PROGETTO INTELLIGENZA ARTIFICIALE

Giulia Perri 242645 , Antonio Paonessa 252318,
Desirè Chiappetta 257192

1 File di dominio 2
1.1 Requisiti 2
1.2 Tipi 2
1.3 Predicati 2
1.4 Azioni 3
2 Classical Planning 7
2.1 Istanza 1 8
2.2 Istanza 2 10
2.3 Planner 12
2.4 A* Modificato 15
2.5 Euristica 17
2.6 Istanza 1 21
2.7 Istanza 2 23
3 Temporal Planning & Robotics 26
3.1 Temporal Planning 26
3.2 Robotics Planning 30

1 File di dominio

Nel file di dominio abbiamo delineato gli elementi universali del problema in esame, ossia tutti quei fattori che restano costanti a prescindere dal contesto particolare che intendiamo affrontare. In particolare, abbiamo definito: requisiti, tipi, funzioni, predicati e azioni.

1.1 Requisiti

I **requisiti**, che indicano che un dato planner deve supportare un determinato aspetto del linguaggio, abbiamo usato i seguenti requisiti:

- Strips, consente l'uso di effetti di aggiunta e cancellazione di base;
- Typing, consente l'uso della tipizzazione per gli oggetti, simile alle classi e sotto-classi nella programmazione orientata agli oggetti;

```
(:requirements :strips :typing)
```

1.2 Tipi

I **tipi**, consentono di creare tipi di base e sottotipi ai quali possiamo applicare predicati.

- Location: Rappresenta un luogo generico.
- Box: Rappresenta una scatola.
- Agent: Rappresenta un agente, ad esempio un robot.
- Workstation: Rappresenta una postazione di lavoro.
- Content: Rappresenta un contenuto generico e astratto.
- Content-type è utilizzato per specificare i vari tipi di contenuto che possono esistere nel dominio.
- Carrier: Rappresenta un mezzo di trasporto.
- Place: Rappresenta una locazione del Carrier.

```
(:types
   location box content agent workstation content-type carrier place - object )
```

1.3 Predicati

I predicati rappresentano informazioni di stato e relazioni tra gli oggetti nel dominio di pianificazione.

- `(connected ?loc1 ?loc2 - location)`: Indica che due posizioni (`?loc1` e `?loc2`) sono collegate tra loro.
- ` (agent-at-loc ?agent - agent ?loc - location) `: Indica che un agente (`?agent`) si trova in un determinato luogo(`?loc`).
- ` (content-at-loc ?content - content ?loc - location) `: Indica che un contenuto (`?content`) si trova in una determinata posizione (`?loc`).
- ` (box-at-loc ?box - box ?loc - location) `: Indica che una scatola (`?box`) si trova in un determinato luogo (`?loc`).

- ` (ws-at-loc ?ws - workstation ?loc - location) ` : Indica che una postazione di lavoro (`?ws`) si trova in un determinato luogo (`?loc`).
- `(empty-box  ?box - box)` : Indica l'assenza di contenuto in una scatola (`?box`).
- `(filled-box  ?box - box ?content - content)` : Indica che una scatola (`?box`) è riempita con un determinato contenuto (`?content`).
- `(box-at-ws ?box - box  ?ws - workstation)` : Indica che una scatola (`?box`) si trova presso una postazione di lavoro (`?ws`).
- `(box-at-place ?box - box ?place - place)` : Indica che una scatola (`?box`) si trova presso una locazione (`?place`).
- `(is-type ?content - content ?t - content-type)` : Questo predicato indica che un determinato contenuto (`?content`) è di un tipo specifico (`?t`), rappresentando una relazione di tipo tra il contenuto e il suo tipo.
- `(content-type-at-workstation ?ws - workstation ?t - content-type)` : Indica che un contenuto di un certo tipo (`?t`) si trova presso una postazione di lavoro (`?ws`).
- `(empty-place  ?place - place)` : Indica l'assenza di scatola in una locazione(`?place`).
- `(place-at-carrier  ?place- place ?carrier - carrier)` : Indica che la locazione (`?place`) è relativa ad un determinato mezzo di trasporto (`? carrier `).
- `(carrier-at-agent ?carrier - carrier ?agent - agent)` : Indica che il mezzo di trasporto (`?carrier`) è relativa ad un agente (`? agent`).

```
(:predicates
  (connected ?loc1 ?loc2 - location)
  (agent-at-loc ?agent - agent ?loc - location)
  (content-at-loc ?content - content ?loc - location)
  (box-at-loc ?box - box ?loc - location)
  (ws-at-loc ?ws - workstation ?loc - location)
  (empty-box ?box - box)
  (filled-box ?box - box ?content - content)
  (box-at-ws ?box - box ?ws - workstation)
  (box-at-place ?box - box ?place - place)
  (is-type ?content - content ?t - content-type)
  (content-type-at-ws ?t - content-type ?ws - workstation)
  (empty-place ?place - place)
  (place-at-carrier ?place - place ?carrier - carrier)
  (carrier-at-agent ?carrier - carrier ?agent - agent)
)
```

1.4 Azioni

Un'azione, generalmente, definisce una trasformazione dello stato del mondo.

Pick-up:

Si procede a definire la action relativa al ritiro di una box da una location. I parametri utili alla action in oggetto sono: la location, la scatola da ritirare, l'agente robotico, il carrier e lo slot del carrier in cui mettere la box.

Le precondizioni che consentono l'esecuzione prevedono che:

- l'agente robotico e la box si trovino nella stessa locazione;
- Il place sia relativo al carrier, a sua volta relativo all'agente;
- Il place sia vuoto.

Gli effetti dell'esecuzione portano ad uno stato in cui:

- Il place non è più vuoto e contiene la box inserita come parametro;
- la box non risulta più alla location.

```
(:action pick-up
  :parameters (
    ?loc - location
    ?box - box
    ?agent - agent
    ?carrier - carrier
    ?place - place
  )
  :precondition (and
    (agent-at-loc ?agent ?loc)
    (box-at-loc ?box ?loc)
    (carrier-at-agent ?carrier ?agent)
    (place-at-carrier ?place ?carrier)
    (empty-place ?place)
  )
  :effect (and
    (box-at-place ?box ?place)
    (not (empty-place ?place))
    (not (box-at-loc ?box ?loc))
  )
)
```

Move:

Si procede a definire la action relativa allo spostamento di un agente da una location all'altra. I parametri utili alla action in oggetto sono: le location (from e to) e l'agente robotico coinvolto.

Le precondizioni che consentono l'esecuzione prevedono che:

- l'agente robotico si trovi nella location di partenza;
- le locations siano connesse.

Gli effetti dell'esecuzione portano ad uno stato in cui:

- l'agente robotico non si trovi nella location di partenza;
- l'agente robotico si trovi nella location di arrivo.

```
(:action move
        :parameters (?from ?to - location ?agent - agent)
        :precondition (and
            (at-agent ?agent ?from)
            (connected ?from ?to)
        )
```

```
        :effect (and
            (not (at-agent ?agent ?from))
            (at-agent ?agent ?to)
        )
    )
```

Deliver:

Si procede a definire la action relativa alla consegna di una box ad una workstation. I parametri utili alla action in oggetto sono: la box da consegnare, la workstation a cui consegnare la box, la location in cui si trova la workstation, l'agente robotico coinvolto, il carrier e il posto sul carrier.

Le precondizioni che consentono l'esecuzione prevedono che:

- l'agente robotico si trovi nella stessa location della workstation a cui consegnare la box
- l'agente robotico stia portando la box da consegnare

Gli effetti dell'esecuzione portano ad uno stato in cui:

- l'agente robotico non porta più la box;
- la box si trova nella workstation a cui è stata consegnata.

```
(:action deliver
  :parameters (
    ?box - box
    ?ws - workstation
    ?location - location
    ?agent - agent
    ?carrier - carrier
    ?place - place
  )
  :precondition (and
    (ws-at-loc ?ws ?location)
    (agent-at-loc ?agent ?location)
    (carrier-at-agent ?carrier ?agent)
    (place-at-carrier ?place ?carrier)
    (box-at-place ?box ?place)
  )
  :effect (and
    (not (box-at-place ?box ?place))
    (empty-place ?place)
    (box-at-ws ?box ?ws)
  )
)
```

Fill:

Si procede a definire la action relativa al riempimento di una box con un contenuto. I parametri utili alla action in oggetto sono: la box da riempire, il contenuto della box, la location in cui si trova il contenuto, l'agente robotico coinvolto, il carrier e il place. Le precondizioni che consentono l'esecuzione prevedono che:

- la box sia vuota;
- l'agente robotico stia portando la box da riempire;
- il contenuto e l'agente robotico si trovino nella stessa location.

Gli effetti dell'esecuzione portano ad uno stato in cui:

- la box non è più vuota;
- la box è riempita con il contenuto;
- il contenuto non si trova più nella location.

```
(:action fill
  :parameters (
    ?box - box
    ?content - content
    ?loc - location
    ?agent - agent
    ?carrier - carrier
    ?place - place
  )
  :precondition (and
    (content-at-loc ?content ?loc)
    (agent-at-loc ?agent ?loc)
    (carrier-at-agent ?carrier ?agent)
    (place-at-carrier ?place ?carrier)
    (box-at-place ?box ?place)
    (empty-box ?box)
  )
  :effect (and
    (not (empty-box ?box))
    (filled-box ?box ?content)
    (not (content-at-loc ?content ?loc))
  )
)
```

Empty:

Si procede a definire la action relativa allo svuotamento di una box. I parametri utili alla action in oggetto sono: la box da svuotare, il contenuto della box, la workstation che riceve il contenuto, l'agente robotico coinvolto e la location. Le precondizioni che consentono l'esecuzione prevedono che:

- la box sia riempita con il contenuto da consegnare;
- la box si trovi nella workstation che deve ricevere il contenuto;
- l'agente e la workstation si trovino nella stessa locazione.

Gli effetti dell'esecuzione portano ad uno stato in cui:

- la box non ha più contenuto;
- la box viene rilasciata nella locazione;
- la workstation ha il contenuto della box.

```
(:action empty
  :parameters (
    ?box - box
    ?content - content
    ?t - content-type
    ?ws - workstation
    ?agent - agent
    ?loc - location
  )
  :precondition (and
    (is-type ?content ?t)
    (box-at-ws ?box ?ws)
    (ws-at-loc ?ws ?loc)
    (filled-box ?box ?content)
    (agent-at-loc ?agent ?loc)
  )
  :effect (and
    (not (filled-box ?box ?content))
    (empty-box ?box)
    (box-at-loc ?box ?loc)
    (not (box-at-ws ?box ?ws))
    (content-type-at-ws ?t ?ws)
  )
)
```

2 Classical Planning

Sono state realizzate due istanze del problema, una che rispetta il dominio realizzato al punto 1 ed una che si basa, invece, su alcune estensioni; è stato definito, inoltre, un planner che sfrutta l'algoritmo A* ed un'euristica specifica per il dominio in esame.

2.1 Istanza 1

Nella prima istanza tutte le box e tutti i content sono collocati inizialmente in una singola location, la *central_warehouse*, non dotata di workstation; un singolo agente robotico, anch'esso collocato inizialmente alla *central_warehouse*, si occupa di consegnare le box alle workstation. Il goal richiede che alcune workstation ricevano un content, altre ne ricevano più di uno, altre ancora non ricevano nulla.
È stato scelto di definire un problema con tre location aggiuntive, oltre alla *central_warehouse*, quattro workstation e tre diversi tipi di content. Il goal prevede che la ws1 riceva un solo content, la ws2 e la ws3 ne ricevano diversi mentre la ws4 non ne riceva nessuno.

```
(define (problem instance1) (:domain industrial_manufacturing)
  (:objects
    ; define the world
    central_warehouse loc1 loc2 - location
    ws1 ws2 ws3 ws4 ws5 - workstation
    b1 b2 - box
    a1 - agent
    c1 - carrier
    p1 - place
    valve1 valve2 bolt1 bolt2 hammer1 hammer2 - content
    valve bolt hammer - content-type
  )
  (:init
    ; define the type of content
    (is-type valve1 valve)
    (is-type valve2 valve)
    (is-type bolt1 bolt)
    (is-type bolt2 bolt)
    (is-type hammer1 hammer)
    (is-type hammer2 hammer)

    ; define the empty box located at central_warehouse
    (box-at-loc b1 central_warehouse)
    (box-at-loc b2 central_warehouse)
    (empty-box b1)
    (empty-box b2)

    ; define content located at central_warehouse
    (content-at-loc valve1 central_warehouse)
    (content-at-loc valve2 central_warehouse)
    (content-at-loc bolt1 central_warehouse)
    (content-at-loc bolt2 central_warehouse)
    (content-at-loc hammer1 central_warehouse)
    (content-at-loc hammer2 central_warehouse)

    ; define workstation located at central_warehouse
    (ws-at-loc ws1 loc1)
    (ws-at-loc ws2 loc1)
    (ws-at-loc ws3 loc2)
```

```
    (ws-at-loc ws4 loc2)
    (ws-at-loc ws5 loc2)

    ; define workstation located at central_warehouse
    (connected central_warehouse loc1)
    (connected central_warehouse loc2)
    (connected loc1 central_warehouse)
    (connected loc2 central_warehouse)

    ; define agent located initially located at central_warehouse
    (agent-at-loc a1 central_warehouse)
    (carrier-at-agent c1 a1)
    (place-at-carrier p1 c1)
    (empty-place p1)
  )
  (:goal
    ; define goals to satisfied
    (and
      (content-type-at-ws bolt ws1)
      (content-type-at-ws valve ws1)
      (content-type-at-ws bolt ws2)
      (content-type-at-ws hammer ws3)
      (content-type-at-ws valve ws3)
      (content-type-at-ws hammer ws4)
    )
   )
  )
```

Abbiamo eseguito l'istanza con Fast Forward (FF) utilizzando il seguente comando:

```
(planutils) root@bf885b8679bc:/workspace/ProgettoAI/punto1$ ff domain.pddl
./problems/instance1.pddl
```

Questo comando ha utilizzato il dominio definito nel file domain.pddl e l'istanza del problema specificata nel file ./problems/instance1.pddl. Il risultato è il seguente:

```
ff: found legal plan as follows

step    0: PICK-UP CENTRAL_WAREHOUSE B2 A1 C1 P1
        1: FILL B2 BOLT1 CENTRAL_WAREHOUSE A1 C1 P1
        2: MOVE CENTRAL_WAREHOUSE LOC1 A1
        3: DELIVER B2 WS2 LOC1 A1 C1 P1
        4: EMPTY B2 BOLT1 BOLT WS2 A1 LOC1
        5: PICK-UP LOC1 B2 A1 C1 P1
        6: MOVE LOC1 CENTRAL_WAREHOUSE A1
        7: FILL B2 BOLT2 CENTRAL_WAREHOUSE A1 C1 P1
        8: MOVE CENTRAL_WAREHOUSE LOC1 A1
        9: DELIVER B2 WS1 LOC1 A1 C1 P1
       10: EMPTY B2 BOLT2 BOLT WS1 A1 LOC1
       11: PICK-UP LOC1 B2 A1 C1 P1
       12: MOVE LOC1 CENTRAL_WAREHOUSE A1
       13: FILL B2 VALVE1 CENTRAL_WAREHOUSE A1 C1 P1
       14: MOVE CENTRAL_WAREHOUSE LOC1 A1
       15: DELIVER B2 WS1 LOC1 A1 C1 P1
       16: EMPTY B2 VALVE1 VALVE WS1 A1 LOC1
       17: MOVE LOC1 CENTRAL_WAREHOUSE A1
       18: PICK-UP CENTRAL_WAREHOUSE B1 A1 C1 P1
       19: FILL B1 VALVE2 CENTRAL_WAREHOUSE A1 C1 P1
       20: MOVE CENTRAL_WAREHOUSE LOC2 A1
       21: DELIVER B1 WS3 LOC2 A1 C1 P1
       22: EMPTY B1 VALVE2 VALVE WS3 A1 LOC2
       23: PICK-UP LOC2 B1 A1 C1 P1
       24: MOVE LOC2 CENTRAL_WAREHOUSE A1
       25: FILL B1 HAMMER1 CENTRAL_WAREHOUSE A1 C1 P1
       26: MOVE CENTRAL_WAREHOUSE LOC2 A1
       27: DELIVER B1 WS3 LOC2 A1 C1 P1
       28: EMPTY B1 HAMMER1 HAMMER WS3 A1 LOC2
       29: PICK-UP LOC2 B1 A1 C1 P1
       30: MOVE LOC2 CENTRAL_WAREHOUSE A1
       31: FILL B1 HAMMER2 CENTRAL_WAREHOUSE A1 C1 P1
       32: MOVE CENTRAL_WAREHOUSE LOC2 A1
       33: DELIVER B1 WS4 LOC2 A1 C1 P1
       34: EMPTY B1 HAMMER2 HAMMER WS4 A1 LOC2


time spent:    0.00 seconds instantiating 116 easy, 0 hard action templates
               0.00 seconds reachability analysis, yielding 57 facts and 92 actions
               0.00 seconds creating final representation with 57 relevant facts
               0.00 seconds building connectivity graph
               0.00 seconds searching, evaluating 100 states, to a max depth of 5
               0.00 seconds total time
```

Fast Forward esplora lo spazio degli stati partendo dallo stato iniziale e cercando di raggiungere uno stato che soddisfi l'obiettivo.

2.2 Istanza 2

La seconda istanza ricalca la prima per quanto riguarda condizioni iniziali e goal, ma considera più agenti robotici che possano trasportare più di una scatola alla volta per mezzo di un carrier, la cui capacità è stata modellata tramite il tipo *place*.
Di seguito sono state riportate le modifiche rispetto all'istanza precedente

```
(:objects
  ; define the world
  central_warehouse loc1 loc2 - location
  ws1 ws2 ws3 ws4 ws5 - workstation
  b1 b2 b3 b4 - box
```

```
  a1 a2 - agent
  c1 c2 - carrier
  p11 p12 p21 p22 - place
  valve1 valve2 bolt1 bolt2 hammer1 hammer2 - content
  valve bolt hammer - content-type
)
```

```
; define the empty box located at central_warehouse
(box-at-loc b1 central_warehouse)
(box-at-loc b2 central_warehouse)
(box-at-loc b3 central_warehouse)
(box-at-loc b4 central_warehouse)
(empty-box b1)
(empty-box b2)
(empty-box b3)
(empty-box b4)
```

```
; define agent located initially located at central_warehouse
(agent-at-loc a1 central_warehouse)
(carrier-at-agent c1 a1)
(place-at-carrier p11 c1)
(place-at-carrier p12 c1)
(empty-place p11)
(empty-place p12)
(agent-at-loc a2 central_warehouse)
(carrier-at-agent c2 a2)
(place-at-carrier p21 c2)
(place-at-carrier p22 c2)
(empty-place p21)
(empty-place p22)
```

Abbiamo eseguito l'istanza con Fast Forward (FF) utilizzando il seguente comando:

```
(planutils) root@bf885b8679bc:/workspace/ProgettoAI/punto1$ ff domain.pddl
./problems/instance2.pddl
```

Questo comando ha utilizzato il dominio definito nel file domain.pddl e l'istanza del problema specificata nel file ./problems/instance2.pddl. Il risultato è il seguente:

```
ff: found legal plan as follows

step    0: MOVE CENTRAL_WAREHOUSE LOC1 A1
        1: PICK-UP CENTRAL_WAREHOUSE B4 A2 C2 P21
        2: FILL B4 BOLT1 CENTRAL_WAREHOUSE A2 C2 P21
        3: MOVE CENTRAL_WAREHOUSE LOC1 A2
        4: DELIVER B4 WS2 LOC1 A2 C2 P21
        5: MOVE LOC1 CENTRAL_WAREHOUSE A2
```

```
 6: PICK-UP CENTRAL_WAREHOUSE B3 A2 C2 P21
 7: EMPTY B4 BOLT1 BOLT WS2 A1 LOC1
 8: FILL B3 BOLT2 CENTRAL_WAREHOUSE A2 C2 P21
 9: MOVE CENTRAL_WAREHOUSE LOC1 A2
10: PICK-UP LOC1 B4 A1 C1 P11
11: DELIVER B4 WS1 LOC1 A1 C1 P11
12: DELIVER B3 WS1 LOC1 A2 C2 P21
13: MOVE LOC1 CENTRAL_WAREHOUSE A2
14: PICK-UP CENTRAL_WAREHOUSE B2 A2 C2 P21
15: EMPTY B3 BOLT2 BOLT WS1 A1 LOC1
16: FILL B2 VALVE1 CENTRAL_WAREHOUSE A2 C2 P21
17: PICK-UP CENTRAL_WAREHOUSE B1 A2 C2 P22
18: FILL B1 HAMMER1 CENTRAL_WAREHOUSE A2 C2 P22
19: MOVE CENTRAL_WAREHOUSE LOC2 A2
20: DELIVER B2 WS3 LOC2 A2 C2 P21
21: PICK-UP LOC1 B3 A1 C1 P11
22: EMPTY B2 VALVE1 VALVE WS3 A2 LOC2
23: MOVE LOC1 CENTRAL_WAREHOUSE A1
24: FILL B3 VALVE2 CENTRAL_WAREHOUSE A1 C1 P11
25: MOVE CENTRAL_WAREHOUSE LOC1 A1
26: DELIVER B3 WS1 LOC1 A1 C1 P11
27: EMPTY B3 VALVE2 VALVE WS1 A1 LOC1
28: DELIVER B1 WS3 LOC2 A2 C2 P22
29: PICK-UP LOC2 B2 A2 C2 P21
30: MOVE LOC2 CENTRAL_WAREHOUSE A2
31: FILL B2 HAMMER2 CENTRAL_WAREHOUSE A2 C2 P21
32: MOVE CENTRAL_WAREHOUSE LOC2 A2
33: DELIVER B2 WS4 LOC2 A2 C2 P21
34: EMPTY B1 HAMMER1 HAMMER WS3 A2 LOC2
35: EMPTY B2 HAMMER2 HAMMER WS4 A2 LOC2


time spent:    0.00 seconds instantiating 664 easy, 0 hard action templates
               0.00 seconds reachability analysis, yielding 107 facts and 472 actions
               0.00 seconds creating final representation with 107 relevant facts
               0.00 seconds building connectivity graph
               0.00 seconds searching, evaluating 364 states, to a max depth of 5
               0.00 seconds total time
```

2.3 Planner

Il planner è stato realizzato con l'ausilio della libreria PDDL4J.
È stata definita la classe *IMPlanner*, che mette a disposizione il metodo *solve* e lo invoca su un dominio ed un problema specificati nel main; all'interno del main viene settato anche il peso dell'euristica e la *splitSize*, ovvero la dimensione massima dei sottoproblemi in cui viene suddiviso il problema principale.

```
public class IMPlanner extends AbstractPlanner {

    public static void main(String[] args) {
        // Percorso alla directory dei benchmark
        Final                String                benchmark                =
"src/main/java/fr/uga/pddl4j/project/benchmark/";
        final String domainName = "domain.pddl";
        final String problemName = "problems/?.pddl";

        // Crea il pianificatore
        final IMPlanner planner = new IMPlanner();
```

```
        // Imposta il dominio del problema da risolvere
        planner.setDomain(benchmark + domainName);

        // Imposta il problema da risolvere
        planner.setProblem(benchmark + problemName);

        // Imposta il timeout della ricerca in secondi
        planner.setTimeout(1000);

        // Imposta il livello di log
        planner.setLogLevel(LogLevel.INFO);

        // Imposta il peso euristico e la dimensione della divisione
        planner.setHeuristicWeight(1.2);
        planner.setSplitSize(2);

        // Risolvi e stampa il risultato
        try {
            planner.solve();
        } catch (InvalidConfigurationException e) {
            e.printStackTrace();
        }
    }

    // Altri metodi e codice della classe
}
```

Il metodo *solve* è basato sulla suddivisione del problema in sottoproblemi tramite il metodo *buildSubproblems* e, per risolverli, utilizza una versione modificata di A* (che verrà approfondita nel paragrafo 2.4). Questa suddivisione viene calcolata in primo luogo in base alla location dei sotto-goal; nel caso in cui con questa prima suddivisione si ottengano sottoproblemi troppo grandi (che superano la dimensione massima di split), si opera una seconda suddivisione in base alle workstation.
La suddivisione in sottoproblemi permette di trattare in modo più semplice istanze dotate di goal complessi, risolvendo i sotto-goal in modo sequenziale e indirizzando gli agenti verso una location alla volta.

```
@Override
public Plan solve(Problem problem) throws ProblemNotSupportedException {
    // Strutture di utilità
    Problem[] subProblems = buildSubproblems(problem);
    Node[] subSolutions = new Node[subProblems.length];
    Plan[] subplans = new Plan[subProblems.length];

    // Parametri
    SearchStrategy.Name strategyName = SearchStrategy.Name.ASTAR;
    StateHeuristic.Name heuristic = StateHeuristic.Name.IM_HEURISTIC;
    int timeout = 1000000;
    double heuristicWeight = 1.2; // Aggiunto valore del peso euristico
```

```
    // Risoluzione con A* e IMHeuristic
    for (int i = 0; i < subProblems.length; i++) {
        if (i == 0) {
            // Risolve il sotto-problema 0
            StateSpaceSearch                        alg                        =
StateSpaceSearch.getInstance(strategyName, heuristic, heuristicWeight);
            subSolutions[i] = alg.searchSolutionNode(subProblems[i]);
            subplans[i] = alg.extractPlan(subSolutions[i], subProblems[i]);
        } else {
            //  Risolve  il  sotto-problema  dal  precedente  sotto-problema
trovato
            subSolutions[i - 1].setParent(null);
            StateSpaceSearch   alg   =   new   IMAStar(timeout,   heuristic,
heuristicWeight, subSolutions[i - 1]);
            subSolutions[i] = alg.searchSolutionNode(subProblems[i]);
            subplans[i] = alg.extractPlan(subSolutions[i], subProblems[i]);
        }

        // Stampa il piano trovato per il sotto-problema corrente

System.out.println("\n=======================================================
=============");
        System.out.printf("Plan %d: \n", i);
        System.out.println("-----------------------------------------------
------------------");
        if (subplans[i] != null) {
            System.out.print(problem.toString(subplans[i]));
        } else {
            System.out.println("Plan empty");
        }

System.out.println("=======================================================
===========");
    }

    // Combina tutti i piani parziali in un piano sequenziale
    List<Action> sol = new LinkedList<>();
    for (Plan p : subplans) {
        if (p != null) {
            sol.addAll(p.actions());
        }
    }

    Plan solution = new SequentialPlan();
    Iterator<Action> it = sol.iterator();
    for (int i = 0; i < sol.size(); i++) {
        solution.add(i, it.next());
    }

    return solution;
}
```

2.4 A* Modificato

Nella classe *IMAStar* è stata definita una versione modificata dell'algoritmo A* per la ricerca di un plan per ogni sottoproblema.
Nel metodo *search* vengono sfruttate tre strutture: *closeSet* contiene i nodi già esplorati, *openSet* contiene i nodi in sospeso e *open* mantiene i nodi ordinati secondo la funzione di A* (g+h).

```
public Node search(final Problem codedProblem) {
    Objects.requireNonNull(codedProblem);
    final long begin = System.currentTimeMillis();
final               StateHeuristic               heuristic               =
StateHeuristic.getInstance(this.getHeuristic(), codedProblem);
    // Get the initial state from the planning problem
    initial.setParent(null);
    final State init = initial;
    // Initialize the closed list of nodes (store the nodes explored)
    final Map<State, Node> closeSet = new HashMap<>();
    final Map<State, Node> openSet = new HashMap<>();
    // Initialize the opened list (store the pending node)
    final double currWeight = getWeight();
    // The list stores the node ordered according to the A* (getFValue = g +
h) function
    final   PriorityQueue<Node>   open   =   new   PriorityQueue<>(100,   new
NodeComparator(currWeight));
    // Creates the root node of the tree search
    final Node root = new Node(init, null, -1, 0, heuristic.estimate(init,
codedProblem.getGoal()));
    // Adds the root to the list of pending nodes
    open.add(root);
    openSet.put(init, root);
```

La ricerca inizia con un ciclo di while che viene eseguito fino a che ci sono ancora nodi in sospeso, la soluzione è *null* e non si è verificato un *timeout*. Per ogni iterazione viene preso il primo nodo della lista ordinata e, se non è soddisfatto il goal, si provano ad applicare gli operatori del problema al nodo: per ogni azione (se è applicabile) vengono applicati gli effetti al nodo successore se le condizioni sono soddisfatte nello stato corrente.

```
    this.resetNodesStatistics();
    Node solution = null;
    final long timeout = this.getTimeout() * 1000;
    long time = 0;
    // Start of the search
    while (!open.isEmpty() && solution == null && time < timeout) {
        // Pop the first node in the pending list open
        final Node current = open.poll();
        openSet.remove(current);
        closeSet.put(current, current);
        // If the goal is satisfy in the current node then extract the search
and return it
        if (current.satisfy(codedProblem.getGoal())) {
            solution = current;
        } else {
```

```
            // Try to apply the operators of the problem to this node
            int index = 0;
            for (Action op : codedProblem.getActions()) {

                // Test if a specified operator is applicable in the current
state
                if (op.isApplicable(current)) {
                    //System.out.println("IS APPLICABLE");
                    Node state = new Node(current);
                    this.setCreatedNodes(this.getCreatedNodes() + 1);

                    // Apply the effect of the applicable operator
                    // Test if the condition of the effect is satisfied in
the current state
                    // Apply the effect to the successor node
                    op.getConditionalEffects().stream().filter(ce         ->
current.satisfy(ce.getCondition()))
                        .forEach(ce -> state.apply(ce.getEffect()));
                    final     double     g     =     current.getCost()     +
op.getCost().getValue();
                    Node result = openSet.get(state);
                    if (result == null) {
                        result = closeSet.get(state);
                        if (result != null) {
                            if (g < result.getCost()) {
                                result.setCost(g);
                                result.setParent(current);
                                result.setAction(index);
                                result.setDepth(current.getDepth() + 1);
                                open.add(result);
                                openSet.put(result, result);
                                closeSet.remove(result);
                            }
                        } else {
                            state.setCost(g);
                            state.setParent(current);
                            state.setAction(index);
                            state.setHeuristic(heuristic.estimate(state,
codedProblem.getGoal()));
                            state.setDepth(current.getDepth() + 1);
                            open.add(state);
                            openSet.put(state, state);
                        }
                    } else if (g < result.getCost()) {
                        result.setCost(g);
                        result.setParent(current);
                        result.setAction(index);
                        result.setDepth(current.getDepth() + 1);
                    }
                }
                index++;
            }
        }
```

```
        // Compute the searching time
        time = System.currentTimeMillis() - begin;
    }

    this.setExploredNodes(closeSet.size());
    this.setPendingNodes(openSet.size());
    this.setMemoryUsed(GraphLayout.parseInstance(closeSet).totalSize()
        + GraphLayout.parseInstance(openSet).totalSize());
    this.setSearchingTime(time);

    // return the search computed or null if no search was found
    return solution;
}
```

2.5 Euristica

Un'euristica specifica per il dominio è stata definita nella classe *IMHeuristic*.
Sono stati sfruttati i Relaxed Planning Graphs, ossia grafi a livelli alternati di proposizioni e azioni, che rappresentano le possibili evoluzioni di uno stato iniziale verso uno stato finale e in cui gli effetti negativi delle azioni sono ignorati; questi semplificano il problema e rendono il calcolo dell'euristica più rapido.
All'interno del metodo *estimate*, che stima la distanza dallo stato corrente al goal, vengono in primo luogo definite alcune strutture di supporto. In seguito, si esegue un controllo sullo stato: se il goal è irraggiungibile o se esiste almeno un content non necessario per una delle workstation relative al goal corrente, non viene eseguita alcuna stima, ma si restituisce il massimo valore possibile; se queste condizioni non sono verificate, invece, si prosegue con il calcolo vero e proprio dell'euristica, che considera quattro componenti:

- *onGoal(state, goal)*
- *counter(state, "empty-box")*
- *counter(state, "content-type-at-ws")*
- *filledBoxAtAgentLocatedAtCW(state)*

```
public class IMHeuristic extends RelaxedGraphHeuristic {
    private final Problem p;
    private final Map<String, List<Integer>> predicates;
    private final Map<Integer, List<String>> fluents;
    private Map<String, Set<String>> wsContentTypeRequested;
    private Map<String, String> filled;
    private Set<String> goalTypeRequested;

    public IMHeuristic(Problem problem) {
        super(problem);
        this.p = problem;
        this.predicates = extractPredicates();
        this.fluents = extractFluents();
    }

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

```
    }

    @Override
    public int estimate(State state, Condition goal) {
        super.setGoal(goal);
        this.expandRelaxedPlanningGraph(state);
        this.wsContentTypeRequested = new HashMap<>();
        this.goalTypeRequested = new HashSet<>();
        // building map {ws: content-types requested}
        // building goal type requested list

        this.filled = new HashMap<>();
        // building filled map

      return !super.isGoalReachable() ||
             existUnnecessaryContentAtWs(state)
             ?                      Integer.MAX_VALUE                      :
               onGoal(state,                                              goal)
               +                counter(state,                 "empty-box")
               -             counter(state,            "content-type-at-ws")
               - filledBoxAtAgentLocatedAtCW(state);
    }
}
```

existUnnecessaryContentAtWs:

La funzione *existUnnecessaryContentAtWs* è progettata per verificare lo stato corrente al fine di determinare se questo tipo di contenuto è presente presso una workstation, ma non è richiesto da almeno una delle workstation target. Questo controllo è cruciale per ottimizzare il processo di pianificazione, garantendo che le risorse non siano allocate inutilmente a workstation che non ne hanno bisogno.

```
/**
 * Check if in the current state exists an invalid configuration.
 * */
private boolean existUnnecessaryContentAtWs(State state) {
    List<Integer> isTypeContentsPreds = this.predicates.get("is-type");
    for (int i : this.predicates.get("box-at-ws")) {
        BitVector tmp = new BitVector();
        tmp.set(i);
        if (state.satisfy(new Condition(tmp, new BitVector()))) {
            String box = this.fluents.get(i).get(1);
            String ws = this.fluents.get(i).get(2);
            if (!wsContentTypeRequested.containsKey(ws)) continue;
            String contentInTheBox = this.filled.get(box);
            if (contentInTheBox == null) continue;
            for (int j : isTypeContentsPreds) {
                BitVector tmp2 = new BitVector();
                tmp2.set(j);
                if (state.satisfy(new Condition(tmp2, new BitVector()))) {
                    String typedContent = this.fluents.get(j).get(1);
                    String type = this.fluents.get(j).get(2);
                    if (
```

```
                        typedContent.equals(contentInTheBox)&&
                         ! wsContentTypeRequested.get(ws).remove(type)
                       ) {
                        return true;
                    }
                }
            }
        }
    }
    return false;
}
```

onGoal:

La funzione *onGoal* conta i predicati contenuti nel goal che sono soddisfatti nello stato corrente:

```
/**
 * Counts the predicates in the goal that are satisfied in the current state.
 * */
private int onGoal(State state, Condition goal) {
    BitVector positiveFluents = goal.getPositiveFluents();
    int count = 0;
    for(
       int i = positiveFluents.nextSetBit(0);
       i>=0;
       i = positiveFluents.nextSetBit(i+1)
    ){
        BitVector tmp = new BitVector();
        tmp.set(i);
        if(state.satisfy(new Condition(tmp,new BitVector())))count++;
    }
    BitVector negativeFluents = goal.getNegativeFluents();
    for(
       int i = negativeFluents.nextSetBit(0);
       i>=0;
       i = negativeFluents.nextSetBit(i+1)
    ){
        BitVector tmp = new BitVector();
        tmp.set(i);
        if(state.satisfy(new Condition(tmp,new BitVector())))count++;
    }
    return goal.cardinality() - count;
}
```

counter:

La funzione *counter* conta i predicati nello stato corrente che hanno il nome passato come parametro.

```
/**
 * Counts the predicates that are satisfied in the current state.
 * */
private int counter(State state, String predName) {
```

```
 int count = 0;
 for (int i : this.predicates.get(predName)) {
 BitVector tmp = new BitVector();
 tmp.set(i);
 if (state.satisfy(new Condition(tmp, new BitVector()))) count++;
 }
 return count;
}
```

filledBoxAtAgentLocatedAtCW:

La funzione *filledBoxAtAgentLocatedAtCW(state) prevede* di contare il numero il numero di box che soddisfano le seguenti condizioni:

- La box è sotto il controllo di un'agente che si trova nella central warehouse;
- La box è riempita da un contenuto;
- Il contenuto della box è del tipo necessario per soddisfare la richiesta di una workstation.

```
/**
  * ounts boxes on the agent's carrier at the central warehouse    that are
fully filled with content necessary to fulfill a goal.
* */
private int filledBoxAtAgentLocatedAtCW(State state) {
    List<String> agents = new LinkedList<>();
    for (int i : this.predicates.get("agent-at-loc")) {
        BitVector tmp = new BitVector();
        tmp.set(i);
        if (state.satisfy(new Condition(tmp, new BitVector()))) {
            String agent = this.fluents.get(i).get(1);
            String loc = this.fluents.get(i).get(2);
            if (loc.equals("central_warehouse"))
                agents.add(agent);
        }
    }
    List<String> carriers = new LinkedList<>();
    for (int i : this.predicates.get("carrier-at-agent")) {
        BitVector tmp = new BitVector();
        tmp.set(i);
        if (state.satisfy(new Condition(tmp, new BitVector()))) {
            String carrier = this.fluents.get(i).get(1);
            String agent = this.fluents.get(i).get(2);
            if (agents.contains(agent))
                carriers.add(carrier);
        }
    }
    List<String> places = new LinkedList<>();
    for (int i : this.predicates.get("place-at-carrier")) {
        BitVector tmp = new BitVector();
        tmp.set(i);
        if (state.satisfy(new Condition(tmp, new BitVector()))) {
            String place = this.fluents.get(i).get(1);
            String carrier = this.fluents.get(i).get(2);
            if (carriers.contains(carrier)) {
                places.add(place);
            }
        }
```

```
    }

    int amount=0;
    List<Integer> isTypeContentsPreds = this.predicates.get("is-type");
    for (int i : this.predicates.get("box-at-place")) {
        BitVector tmp = new BitVector();
        tmp.set(i);
        if (state.satisfy(new Condition(tmp, new BitVector()))) {
            String box = this.fluents.get(i).get(1);
            String place = this.fluents.get(i).get(2);
            if (!filled.containsKey(box) && places.contains(place))
                continue;
            String content = filled.get(box);
            for (int j : isTypeContentsPreds) {
                BitVector tmp2 = new BitVector();
                tmp2.set(j);
                if (state.satisfy(new Condition(tmp2, new BitVector()))) {
                    String typedContent = this.fluents.get(j).get(1);
                    String type = this.fluents.get(j).get(2);
                    if (typedContent.equals(content) &&
goalTypeRequested.remove(type)) {
                        amount++;
                        break;
                    }
                }
            }
            if (goalTypeRequested.isEmpty())
                break;
        }
    }
    return amount;
}
```

2.6 Istanza 1

Il planner che abbiamo implementato utilizza l'euristica sopra descritta e suddivide il problema in quattro sotto-obiettivi, risolvendoli uno alla volta. Nonostante questa suddivisione, il numero totale di azioni eseguite dal planner rimane lo stesso rispetto al Fast Forward, senza peggiorare l'efficienza complessiva. Questo è possibile perché il planner utilizza un solo agente, garantendo così che la soluzione sia ottimale e priva di sovrapposizioni o azioni ridondanti.

```
problem instantiation done successfully (92 actions, 74 fluents)

================================================================
SubProblems instantiated with:
----------------------------------------------------------------

Goal0: (and (content-type-at-ws bolt ws1)
  (content-type-at-ws valve ws1))
Goal1: (and (content-type-at-ws bolt ws2))
Goal2: (and (content-type-at-ws hammer ws3)
  (content-type-at-ws valve ws3))
Goal3: (and (content-type-at-ws hammer ws4))

================================================================
```

```
==========================================================================
Plan 0:
--------------------------------------------------------------------------
00: (    pick-up central_warehouse b2 a1 c1 p1) [0]
01: (fill b2 valve1 central_warehouse a1 c1 p1) [0]
02: (           move central_warehouse loc1 a1) [0]
03: (             deliver b2 ws1 loc1 a1 c1 p1) [0]
04: (        empty b2 valve1 valve ws1 a1 loc1) [0]
05: (           move loc1 central_warehouse a1) [0]
06: (    pick-up central_warehouse b1 a1 c1 p1) [0]
07: ( fill b1 bolt1 central_warehouse a1 c1 p1) [0]
08: (           move central_warehouse loc1 a1) [0]
09: (             deliver b1 ws1 loc1 a1 c1 p1) [0]
10: (          empty b1 bolt1 bolt ws1 a1 loc1) [0]
==========================================================================

==========================================================================
Plan 1:
--------------------------------------------------------------------------
0: (                pick-up loc1 b2 a1 c1 p1) [0]
1: (          move loc1 central_warehouse a1) [0]
2: (fill b2 bolt2 central_warehouse a1 c1 p1) [0]
3: (          move central_warehouse loc1 a1) [0]
4: (            deliver b2 ws2 loc1 a1 c1 p1) [0]
5: (         empty b2 bolt2 bolt ws2 a1 loc1) [0]
==========================================================================

==========================================================================
Plan 2:
--------------------------------------------------------------------------
00: (                  pick-up loc1 b1 a1 c1 p1) [0]
01: (            move loc1 central_warehouse a1) [0]
02: ( fill b1 valve2 central_warehouse a1 c1 p1) [0]
03: (            move central_warehouse loc2 a1) [0]
04: (              deliver b1 ws3 loc2 a1 c1 p1) [0]
05: (         empty b1 valve2 valve ws3 a1 loc2) [0]
06: (                  pick-up loc2 b1 a1 c1 p1) [0]
07: (            move loc2 central_warehouse a1) [0]
08: (fill b1 hammer2 central_warehouse a1 c1 p1) [0]
09: (            move central_warehouse loc2 a1) [0]
10: (              deliver b1 ws3 loc2 a1 c1 p1) [0]
11: (       empty b1 hammer2 hammer ws3 a1 loc2) [0]
==========================================================================

==========================================================================
Plan 3:
--------------------------------------------------------------------------
0: (                  pick-up loc2 b1 a1 c1 p1) [0]
1: (            move loc2 central_warehouse a1) [0]
2: (fill b1 hammer1 central_warehouse a1 c1 p1) [0]
3: (            move central_warehouse loc2 a1) [0]
4: (              deliver b1 ws4 loc2 a1 c1 p1) [0]
5: (       empty b1 hammer1 hammer ws4 a1 loc2) [0]
==========================================================================
```

```
found plan as follows:

00: (     pick-up central_warehouse b2 a1 c1 p1) [0]
01: ( fill b2 valve1 central_warehouse a1 c1 p1) [0]
02: (            move central_warehouse loc1 a1) [0]
03: (              deliver b2 ws1 loc1 a1 c1 p1) [0]
04: (         empty b2 valve1 valve ws1 a1 loc1) [0]
05: (            move loc1 central_warehouse a1) [0]
06: (     pick-up central_warehouse b1 a1 c1 p1) [0]
07: (  fill b1 bolt1 central_warehouse a1 c1 p1) [0]
08: (            move central_warehouse loc1 a1) [0]
09: (              deliver b1 ws1 loc1 a1 c1 p1) [0]
10: (           empty b1 bolt1 bolt ws1 a1 loc1) [0]
11: (                  pick-up loc1 b2 a1 c1 p1) [0]
12: (            move loc1 central_warehouse a1) [0]
13: (  fill b2 bolt2 central_warehouse a1 c1 p1) [0]
14: (            move central_warehouse loc1 a1) [0]
15: (              deliver b2 ws2 loc1 a1 c1 p1) [0]
16: (           empty b2 bolt2 bolt ws2 a1 loc1) [0]
17: (                  pick-up loc1 b1 a1 c1 p1) [0]
18: (            move loc1 central_warehouse a1) [0]
19: ( fill b1 valve2 central_warehouse a1 c1 p1) [0]
20: (            move central_warehouse loc2 a1) [0]
21: (              deliver b1 ws3 loc2 a1 c1 p1) [0]
22: (         empty b1 valve2 valve ws3 a1 loc2) [0]
23: (                  pick-up loc2 b1 a1 c1 p1) [0]
24: (            move loc2 central_warehouse a1) [0]
25: (fill b1 hammer2 central_warehouse a1 c1 p1) [0]
26: (            move central_warehouse loc2 a1) [0]
27: (              deliver b1 ws3 loc2 a1 c1 p1) [0]
28: (       empty b1 hammer2 hammer ws3 a1 loc2) [0]
29: (                  pick-up loc2 b1 a1 c1 p1) [0]
30: (            move loc2 central_warehouse a1) [0]
31: (fill b1 hammer1 central_warehouse a1 c1 p1) [0]
32: (            move central_warehouse loc2 a1) [0]
33: (              deliver b1 ws4 loc2 a1 c1 p1) [0]
34: (       empty b1 hammer1 hammer ws4 a1 loc2) [0]

time spent:       0,04 seconds parsing
                  0,08 seconds encoding
                  0,00 seconds searching
                  0,11 seconds total time

memory used:      0,48 MBytes for problem representation
                  0,00 MBytes for searching
                  0,48 MBytes total
```

2.7 Istanza 2

Il planner ottimizza il numero di azioni caricando i carrier al massimo delle loro capacità quando possibile, riducendo così le azioni da 35 a 29 rispetto al planner precedente. Questo approccio utilizza un'euristica che favorisce configurazioni efficienti:

- Riempie completamente il carrier dell'agente a1, che ha il compito di soddisfare goal0 in una singola locazione.
- Pianifica anche per il futuro, riempiendo i carrier dell'agente a2 con contenuti che saranno necessari per altri obiettivi.

```
problem instantiation done successfully (472 actions, 128 fluents)



=================================================================
SubProblems instantiated with:
-----------------------------------------------------------------

Goal0: (and (content-type-at-ws bolt ws1)
  (content-type-at-ws valve ws1))
Goal1: (and (content-type-at-ws bolt ws2))
Goal2: (and (content-type-at-ws hammer ws3)
  (content-type-at-ws valve ws3))
Goal3: (and (content-type-at-ws hammer ws4))

=================================================================

=================================================================
Plan 0:
-----------------------------------------------------------------
00: (    pick-up central_warehouse b4 a2 c2 p22) [0]
01: ( fill b4 bolt2 central_warehouse a2 c2 p22) [0]
02: (    pick-up central_warehouse b3 a1 c1 p12) [0]
03: (fill b3 valve1 central_warehouse a1 c1 p12) [0]
04: (    pick-up central_warehouse b2 a1 c1 p11) [0]
05: ( fill b2 bolt1 central_warehouse a1 c1 p11) [0]
06: (            move central_warehouse loc1 a1) [0]
07: (             deliver b2 ws1 loc1 a1 c1 p11) [0]
08: (           empty b2 bolt1 bolt ws1 a1 loc1) [0]
09: (    pick-up central_warehouse b1 a2 c2 p21) [0]
10: (fill b1 valve2 central_warehouse a2 c2 p21) [0]
11: (             deliver b3 ws1 loc1 a1 c1 p12) [0]
12: (         empty b3 valve1 valve ws1 a1 loc1) [0]
=================================================================

=================================================================
Plan 1:
-----------------------------------------------------------------
0: ( move central_warehouse loc1 a2) [0]
1: (  deliver b4 ws2 loc1 a2 c2 p22) [0]
2: (empty b4 bolt2 bolt ws2 a1 loc1) [0]
=================================================================

=================================================================
Plan 2:
-----------------------------------------------------------------
0: (                 pick-up loc1 b2 a2 c2 p22) [0]
1: (             move loc1 central_warehouse a2) [0]
2: (fill b2 hammer1 central_warehouse a2 c2 p22) [0]
3: (             move central_warehouse loc2 a2) [0]
```

```
4: (              deliver b1 ws3 loc2 a2 c2 p21) [0]
5: (          empty b1 valve2 valve ws3 a2 loc2) [0]
6: (              deliver b2 ws3 loc2 a2 c2 p22) [0]
7: (        empty b2 hammer1 hammer ws3 a2 loc2) [0]
==========================================================================

==========================================================================
Plan 3:
--------------------------------------------------------------------------
0: (                  pick-up loc2 b1 a2 c2 p22) [0]
1: (             move loc2 central_warehouse a2) [0]
2: (fill b1 hammer2 central_warehouse a2 c2 p22) [0]
3: (             move central_warehouse loc2 a2) [0]
4: (              deliver b1 ws4 loc2 a2 c2 p22) [0]
5: (        empty b1 hammer2 hammer ws4 a2 loc2) [0]
==========================================================================

found plan as follows:

00: (     pick-up central_warehouse b4 a2 c2 p22) [0]
01: (  fill b4 bolt2 central_warehouse a2 c2 p22) [0]
02: (     pick-up central_warehouse b3 a1 c1 p12) [0]
03: ( fill b3 valve1 central_warehouse a1 c1 p12) [0]
04: (     pick-up central_warehouse b2 a1 c1 p11) [0]
05: (  fill b2 bolt1 central_warehouse a1 c1 p11) [0]
06: (              move central_warehouse loc1 a1) [0]
07: (               deliver b2 ws1 loc1 a1 c1 p11) [0]
08: (             empty b2 bolt1 bolt ws1 a1 loc1) [0]
09: (     pick-up central_warehouse b1 a2 c2 p21) [0]
10: ( fill b1 valve2 central_warehouse a2 c2 p21) [0]
11: (               deliver b3 ws1 loc1 a1 c1 p12) [0]
12: (           empty b3 valve1 valve ws1 a1 loc1) [0]
13: (              move central_warehouse loc1 a2) [0]
14: (               deliver b4 ws2 loc1 a2 c2 p22) [0]
15: (             empty b4 bolt2 bolt ws2 a1 loc1) [0]
16: (                   pick-up loc1 b2 a2 c2 p22) [0]
17: (              move loc1 central_warehouse a2) [0]
18: (fill b2 hammer1 central_warehouse a2 c2 p22) [0]
19: (              move central_warehouse loc2 a2) [0]
20: (               deliver b1 ws3 loc2 a2 c2 p21) [0]
21: (           empty b1 valve2 valve ws3 a2 loc2) [0]
22: (               deliver b2 ws3 loc2 a2 c2 p22) [0]
23: (         empty b2 hammer1 hammer ws3 a2 loc2) [0]
24: (                   pick-up loc2 b1 a2 c2 p22) [0]
25: (              move loc2 central_warehouse a2) [0]
26: (fill b1 hammer2 central_warehouse a2 c2 p22) [0]
27: (              move central_warehouse loc2 a2) [0]
28: (               deliver b1 ws4 loc2 a2 c2 p22) [0]
29: (         empty b1 hammer2 hammer ws4 a2 loc2) [0]

time spent:       0,03 seconds parsing
                  0,23 seconds encoding
                  0,00 seconds searching
                  0,26 seconds total time
```

```
memory used:      1,80 MBytes for problem representation
                  0,00 MBytes for searching
                  1,80 MBytes total
```

Infine, si è eseguito un confronto con FAST_FORWARD e MAX, euristiche offerte dalla libreria pddl4j. I test sono stati eseguiti definendo:

- Split size = 2
- Heuristic weigth = 1.2

Il risultato più importante lo troviamo all'istanza 4, qui evidente che la nostra euristica fa un'azione in più rispetto al fast forward.

Euristica/Istanza	P01	P02	P03	P04
IMHEURISTIC	Memory usage: 0.20 MB Time spent: 0.10s #actions: 22	Memory usage: 0.39MB Time spent: 0.12s #actions: 20	Memory usage: 2.29MB Time spent: 0.37s #actions: 51	Memory usage: 4.96MB Time spent: 0.62s #actions: 53
FF	Memory usage: 0.20MB Time spent:12s #actions: 22	Memory usage: 0.39MB Time spent: 0.11s #actions: 20	Memory usage: 2.29MB Time spent: 27s #actions: 51	Memory usage: 4.96 MB Time spent:0.41s #actions: 55
MAX	Memory usage: 0.20MB Time spent: 0.10s #actions: 22	Memory usage: 0.39 Time spent: 0.10s #actions: 20	Memory usage: 2.29MB Time spent:0.23s #actions: 51	Not solved in less than 1min

3 Temporal Planning & Robotics

3.1 Temporal Planning

Il dominio precedentemente definito è stato esteso così da supportare le *durative actions*, ossia è stata associata una durata temporale ad ogni azione.
Di seguito vengono riportate le modifiche apportate rispetto alla versione commentata nel capitolo 1.

In primo luogo, ai *requirements* è stato aggiunto il riferimento alle *durative-actions*:

```
(:requirements :strips :typing :fluents :durative-actions)
```

Tra i predicati è stato aggiunto *free-agent*, per evitare che si verifichi l'esecuzione contemporanea di due azioni diverse da parte dello stesso agente:

```
(:predicates
    [...]
    (free-agent ?agent - agent)
)
```

Sono state, inoltre, definite cinque funzioni che indicano la durata delle diverse azioni:

```
(:functions
    (pick-duration)
    (fill-duration)
    (move-duration)
    (deliver-duration)
    (empty-duration)
)
```

Le azioni precedentemente definite sono state modificate; in particolare, oltre all'aggiunta della specifica della durata, ne sono state modificate le *condition* e gli *effect*, utilizzando i costrutti *at start*, *over all* e *at end*:

- Il costrutto *at start* indica una condizione o un effetto vero all'inizio dell'azione.
- Il costrutto *over all* indica una condizione valida per tutta la durata dell'azione.
- Il costrutto *at end* indica un effetto vero al completamento dell'azione.

L'utilizzo di tali costrutti ha permesso, ad esempio, di evitare che la stessa box possa essere presa da due agenti robotici contemporaneamente, cambiandone la location non appena inizia un'azione di *pick up*, invece che al termine dell'azione.

```
(:durative-action pick-up
    :parameters (
      ?loc - location
      ?box - box
      ?agent - agent
      ?carrier - carrier
      ?place - place
    )
    :duration (= ?duration (pick-duration))
    :condition (and
        (at start (free-agent ?agent))
        (over all (agent-at-loc ?agent ?loc))
        (at start (box-at-loc ?box ?loc))
        (over all (carrier-at-agent ?carrier ?agent))
        (over all (place-at-carrier ?place ?carrier))
        (over all (empty-place ?place))
    )
    :effect (and
        (at start (not(free-agent ?agent)))
        (at end (box-at-place ?box ?place))
        (at end (not (empty-place ?place)))
        (at start (not (box-at-loc ?box ?loc)))
        (at end (free-agent ?agent))
    )
)

(:durative-action move
```

```
    :parameters (?from ?to - location ?agent - agent)
    :duration (= ?duration (move-duration))
    :condition (and
        (at start (free-agent ?agent))
        (over all (connected ?from ?to))
        (at start (agent-at-loc ?agent ?from))
    )
    :effect (and
        (at start (not(free-agent ?agent)))
        (at start (not (agent-at-loc ?agent ?from)))
        (at end (agent-at-loc ?agent ?to))
        (at end (free-agent ?agent))
    )
)

(:durative-action deliver
    :parameters (
      ?box - box
      ?ws - workstation
      ?location - location
      ?agent - agent
      ?carrier - carrier
      ?place - place
    )
    :duration (= ?duration (deliver-duration))
    :condition (and
        (at start (free-agent ?agent))
        (over all (ws-at-loc ?ws ?location))
        (over all (agent-at-loc ?agent ?location))
        (over all (carrier-at-agent ?carrier ?agent))
        (over all (place-at-carrier ?place ?carrier))
        (over all (box-at-place ?box ?place))
    )
    :effect (and
        (at start (not(free-agent ?agent)))
        (at end (not (box-at-place ?box ?place)))
        (at end (empty-place ?place))
        (at end (box-at-ws ?box ?ws))
        (at end (free-agent ?agent))
    )
)

(:durative-action fill
    :parameters (
      ?box - box
      ?content - content
      ?loc - location
      ?agent - agent
      ?carrier - carrier
      ?place - place
    )
    :duration (= ?duration (fill-duration))
    :condition (and
        (at start (free-agent ?agent))
        (at start (content-at-loc ?content ?loc))
        (over all (agent-at-loc ?agent ?loc))
```

```
        (over all (carrier-at-agent ?carrier ?agent))
        (over all (place-at-carrier ?place ?carrier))
        (over all (box-at-place ?box ?place))
        (over all (empty-box ?box))
    )
    :effect (and
        (at start (not(free-agent ?agent)))
        (at end (not (empty-box ?box)))
        (at end (filled-box ?box ?content))
        (at start (not (content-at-loc ?content ?loc)))
        (at end (free-agent ?agent))
    )
)

(:durative-action empty
    :parameters (
      ?box - box
      ?content - content
      ?t - content-type
      ?ws - workstation
      ?agent - agent
      ?loc - location
    )
    :duration (= ?duration (empty-duration))
    :condition (and
        (at start (free-agent ?agent))
        (over all (is-type ?content ?t))
        (at start (box-at-ws ?box ?ws))
        (over all (ws-at-loc ?ws ?loc))
        (over all (filled-box ?box ?content))
        (over all (agent-at-loc ?agent ?loc))
    )
    :effect (and
        (at start (not(free-agent ?agent)))
        (at end (not (filled-box ?box ?content)))
        (at end (empty-box ?box))
        (at end (box-at-loc ?box ?loc))
        (at start (not (box-at-ws ?box ?ws)))
        (at end (content-type-at-ws ?t ?ws))
        (at end (free-agent ?agent))
    )
)
```

A questo punto una delle istanze definite per il punto precedente è stata modificata in modo da rispettare il nuovo dominio: nello specifico, nella sezione *init* sono state indicate le durate relative ad ogni azione.

```
(:init
    […]
    (free-agent a1)
    (free-agent a2)
    (= (pick-duration) 2)
    (= (fill-duration) 3)
    (= (move-duration) 5)
    (= (deliver-duration) 2)
```

```
    (= (empty-duration) 3)
)
```

Con l'ausilio di planutils è stato generato un piano per l'istanza. La scelta del planner è ricaduta su LPG-TD, un'estensione di LPG (Local search for Planning Graphs), un planner basato sulla ricerca locale e i planning graph compatibile con le *durative actions*. Di seguito viene riportato l'output ottenuto:

```
 0.0003: (MOVE CENTRAL_WAREHOUSE LOC1 A1) [D:5.0000; C:1.0000]
 0.0005: (PICK-UP CENTRAL_WAREHOUSE B1 A2 C2 P21) [D:2.0000; C:1.0000]
 2.0008: (FILL B1 BOLT3 CENTRAL_WAREHOUSE A2 C2 P21) [D:3.0000; C:1.0000]
 5.0010: (MOVE CENTRAL_WAREHOUSE LOC1 A2) [D:5.0000; C:1.0000]
 10.0013: (DELIVER B1 WS1 LOC1 A2 C2 P21) [D:2.0000; C:1.0000]
 12.0015: (EMPTY B1 BOLT3 BOLT WS1 A1 LOC1) [D:3.0000; C:1.0000]
 15.0017: (MOVE LOC1 CENTRAL_WAREHOUSE A1) [D:5.0000; C:1.0000]
 20.0020: (MOVE CENTRAL_WAREHOUSE LOC3 A1) [D:5.0000; C:1.0000]
 12.0022: (MOVE LOC1 CENTRAL_WAREHOUSE A2) [D:5.0000; C:1.0000]
 17.0025: (PICK-UP CENTRAL_WAREHOUSE B2 A2 C2 P21) [D:2.0000; C:1.0000]
 19.0028: (FILL B2 TOOL1 CENTRAL_WAREHOUSE A2 C2 P21) [D:3.0000; C:1.0000]
 22.0030: (MOVE CENTRAL_WAREHOUSE LOC3 A2) [D:5.0000; C:1.0000]
 27.0033: (DELIVER B2 WS4 LOC3 A2 C2 P21) [D:2.0000; C:1.0000]
 29.0035: (EMPTY B2 TOOL1 TOOL WS4 A1 LOC3) [D:3.0000; C:1.0000]
 32.0037: (PICK-UP LOC3 B2 A1 C1 P11) [D:2.0000; C:1.0000]
 34.0040: (MOVE LOC3 CENTRAL_WAREHOUSE A1) [D:5.0000; C:1.0000]
 39.0042: (FILL B2 BOLT1 CENTRAL_WAREHOUSE A1 C1 P11) [D:3.0000; C:1.0000]
 42.0045: (MOVE CENTRAL_WAREHOUSE LOC3 A1) [D:5.0000; C:1.0000]
 47.0047: (DELIVER B2 WS3 LOC3 A1 C1 P11) [D:2.0000; C:1.0000]
 49.0050: (EMPTY B2 BOLT1 BOLT WS3 A2 LOC3) [D:3.0000; C:1.0000]
 52.0052: (PICK-UP LOC3 B2 A1 C1 P11) [D:2.0000; C:1.0000]
 54.0055: (MOVE LOC3 CENTRAL_WAREHOUSE A1) [D:5.0000; C:1.0000]
 59.0057: (FILL B2 VALVE1 CENTRAL_WAREHOUSE A1 C1 P11) [D:3.0000; C:1.0000]
 62.0060: (MOVE CENTRAL_WAREHOUSE LOC3 A1) [D:5.0000; C:1.0000]
 67.0062: (DELIVER B2 WS3 LOC3 A1 C1 P11) [D:2.0000; C:1.0000]
 69.0065: (EMPTY B2 VALVE1 VALVE WS3 A2 LOC3) [D:3.0000; C:1.0000]
 69.0068: (MOVE LOC3 CENTRAL_WAREHOUSE A1) [D:5.0000; C:1.0000]
 74.0070: (MOVE CENTRAL_WAREHOUSE LOC2 A1) [D:5.0000; C:1.0000]
 72.0072: (PICK-UP LOC3 B2 A2 C2 P21) [D:2.0000; C:1.0000]
 74.0075: (MOVE LOC3 CENTRAL_WAREHOUSE A2) [D:5.0000; C:1.0000]
 79.0078: (FILL B2 VALVE2 CENTRAL_WAREHOUSE A2 C2 P21) [D:3.0000; C:1.0000]
 82.0080: (MOVE CENTRAL_WAREHOUSE LOC2 A2) [D:5.0000; C:1.0000]
 87.0082: (DELIVER B2 WS2 LOC2 A2 C2 P21) [D:2.0000; C:1.0000]
 89.0085: (EMPTY B2 VALVE2 VALVE WS2 A1 LOC2) [D:3.0000; C:1.0000]
 92.0088: (PICK-UP LOC2 B2 A1 C1 P11) [D:2.0000; C:1.0000]
 94.0090: (MOVE LOC2 CENTRAL_WAREHOUSE A1) [D:5.0000; C:1.0000]
 99.0092: (FILL B2 BOLT2 CENTRAL_WAREHOUSE A1 C1 P11) [D:3.0000; C:1.0000]
 102.0095: (MOVE CENTRAL_WAREHOUSE LOC2 A1) [D:5.0000; C:1.0000]
 107.0098: (DELIVER B2 WS2 LOC2 A1 C1 P11) [D:2.0000; C:1.0000]
 109.0100: (EMPTY B2 BOLT2 BOLT WS2 A2 LOC2) [D:3.0000; C:1.0000]
```

3.2 Robotics Planning

Per eseguire il piano mediante l'utilizzo di PlanSys2 è stato creato il workspace *project_ai e* sono stati definiti:

- Il file di launch in Python, necessario per avviare i nodi ROS.

```
def generate_launch_description():
    # Get the launch directory
    example_dir = get_package_share_directory('project_ai')
    namespace = LaunchConfiguration('namespace')

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

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

    # Specify the actions
    pickup_cmd = Node(
        package='project_ai',
        executable='pickup_action',
        name='pickup_action',
        namespace=namespace,
        output='screen',
        parameters=[])

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

    move_cmd = Node(
        package='project_ai',
        executable='move_action',
        name='move_action',
        namespace=namespace,
        output='screen',
        parameters=[])

    deliver_cmd = Node(
        package='project_ai',
        executable='deliver_action',
        name='deliver_action',
        namespace=namespace,
        output='screen',
        parameters=[])

    empty_cmd = Node(
```

```
        package='project_ai',
        executable='empty_action',
        name='empty_action',
        namespace=namespace,
        output='screen',
        parameters=[])
    ld = LaunchDescription()

    ld.add_action(declare_namespace_cmd)

    # Declare the launch options
    ld.add_action(plansys2_cmd)

    ld.add_action(pickup_cmd)
    ld.add_action(fill_cmd)
    ld.add_action(move_cmd)
    ld.add_action(deliver_cmd)
    ld.add_action(empty_cmd)

    return ld
```

- Le *fake actions* in C++; di seguito è stata riportata quella relativa alla *move*.

```
#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 MoveAction : public plansys2::ActionExecutorClient
{
public:
  MoveAction()
  : plansys2::ActionExecutorClient("move", 200ms)
  {
    progress_ = 0.0;
  }

private:
  void do_work()
  {
    if (progress_ < 1.0) {
      progress_ += 0.02;
      send_feedback(progress_, "Move running");
    } else {
      finish(true, 1.0, "Move completed");

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

    std::cout << "\r\e[K" << std::flush;
    std::cout << "Moving ... [" << 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<MoveAction>();

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

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

  rclcpp::shutdown();

  return 0;
}
```

- I file *CMakeLists.txt* e *package.xml* per effettuare la build del progetto e settare le dipendenze.

Il workspace contiene le seguenti cartelle:

- *src*: contiene i file C++ delle fake actions.
- *pddl*: contiene il file di dominio.
- *launch*: contiene il file *launch.py*, il file di testo con i comandi necessari per istanziare il problema ed il plan generato con planutils, opportunamente modificato per essere letto da PlanSys2.

In seguito alla build del progetto sono state generate anche le cartelle *build*, *install* e *log*.
Il progetto è stato lanciato tramite i comandi seguenti:

```
colcon build --symlink-install
source install/setup.bash
ros2 launch project_ai launch.py
```

Infine, il problema è stato istanziato da un secondo terminale con il comando:

```
ros2 run plansys2_terminal plansys2_terminal
```

E tramite i comandi definiti nel file *commands.txt*.