1veic1 veic1 a1 food) [0]
54: (          moveveichle a1 dep loc4 veic1) [0]
55: (    givecontent loc4 food b1 a1 p5 veic1) [0]
56: (          moveveichle a1 loc4 dep veic1) [0]
57: (discharge dep b1 pl1veic1 veic1 a1 drugs) [0]
58: (                   fill dep tools b1 a1) [0]
59: (   charge dep b1 pl1veic2 veic2 a1 tools) [0]
60: (          moveveichle a2 dep loc5 veic2) [0]
61: (   givecontent loc5 tools b1 a2 p6 veic2) [0]
62: (          moveveichle a2 loc5 dep veic2) [0]
63: (discharge dep b1 pl1veic2 veic2 a1 drugs) [0]
64: (                   fill dep tools b1 a1) [0]
65: (   charge dep b1 pl1veic1 veic1 a1 tools) [0]
66: (          moveveichle a1 dep loc4 veic1) [0]
67: (         satisfiedwithone food tools p1) [0]
68: (   givecontent loc4 tools b1 a1 p5 veic1) [0]

time spent:       0.03 seconds parsing
```

```
                0.21 seconds encoding
              147.38 seconds searching
              147.61 seconds total time

memory used:    4.84 MBytes for problem representation
              329.85 MBytes for searching
              334.69  MBytes total
```

3.4 Analisi terza istanza

Per risolvere la terza istanza in tempi che fossero ragionevoli e al di sotto dei 10 minuti, si è optato per un peso dell'euristica pari a 15, il quale ha portato ad una risoluzione in poco più di 8 minuti, generando un pan di 112 azioni.

```
* Starting search
* Search succeeded

found plan as follows:

000: (                  fill dep drugs b1 a1) [0]
001: (                  fill dep drugs b2 a1) [0]
002: (                  fill dep drugs b3 a1) [0]
003: (                  fill dep drugs b4 a1) [0]
004: (   charge dep b1 pl1veic1 veic1 a1 drugs) [0]
005: (   charge dep b2 pl2veic1 veic1 a1 drugs) [0]
006: (   charge dep b3 pl1veic2 veic2 a1 drugs) [0]
007: (   charge dep b4 pl2veic2 veic2 a1 drugs) [0]
008: (           moveveichle a1 dep loc5 veic2) [0]
009: (   givecontent loc5 drugs b4 a1 p6 veic2) [0]
010: (           moveveichle a1 loc5 dep veic2) [0]
011: (discharge dep b4 pl1veic2 veic2 a1 drugs) [0]
012: (                  fill dep drugs b4 a1) [0]
013: (   charge dep b4 pl1veic2 veic2 a1 drugs) [0]
014: (           moveveichle a2 dep loc7 veic2) [0]
015: (   givecontent loc7 drugs b3 a2 p8 veic2) [0]
016: (           moveveichle a2 loc7 dep veic2) [0]
017: (discharge dep b3 pl1veic2 veic2 a1 drugs) [0]
018: (                  fill dep drugs b3 a1) [0]
019: (   charge dep b3 pl1veic2 veic2 a1 drugs) [0]
020: (           moveveichle a1 dep loc4 veic1) [0]
021: (   givecontent loc4 drugs b2 a1 p5 veic1) [0]
022: (           moveveichle a1 loc4 dep veic1) [0]
023: (discharge dep b2 pl2veic1 veic1 a2 tools) [0]
024: (                  fill dep drugs b2 a1) [0]
025: (   charge dep b2 pl2veic1 veic1 a2 drugs) [0]
026: (           moveveichle a1 dep loc6 veic2) [0]
027: (   givecontent loc6 drugs b3 a1 p7 veic2) [0]
```

```
028: (           moveveichle a1 loc6 dep veic2) [0]
029: (discharge dep b3 pl1veic2 veic2 a1 drugs) [0]
030: (                    fill dep drugs b3 a1) [0]
031: (   charge dep b3 pl1veic2 veic2 a1 drugs) [0]
032: (           moveveichle a1 dep loc3 veic1) [0]
033: (   givecontent loc3 drugs b2 a1 p4 veic1) [0]
034: (           moveveichle a1 loc3 dep veic1) [0]
035: (discharge dep b2 pl2veic1 veic1 a2 tools) [0]
036: (                    fill dep drugs b2 a1) [0]
037: (   charge dep b2 pl2veic1 veic1 a1 drugs) [0]
038: (           moveveichle a1 dep loc2 veic2) [0]
039: (   givecontent loc2 drugs b3 a1 p3 veic2) [0]
040: (           moveveichle a1 loc2 dep veic2) [0]
041: (discharge dep b3 pl2veic2 veic2 a2 tools) [0]
042: (                    fill dep drugs b3 a1) [0]
043: (   charge dep b3 pl2veic2 veic2 a1 drugs) [0]
044: (           moveveichle a1 dep loc1 veic1) [0]
045: (   givecontent loc1 drugs b2 a1 p2 veic1) [0]
046: (           moveveichle a1 loc1 dep veic1) [0]
047: (discharge dep b2 pl2veic1 veic1 a2 tools) [0]
048: (                     fill dep food b2 a1) [0]
049: (    charge dep b2 pl2veic1 veic1 a1 food) [0]
050: (           moveveichle a2 dep loc4 veic1) [0]
051: (    givecontent loc4 food b2 a2 p5 veic1) [0]
052: (           moveveichle a2 loc4 dep veic1) [0]
053: (discharge dep b2 pl2veic1 veic1 a2 tools) [0]
054: (                     fill dep food b2 a1) [0]
055: (    charge dep b2 pl2veic1 veic1 a1 food) [0]
056: (           moveveichle a1 dep loc3 veic1) [0]
057: (    givecontent loc3 food b2 a1 p4 veic1) [0]
058: (           moveveichle a1 loc3 dep veic1) [0]
059: (discharge dep b2 pl2veic1 veic1 a2 tools) [0]
060: (                     fill dep food b2 a1) [0]
061: (    charge dep b2 pl2veic1 veic1 a1 food) [0]
062: (           moveveichle a1 dep loc7 veic1) [0]
063: (    givecontent loc7 food b2 a1 p8 veic1) [0]
064: (           moveveichle a1 loc7 dep veic1) [0]
065: (discharge dep b2 pl2veic1 veic1 a2 tools) [0]
066: (                     fill dep food b2 a1) [0]
067: (    charge dep b2 pl2veic1 veic1 a2 food) [0]
068: (           moveveichle a2 dep loc6 veic1) [0]
069: (    givecontent loc6 food b2 a2 p7 veic1) [0]
070: (           moveveichle a2 loc6 dep veic1) [0]
071: (discharge dep b2 pl1veic1 veic1 a1 drugs) [0]
072: (                     fill dep food b2 a1) [0]
073: (    charge dep b2 pl1veic1 veic1 a1 food) [0]
074: (           moveveichle a1 dep loc5 veic1) [0]
075: (    givecontent loc5 food b2 a1 p6 veic1) [0]
```

```
076: (           moveveichle a1 loc5 dep veic1) [0]
077: (discharge dep b2 pl2veic1 veic1 a2 tools) [0]
078: (                     fill dep food b2 a1) [0]
079: (    charge dep b2 pl2veic1 veic1 a1 food) [0]
080: (           moveveichle a1 dep loc1 veic1) [0]
081: (    givecontent loc1 food b2 a1 p1 veic1) [0]
082: (           moveveichle a1 loc1 dep veic1) [0]
083: (discharge dep b2 pl1veic1 veic1 a1 drugs) [0]
084: (                     fill dep tools b2 a1) [0]
085: (   charge dep b2 pl1veic1 veic1 a1 tools) [0]
086: (           moveveichle a1 dep loc1 veic1) [0]
087: (   givecontent loc1 tools b2 a1 p1 veic1) [0]
088: (           moveveichle a1 loc1 dep veic1) [0]
089: (discharge dep b2 pl1veic1 veic1 a1 drugs) [0]
090: (                     fill dep tools b2 a1) [0]
091: (   charge dep b2 pl1veic1 veic1 a1 tools) [0]
092: (           moveveichle a1 dep loc4 veic1) [0]
093: (   givecontent loc4 tools b2 a1 p5 veic1) [0]
094: (           moveveichle a1 loc4 dep veic1) [0]
095: (discharge dep b2 pl1veic1 veic1 a1 drugs) [0]
096: (                     fill dep tools b2 a1) [0]
097: (   charge dep b2 pl1veic1 veic1 a1 tools) [0]
098: (           moveveichle a1 dep loc5 veic1) [0]
099: (   givecontent loc5 tools b2 a1 p6 veic1) [0]
100: (           moveveichle a1 loc5 dep veic1) [0]
101: (discharge dep b2 pl1veic1 veic1 a1 drugs) [0]
102: (                     fill dep tools b2 a1) [0]
103: (   charge dep b2 pl1veic1 veic1 a1 tools) [0]
104: (           moveveichle a1 dep loc6 veic1) [0]
105: (   givecontent loc6 tools b2 a1 p7 veic1) [0]
106: (           moveveichle a1 loc6 dep veic1) [0]
107: (discharge dep b2 pl1veic1 veic1 a1 drugs) [0]
108: (                     fill dep tools b2 a1) [0]
109: (   charge dep b2 pl1veic1 veic1 a1 tools) [0]
110: (           moveveichle a1 dep loc7 veic1) [0]
111: (   givecontent loc7 tools b2 a1 p8 veic1) [0]
112: (          satisfiedwithone tools food p1) [0]

time spent:       0.04 seconds parsing
                  0.37 seconds encoding
                506.83 seconds searching
                507.24 seconds total time

memory used:      8.59 MBytes for problem representation
                543.25 MBytes for searching
                551.84 MBytes total
```

4. Temporal planning and robotics

4.1 Temporal planning

Il terzo punto della traccia richiede la conversione del dominio affinché sia in grado di generare una sequenza temporale di azioni caratterizzate da una durata.
Un problema di pianificazione temporale consiste nel cercare una sequenza di azioni che non sia solo causalmente eseguibile (come nella pianificazione classica), ma anche programmabile, in conformità con un dato insieme di vincoli sulla durata dell'azione, lungo una linea temporale di lunghezza illimitata.

Un'azione durativa è una formulazione in PDDL di un'azione che richiede un certo tempo per essere completata. La quantità di tempo è esprimibile come valore o come disuguaglianza, ovvero consente sia azioni a durata fissa che a durata variabile. Similmente alle azioni tradizionali, è possibile specificare degli effetti ed è inoltre possibile esprimere una condizione da verificare con la parola “condition” piuttosto che “precondition”. Questo cambiamento semantico è stato introdotto per rappresentare che un'azione durativa può non solo condizionare l'inizio dell'azione, ma può avere condizioni che devono essere vere alla fine o per tutta la durata stessa. Ad esempio, nel caso di un volo può essere importante specificare che la pista di partenza/atterraggio rimanga libera per tutto il volo.
A tal proposito, il costrutto at start seguito da un predicato, specifica che la condizione da questo espressa deve essere valida prima del performarsi dell’azione mentre il costrutto over all che deve essere valida prima del performarsi dell’azione e per tutta la durata della stessa. Similmente, tali costrutti possono essere utilizzati all’interno della specificazione degli effetti di ogni azione

Per implementare le durative actions si è partiti come base dal dominio realizzato nella fase di modellazione, introducendo i pesi temporali attraverso i costrutti chiamati fluents, i quali hanno la funzione di variabili di stato ma il loro valore è un numero anziché vero o falso. In particolare sono stati aggiunti i seguenti fluents:

- weight: modellazione del peso del contenuto
- vehicleWeight: modellazione del peso del mezzo
- boxWeight: modellazione del peso della scatola
- pathCost: modellazione del costo del percorso, creato per poterlo utilizzare ai fini della minimizzazione del costo del percorso

```
(:functions
   (weight ?c - content)
   (vehicleWeight ?v - vehicle)
   (boxWeight ?b - box)
   (pathCost ?a - agent)
 )
```

Inoltre, sottoforma di predicati sono stati introdotti i concetti di posto occupato sopra il veicolo e di modellazione di un agente libero da occupazioni.

```
(:predicates (filled ?c - content ?b - box)
   (empty ?b - box)
   (hasContent ?p - person ?c - content)
   (placeVehicle ?p - place ?v - vehicle)
   (placeAvailable ?p - place)
   (placeOccupied ?p - place)
   (boxloaded ?b - box ?v - vehicle)
   (at ?x - (either agent person box content vehicle) ?l - location)
   (needContent ?c - content ?p - person)
   (agentFree ?a - agent))
```

Alle operazioni di riempimento della scatola (fill), di caricamento della scatola sul veicolo (charge), di scarico della scatola dal veicolo (discharge) e di rilascio del contenuto (giveContent) è stato assegnato un peso della durata pari a uno. Per quanto riguarda l'azione di spostamento dell'agente (moveAgent) la durata è un valore fissato pari a due in quanto specifica il movimento del solo agente indipendentemente da altri mezzi.

Infine, per il movimento dell'agente insieme ad un carrello (moveVehicle) la durata è strettamente dipendente dal contenuto delle scatole nel carrello, ciascun dei quali possiede un peso specifico, per cui è calcolato come il doppio del peso del mezzo.

fill

```
(:durative-action fill
  :parameters(?l - location ?c - content ?b - box ?a - agent)
  :duration(= ?duration 1)
  :condition(and (at start (empty ?b))
                 (over all (at ?c ?l))
                 (over all (at ?b ?l))
                 (over all (at ?a ?l))
                 (at start (agentFree ?a))
  :effect(and
           (at start(not(agentFree ?a)))
           (at end (agentFree ?a))
           (at end (filled ?c ?b))
           (at start (not(empty ?b)))
           (at end (assign (boxWeight ?b) (weight ?c)))
           (at end (increase (pathCost ?a)1))))
```

charge

```
(:durative-action charge
      :parameters(?l - location ?a - agent
```

```
            ?b - box ?p - place ?v - vehicle)
      :duration(= ?duration 1)
      :condition(and
                  (over all (at ?v ?l))
                  (at start (at ?b ?l))
                  (over all (at ?a ?l))
                  (over all (placeVehicle ?p ?v))
                  (at start (placeAvailable ?p))
                  (at start (agentFree ?a)))
      :effect(and
           (at start(not(agentFree ?a)))
           (at end (agentFree ?a))
           (at start (not(at ?b ?l)))
           (at end (boxloaded ?b ?v))
           (at start(not(placeAvailable ?p)))
           (at start (placeOccupied ?p))
           (at end (increase (vehicleWeight ?v) (boxWeight ?b)))
           (at end (increase (pathCost ?a)1))))
```

discharge

```
(:durative-action discharge
      :parameters(?l - location ?a - agent
          ?b - box ?p - place ?v - vehicle)
      :duration(= ?duration 1)
      :condition(and
                  (over all (at ?v ?l))
                  (over all (at ?a ?l))
                  (at start(boxloaded ?b ?v))
                  (over all (placeVehicle ?p ?v))
                  (over all (placeOccupied ?p))
                  (over all (empty ?b))
                  (at start (agentFree ?a)))
      :effect(and
           (at start(not(agentFree ?a)))
           (at end (agentFree ?a))
           (at end (at ?b ?l))
           (at start (not(boxloaded ?b ?v)))
           (at end(not(placeOccupied ?p)))
           (at end (placeAvailable ?p))

           (at end (decrease(vehicleWeight ?v) (boxWeight ?b)))
           (at end (decrease (pathCost ?a)1))))
```

move_vehicle

```
(:durative-action move_vehicle
```

```
        :parameters(?a - agent ?from ?to - location ?v - vehicle)
        :duration(= ?duration ( *( vehicleWeight ?v ) 2))
        :condition(and
                    (at start (at ?a ?from))
                    (at start (at ?v ?from))
                    (at start(agentFree ?a)))
        :effect(and
                (at start (not(agentFree ?a)))
                (at end(agentFree ?a))
                (at start (not(at ?a ?from)))
                (at end (at ?a ?to))
                (at start (not(at ?v ?from)))
                (at end (at ?v ?to))
                (at end (increase (pathCost ?a)(*( vehicleWeight ?v ) 2)))))
```

move_agent

```
 (:durative-action move_agent
        :parameters(?a - agent ?from ?to - location )
        :duration(= ?duration 2)
        :condition(and
                    (at start (at ?a ?from))
                    (at start(agentFree ?a))
        )
        :effect(and
                (at start (not(agentFree ?a)))
                (at end(agentFree ?a))
                (at start (not(at ?a ?from)))
                (at end (at ?a ?to))
                (at end (increase (pathCost ?a) 2))))
```

give_content

```
(:durative-action give_content
        :parameters(?l - location ?c - content
                    ?b - box ?a - agent ?p - person ?v - vehicle)
        :duration(= ?duration 1)
        :condition(and
                    (at start(filled ?c ?b))
                    (over all (at ?v ?l))
                    (over all (at ?p ?l))
                    (over all (at ?a ?l))
                    (over all (boxLoaded ?b ?v))
                    (over all (needContent ?c ?p))
                    (at start (agentFree ?a))
        )
```

```
:effect(and
            (at start (not(agentFree ?a)))
            (at end (agentFree ?a))
            (at start (not(filled ?c ?b)))
            (at end(empty ?b))
            (at end (hasContent ?p ?c))

            (at end (assign (boxWeight ?b)0))
            (at end (increase (pathCost ?a)1))))
```

Si è dunque convertito il primo problema affrontato nel quesito due introducendo dei pesi a ciascuno dei contenuti: peso pari a 1 alle medicine, peso pari a 2 al cibo e peso pari a 3 agli strumenti. Inoltre, per modellare l'assenza di contenuto all'interno delle scatole è stato impostato loro un peso pari a 0.

Inoltre, rispetto alla prima formulazione, si è inizializzato il fattore che l'agente richiesto fosse inizialmente libero da occupazioni.

```
(define (problem emergencyDelivery)
    ( :domain durativeEmergency )
    ( :objects
        dep loc1 loc2 - location
        b1 b2 b3 b4 b5 - box
        p1 p2 p3 - person
        a - agent
        veic - vehicle
        pl1 pl2 pl3 pl4 - place
        drugs food tools – content)

    ( :init
        (= (weight drugs) 1)
        (= (weight food) 2)
        (= (weight tools) 3)

        (= (boxweight b1) 0)
        (= (boxweight b2) 0)
        (= (boxweight b3) 0)
        (= (boxweight b4) 0)
        (= (boxweight b5) 0)

        (= (vehicleWeight veic) 1)
        (= (pathCost a) 0)

        (agentFree a)


        ( at a dep )
        (at veic dep)
```

```
        ( at b1 dep )
        ( at b2 dep )
        ( at b3 dep )
        ( at b4 dep )
        ( at b5 dep )

        ( at food dep )
        ( at tools dep )
        ( at drugs dep )

        ( at p1 loc1 )
        ( at p2 loc1)
        ( at p3 loc2 )

        ( needContent food p1)
        ( needContent drugs p1)
        ( needContent drugs p2)
        ( needContent food p3)

        (empty b1)
        (empty b2)
        (empty b3)
        (empty b4)
        (empty b5)

        ( placeAvailable pl1 )
        ( placeAvailable pl2 )
        ( placeAvailable pl3 )
        ( placeAvailable pl4 )
        ( placeVehicle pl1 veic )
        ( placeVehicle pl2 veic )
        ( placeVehicle pl3 veic )
        ( placeVehicle pl4 veic )
    )
    ( :goal
        (and
            ( hasContent p1 food )
            ( hasContent p1 drugs )
            ( hasContent p2 drugs )
            ( hasContent p3 food )
        )

    )

    (:metric minimize (pathCost a))

)
```

È stata inoltre impostata la minimizzazione del costo del percorso del singolo agente, ovvero la minimizzazione della durata delle azioni fino all'ottenimento del goal.

Una volta creata l'istanza del problema, si è resa necessaria l'individuazione di un planner per effettuare la risoluzione dello stesso. La libreria planutils fornisce una serie di planner, tra cui POPF, TFD, LPG-TD, che permettono la risoluzione dell'istanza del problema in quanto supportano le durative actions.

I risultati ottenuti con i planner POPF e TFD sono molto simili, poiché essi in breve tempo riescono ad generare un plan, che però non risulta ottimale nell'ottimizzazione della metrica scelta; pertanto la scelta del planner è ricaduta su LPG-TD, in quanto riesce a fornire una soluzione ottimale in un tempo relativamente breve sia utilizzando la modalità "-speed" (che permette di ottenere un plan nel minor tempo possibile), che utilizzando la modalità "-quality" (che consente di ottenere un plan migliore, a discapito del tempo di ricerca).

LPG-td è un'estensione di LPG per gestire le nuove funzionalità del linguaggio PDDL2.2. L'algoritmo LPG, acronimo di "Linear Planning Graphs," è stato sviluppato per affrontare problemi di pianificazione in cui le azioni possono avere effetti continui e non solo discreti. Questo significa che invece di considerare solo azioni che cambiano lo stato di un sistema da uno stato all'altro in modo discreto, l'algoritmo LPG può gestire azioni che hanno effetti più complessi e gradualmente variabili.

A tutti gli effetti i plan ottenuti attraverso LPG-TD risultano essere meglio ottimizzati, minimizzando le perdite di tempo dovute allo spostamento del veicolo.

Di seguito è riportato il plan ottenuto con la modalità "speed":

```
; Version LPG-td-1.4
; Seed 89465689
; Command line: ./lpg-td -o emergency_durative_actions.pddl -f
firstProblemDurative.pddl -n 1
; Problem firstProblemDurative.pddl
; Time 0.01
; Search time 0.01
; Parsing time 0.00
; Mutex time 0.00
; MetricValue 44.00

0.0002:   (FILL DEP DRUGS B5 A) [1.0000]
1.0005:   (CHARGE DEP A B5 PL4 VEIC) [1.0000]
2.0008:   (FILL DEP FOOD B2 A) [1.0000]
3.0010:   (CHARGE DEP A B2 PL3 VEIC) [1.0000]
4.0012:   (FILL DEP DRUGS B4 A) [1.0000]
5.0015:   (CHARGE DEP A B4 PL1 VEIC) [1.0000]
```

```
6.0017:   (MOVE_VEHICLE A DEP LOC1 VEIC) [10.0000]
16.0020:   (GIVE_CONTENT LOC1 DRUGS B5 A P1 VEIC) [1.0000]
17.0022:   (GIVE_CONTENT LOC1 DRUGS B4 A P2 VEIC) [1.0000]
18.0025:   (MOVE_VEHICLE A LOC1 DEP VEIC) [6.0000]
24.0027:   (DISCHARGE DEP A B5 PL4 VEIC) [1.0000]
25.0030:   (FILL DEP FOOD B5 A) [1.0000]
26.0032:   (FILL DEP FOOD B3 A) [1.0000]
27.0035:   (CHARGE DEP A B3 PL2 VEIC) [1.0000]
28.0037:   (MOVE_VEHICLE A DEP LOC2 VEIC) [10.0000]
38.0040:   (GIVE_CONTENT LOC2 FOOD B2 A P3 VEIC) [1.0000]
39.0043:   (MOVE_VEHICLE A LOC2 LOC1 VEIC) [6.0000]
45.0045:   (GIVE_CONTENT LOC1 FOOD B3 A P1 VEIC) [1.0000]
```

2.2 Robotics planning

La pianificazione robotica si occupa della progettazione e dell'implementazione di algoritmi e strategie che consentono a robot autonomi di pianificare e programmare le loro azioni per raggiungere obiettivi specifici in ambienti dinamici e spesso incerti modellanti la realtà.
Richiamando alcuni concetti della pianificazione classica, la pianificazione robotica parte dall'identificazione degli obiettivi che il robot deve raggiungere. Per pianificare le azioni, il robot deve avere una rappresentazione dell'ambiente in cui opera, nel quale si muove attraverso l'individuazione di un plan che porta al raggiungimento dello scopo prefissato.

Per la risoluzione del terzo task previsto, è richiesta l'implementazione del problema nel punto precedente in ROS2 Planning System (Plansys2 in breve), una piattaforma di pianificazione e gestione di sistemi robotici open source basata su ROS (Robot Operating System 2) che offre strumenti per la pianificazione, l'esecuzione e la gestione delle attività per i robot autonomi ed è in grado di supportare la pianificazione temporale.

Nel contesto di Plansys2, le fake actions sono azioni simulate o rappresentate in modo da consentire al sistema di pianificazione di gestire situazioni specifiche o di modellare comportamenti desiderati che non corrispondono necessariamente a comportamenti reali del robot o dell'agente. Infatti, servono a rappresentare particolari situazioni come, ad esempio, il fatto che il robot "rimanga in attesa" in un determinato stato o posizione per un certo periodo di tempo, e possono essere impiegate per gestire le tempistiche e il tempo modellando pause, ritardi o attese artificiali.

Le fake action utilizzate sono state definite in linguaggio C++ e ricalcano le azioni durative definite nel dominio quali:

- fill: modella il riempimento della scatola con un contenuto
- charge: modella il carico della scatola sul veicolo
- discharge: modella lo scarico della scatola dal veicolo
- giveContent: modella l'azione di consegna della scatola alla persona
- moveAgent: modella il movimento dell'agente
- moveVehicle: modella il movimento del mezzo

Nella definizione delle fake actions, è specificato il comando ActionExecutorClient, il quale invoca il nome del node esecutore dell'azione e definisce l'intervallo temporale di invocazione della funzione do_work(), la quale rappresenta il compito svolto dall'azione, aumenta il contatore che misura il progresso e invia feedback riguardo l'esecuzione.

charge.cpp

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

#include "plansys2_executor/ActionExecutorClient.hpp"

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

using namespace std::chrono_literals;

class Charge : public plansys2::ActionExecutorClient
{
public:
  Charge()
  : plansys2::ActionExecutorClient("charge", 250ms)
  {
    progress_ = 0.0;
  }

private:
  void do_work()
  { std :: vector <std :: string > arguments = get_arguments () ;
    if (progress_ < 1.0) {
      progress_ += 0.2;
      send_feedback(progress_, "Agent "+arguments [1]+ " is loading "+
            arguments [2]+ " on vehicle "+ arguments [4] + " at "+
            arguments [0] + " using place "+arguments [3]);
    } else {
      finish(true, 1.0, "Agent "+arguments [1]+ " loaded  "+
                arguments [2]+ " on vehicle "+arguments [4]+ " at "+
                arguments [0] + " using place "+arguments [3]);

      progress_ = 0.0;
      std::cout << std::endl;
    }

    std::cout << "\r\e[K" << std::flush;
    std::cout << "Agent "+arguments [1]+ " is loading "+arguments
```

```
            [2]+ " on vehicle "+arguments [4]+ " at "+arguments [0] +
            " using place "+arguments [3] +" . . . [ " << std::min(100.0, progress_
* 100.0) << "% ]  " <<
            std::flush;
  }

  float progress_;
};

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

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

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

  rclcpp::shutdown();

  return 0;
}
```

discharge.cpp

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

#include "plansys2_executor/ActionExecutorClient.hpp"

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

using namespace std::chrono_literals;

class Discharge : public plansys2::ActionExecutorClient
{
public:
  Discharge()
  : plansys2::ActionExecutorClient("discharge", 250ms)
  {
    progress_ = 0.0;
  }
```

```
private:
  void do_work()
  { std :: vector <std :: string > arguments = get_arguments () ;
    if (progress_ < 1.0) {
      progress_ += 0.2;
      send_feedback(progress_, "Agent "+arguments [1]+ " is unloading "+
            arguments [2]+ " on vehicle "+arguments [4]+ " at "+
            arguments [0] + " using place "+arguments [3] + " at loc "+ arguments
[0]);
    } else {
      finish(true, 1.0, "Agent "+arguments [1]+ " unloaded "+
                arguments [2]+ " on vehicle "+arguments [4]+ " at "+
                arguments [0] + " using place "+arguments [3]);

      progress_ = 0.0;
      std::cout << std::endl;
    }

    std::cout << "\r\e[K" << std::flush;
    std::cout << "Agent "+arguments [1]+ " is unloading "+arguments
            [2]+ " on vehicle "+arguments [4]+ " at "+arguments [0] + " using place
"+
            arguments [3] +" . . . [ " << std :: min( 100.0 ,
            progress_ * 100.0) << "% ] " <<
            std::flush;
  }

  float progress_;
};

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

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

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

  rclcpp::shutdown();

  return 0;
}
```

fill.cpp

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

#include "plansys2_executor/ActionExecutorClient.hpp"

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

using namespace std::chrono_literals;

class Fill : public plansys2::ActionExecutorClient
{
public:
  Fill()
  : plansys2::ActionExecutorClient("fill", 250ms)
  {
    progress_ = 0.0;
  }

private:
  void do_work()
  { std :: vector <std :: string > arguments = get_arguments () ;
    if (progress_ < 1.0) {
      progress_ += 0.2;
      send_feedback(progress_, "Agent "+arguments [3]+ " is filling box"+
            arguments [2]+ " with "+ arguments [1] + " at "+
            arguments [0]);
    } else {
      finish(true, 1.0, "Agent "+arguments [3]+ " filled box"+
            arguments [2]+ " with "+ arguments [1] + " at "+
            arguments [0]);

      progress_ = 0.0;
      std::cout << std::endl;
    }

    std::cout << "\r\e[K" << std::flush;
    std::cout << "Agent "+arguments [3]+ " is filling "+
            arguments [2]+ " with "+ arguments [1] + " at "+
            arguments [0] +" . . . [ " << std :: min( 100.0 ,
            progress_ * 100.0) << "% ] " <<
            std::flush;
  }

  float progress_;
};
```

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

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

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

  rclcpp::shutdown();

  return 0;
}
```

give_content.cpp

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

#include "plansys2_executor/ActionExecutorClient.hpp"

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

using namespace std::chrono_literals;

class GiveContent : public plansys2::ActionExecutorClient
{
public:
  GiveContent()
  : plansys2::ActionExecutorClient("give_content", 250ms)
  {
    progress_ = 0.0;
  }

private:
  void do_work()
  { std :: vector <std :: string > arguments = get_arguments () ;
    if (progress_ < 1.0) {
      progress_ += 0.2;
      send_feedback(progress_, "Agent "+arguments [3]+ " giving "+
            arguments [1]+ " from "+ arguments [2] + " to "+
            arguments [4] + " at "+arguments [0]);
```

```
    } else {
      finish(true, 1.0 , "Agent "+arguments [3]+ " gave "+
            arguments [1]+ " from "+ arguments [2] + " to "+
            arguments [4] + " at "+arguments [0]);

      progress_ = 0.0;
      std::cout << std::endl;
    }

    std::cout << "\r\e[K" << std::flush;
    std::cout << "Agent " +arguments [3]+ " giving "+
            arguments [1]+ " from "+ arguments [2] + " to "+
            arguments [4] + " at "+arguments [0] +" . . . [ " << std :: min( 100.0
,
            progress_ * 100.0) << "% ] " <<
            std::flush;
  }

  float progress_;
};

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

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

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

  rclcpp::shutdown();

  return 0;
}
```

move_agent.cpp

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

#include "plansys2_executor/ActionExecutorClient.hpp"

#include "rclcpp/rclcpp.hpp"
#include "rclcpp_action/rclcpp_action.hpp"
```

```
using namespace std::chrono_literals;

class MoveAgent : public plansys2::ActionExecutorClient
{
public:
  MoveAgent()
  : plansys2::ActionExecutorClient("move_agent", 250ms)
  {
    progress_ = 0.0;
  }

private:
  void do_work()
  { std :: vector <std :: string > arguments = get_arguments () ;
    if (progress_ < 1.0) {
      progress_ += 0.2;
      send_feedback(progress_, "Agent "+arguments [0]+
            " from "+ arguments [1] + " to "+
            arguments [1]);
    } else {
      finish(true, 1.0, "Agent "+arguments [0]+
            " from "+ arguments [1] + " to "+
            arguments [1]);

      progress_ = 0.0;
      std::cout << std::endl;
    }

    std::cout << "\r\e[K" << std::flush;
    std::cout << "Agent "+arguments [0]+
            " from "+ arguments [1] + " to "+
            arguments [1] +" . . . [ " << std::min(100.0, progress_ * 100.0) << "%
]  " <<
            std::flush;
  }

  float progress_;
};

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

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

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

  rclcpp::shutdown();

  return 0;
}
```

move_vehicle.cpp

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

#include "plansys2_executor/ActionExecutorClient.hpp"

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

using namespace std::chrono_literals;

class MoveVehicle : public plansys2::ActionExecutorClient
{
public:
  MoveVehicle()
  : plansys2::ActionExecutorClient("move_vehicle", 250ms)
  {
    progress_ = 0.0;
  }

private:
  void do_work()
  { std :: vector <std :: string > arguments = get_arguments () ;
    if (progress_ < 1.0) {
      progress_ += 0.2;
      send_feedback(progress_, "Agent "+arguments [0]+ " moving on "+
            arguments [2]+ " from "+ arguments [1] + " to "+
            arguments [1]);
    } else {
      finish(true, 1.0, "Agent "+arguments [0]+ " moved "+
            arguments [2]+ " from "+ arguments [1] + " to "+
            arguments [1]);

      progress_ = 0.0;
      std::cout << std::endl;
    }
```

```
    std::cout << "\r\e[K" << std::flush;
    std::cout << "Agent "+arguments [0]+ " moving on "+
            arguments [2]+ " from "+ arguments [1] + " to "+
            arguments [1] +" . . . [ " << std::min(100.0, progress_ * 100.0) << "%
]  " <<
            std::flush;
  }

  float progress_;
};

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

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

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

  rclcpp::shutdown();

  return 0;
}
```

Definite le fake actions, per associarle alle azioni a cui fanno riferimento, si è personalizzato il file CMakeList.txt presente, il quale è componente chiave nel processo di compilazione e gestione dei progetti software in CMake, un sistema di build open source in grado di definire dipendenze e target di compilazione, il file di launch plansys2_project_launch.py realizzato in Python, anch'esso presente nel workspace, il quale gestisce le operazioni di lancio e si è realizzato il file commands per inizializzare le condizioni del problema.

CMakeLists.txt

```
cmake_minimum_required(VERSION 3.5)
project(plansys2_project)

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(move_agent_action_node src/move_agent_action_node.cpp)
ament_target_dependencies(move_agent_action_node ${dependencies})

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

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

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

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

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

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

install(TARGETS
  move_agent_action_node
  charge_action_node
  discharge_action_node
  move_vehicle_action_node
  fill_action_node
  give_content_action_node
  ARCHIVE DESTINATION lib
  LIBRARY DESTINATION lib
  RUNTIME DESTINATION lib/${PROJECT_NAME}
)

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

  find_package(ament_cmake_gtest REQUIRED)
endif()

ament_export_dependencies(${dependencies})

ament_package()
```

plansys2_project_launch.py

```
import os
from ament_index_python.packages import get_package_share_directory
from typing import List,Dict
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('plansys2_project')
    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/emergency_durative_actions.pddl',
          'namespace': namespace
          }.items())
    # Specify the actions

    move_agent_cmd = Node(
        package='plansys2_project',
        executable='move_agent_action_node',
        name='move_agent_action_node',
        namespace=namespace,
        output='screen',
        parameters=[])
    charge_cmd = Node(
        package='plansys2_project',
        executable='charge_action_node',
        name='charge_action_node',
        namespace=namespace,
        output='screen',
        parameters=[])

    discharge_cmd = Node(
        package='plansys2_project',
        executable='discharge_action_node',
        name='discharge_action_node',
```

```
        namespace=namespace,
        output='screen',
        parameters=[])
    move_vehicle_cmd = Node(
        package='plansys2_project',
        executable='move_vehicle_action_node',
        name='move_vehicle_action_node',
        namespace=namespace,
        output='screen',
        parameters=[])

    give_content_cmd = Node(
        package='plansys2_project',
        executable='give_content_action_node',
        name='give_content_action_node',
        namespace=namespace,
        output='screen',
        parameters=[])

    fill_cmd = Node(
        package='plansys2_project',
        executable='fill_action_node',
        name='fill_action_node',
        namespace=namespace,
        output='screen',
        parameters=[])

    # Create the launch description and populate
    ld = LaunchDescription()
    ld.add_action(declare_namespace_cmd)
    # Declare the launch options
    ld.add_action(plansys2_cmd)
    ld.add_action(move_agent_cmd)
    ld.add_action(charge_cmd)
    ld.add_action(discharge_cmd)
    ld.add_action(move_vehicle_cmd)
    ld.add_action(give_content_cmd)
    ld.add_action(fill_cmd)

    return ld
```

commands

```
        set instance DEP location
        set instance LOC1 location
        set instance LOC2 location

        set instance B1 box
```

```
set instance B2 box
set instance B3 box
set instance B4 box
set instance B5 box

set instance P1 person
set instance P2 person
set instance P3 person

set instance A agent
set instance VEIC vehicle

set instance PL1 place
set instance PL2 place
set instance PL3 place
set instance PL4 place

set instance DRUGS content
set instance FOOD content
set instance TOOLS content

set function (= (weight DRUGS) 1)
set function (= (weight FOOD) 2)
set function (= (weight TOOLS) 3)

set function (= ( boxweight B1) 0)
set function (= ( boxweight B2) 0)
set function (= ( boxweight B3) 0)
set function (= ( boxweight B4) 0)
set function (= ( boxweight B5) 0)

set function (= ( vehicleweight VEIC) 1)
set function (= ( pathcost A) 0)

set predicate (at B1 DEP)
set predicate (at B2 DEP)
set predicate (at B3 DEP)
set predicate (at B4 DEP)
set predicate (at B5 DEP)

set predicate (at A DEP)
set predicate (at VEIC DEP)

set predicate (agentfree A)

set predicate (at FOOD DEP)
set predicate (at DRUGS DEP)
set predicate (at TOOLS DEP)
```

```
set predicate (at P1 LOC1)
set predicate (at P2 LOC1)
set predicate (at P3 LOC2)

set predicate (needcontent FOOD P1)
set predicate (needcontent DRUGS P1)
set predicate (needcontent DRUGS P2)
set predicate (needcontent FOOD P3)

set predicate (empty B1)
set predicate (empty B2)
set predicate (empty B3)
set predicate (empty B4)
set predicate (empty B5)

set predicate (placeavailable PL1)
set predicate (placeavailable PL2)
set predicate (placeavailable PL3)
set predicate (placeavailable PL4)

set predicate (placevehicle PL1 VEIC)
set predicate (placevehicle PL2 VEIC)
set predicate (placevehicle PL3 VEIC)
set predicate (placevehicle PL4 VEIC)
```

Per compiere l'esecuzione del plan attraverso PlanSys2, dopo aver propriamente organizzato i file descritti precedentemente, si procede alla creazione dell'ambiente di esecuzione attraverso il comando *colcon build --symlink-install*, il quale procede ad effettuare la build del progetto.
Successivamente, per effettuare il lancio di Plansys2 si ricorre alle seguenti istruzioni:

- *source install/setup.bash*
- *ros2 launch plansys2_project plansys2_project_launch.py*

Una volta effettuate queste operazioni, Plansys2 viene attivato e rimane in attesa di un problema da risolvere.
Si apre dunque un nuovo terminale in cui poter eseguire il comando *ros2 run plansys2_terminal plansys2_terminal* affinché si possa interagire con il terminale di Plansys2.

Si procede alla definizione del problema attraverso di essa, utilizzando il supporto del file commands visto prima dove sono inserite le istruzioni descritte nel problema definito nel temporal planning al passo 4.1.
Per fare ciò si usa il comando *source /home/aiguy/plansys2_ws/src/ros2_planning_system_examples/plansys2_project/launch/commands*

Fatto ciò, è sufficiente eseguire il plan generato precedentemente nel punto 4.1 all'interno del terminale di plansys2, attraverso l'istruzione *run plan-file /home/aiguy/plansys2_ws/src/ros2_planning_system_examples/plansys2_project/pddl/durativePlan.plan*

Si riporta dunque quanto risultante nei due terminali. In conclusione, si può notare che i tempi di completamento delle azioni sono pari a quelli specificati all'interno delle singole fake action e il plan viene eseguito con successo.