



Intelligenza Artificiale

Relazione progetto didattico - Emergency System

Università della Calabria

Teodoro De Paola - 242592

Aurelio Marotta - 242452

Francesco Nicoletti - 248231





Contents

1	Modeling	5
1.1	Obiettivi Progettuali	5
1.2	Implementazione in PDDL	6
1.2.1	I Tipi in PDDL	6
1.2.2	I Predicati in PDDL	6
1.2.3	Le Azioni in PDDL	6
1.2.4	Il file domain.pddl	6
1.2.5	Istanze del Problema	10
1.2.6	Istanza 1 del Problema	11
1.2.7	Istanza 2 del Problema	12
1.2.8	Istanza 3 del Problema	13
1.2.9	Output delle tre istanze	15
2	Classical Planning	17
2.1	Obiettivi	17
2.2	Struttura delle Classi	17
2.2.1	myASP	18
2.2.2	MyNode e MyHeuristic	30
2.3	Risultati euristica sulle tre istanze	40
3	Temporal Planning and Robotic	43
3.1	Temporal Planning	43
3.2	Robotics Planning	49
3.2.1	Launch-File	51
3.2.2	Command File	53
3.2.3	Plan.txt	55

3.2.4	CMakeList File	55
3.2.5	Organizzazione del Workspace	57
3.2.6	Risultati	57
4	Deliverables	61
4.1	Organizzazione cartella progetto	61



1. Modeling

1.1 Obiettivi Progettuali

Il progetto richiede la modellazione di uno scenario utilizzando il linguaggio **PDDL 1.2**, ispirato alla consegna di beni in una situazione di emergenza. In questo scenario, l'obiettivo principale è coordinare agenti robotici nella consegna di scatole contenenti beni necessari a persone ferite, che si trovano in posizioni predefinite. Tutti gli agenti robotici, i carrier utilizzati per trasportare le scatole, le scatole stesse e i beni da consegnare sono inizialmente collocati in un deposito centrale. È importante notare che è sempre disponibile una quantità sufficiente di beni per soddisfare le necessità delle persone ferite. Nella progettazione di questo scenario, è necessario tenere conto di alcune specifiche:

- **Posizioni delle Persone Ferite:** In una stessa posizione possono trovarsi più persone ferite, quindi è necessario coordinare la consegna dei beni a più destinatari contemporaneamente.
- **Capacità dei Carrier:** La capacità dei carrier, cioè dei veicoli utilizzati per trasportare le scatole, è limitata ma varia da una istanza del problema all'altra. Questo significa che la capacità di ciascun veicolo può essere differente in base alla situazione specifica.
- **Azioni degli Agenti Robotici:** Gli agenti robotici possono svolgere varie azioni, tra cui riempire e svuotare scatole, caricare o scaricare scatole da un carrier e spostarsi da una posizione all'altra sia con che senza un carrier.
- **Flessibilità nei Percorsi:** Non esistono percorsi predefiniti tra le diverse posizioni nel sistema. I robot hanno la libertà di spostarsi tra le posizioni in modo arbitrario, il che richiede una pianificazione efficace per garantire una consegna efficiente dei beni.

1.2 Implementazione in PDDL

Il risultato dello svolgimento del primo punto è il file **domain.pddl**, che modella il dominio del problema mediante tipi, predicati e azioni.

1.2.1 I Tipi in PDDL

- I tipi sono usati per definire categorie o insiemi di oggetti all'interno del dominio del problema. Ad esempio, tipi come "robot", "scatola", "persona", ecc.
- I tipi permettono di raggruppare oggetti simili in categorie per semplificare la descrizione del dominio.
- In PDDL, i tipi vengono definiti con il comando **:types** seguito da una lista dei tipi disponibili.

1.2.2 I Predicati in PDDL

- I predicati sono dichiarazioni logiche che rappresentano relazioni o proprietà tra gli oggetti all'interno del dominio.
- Ad esempio, un predicato come "ferito(x)" rappresenta che una persona è ferita.
- I predicati vengono utilizzati per descrivere lo stato iniziale e gli obiettivi del problema.
- In PDDL, i predicati vengono dichiarati con il comando **:predicates** seguito da una lista di predicati disponibili.

1.2.3 Le Azioni in PDDL

- Le azioni definiscono le operazioni che possono essere eseguite dai piani per raggiungere gli obiettivi del problema.
- Ciascuna azione specifica quali tipi di oggetti sono coinvolti, quali predicati devono essere soddisfatti per eseguire l'azione e quali predicati vengono aggiornati o modificati dopo l'esecuzione dell'azione.
- Le azioni vengono dichiarate con il comando **:action** e includono sezioni come **:parameters** per specificare gli oggetti coinvolti, **:precondition** per specificare i predicati che devono essere veri prima che l'azione possa essere eseguita, e **:effect** per specificare come l'azione modifica lo stato del mondo.

1.2.4 Il file domain.pddl

Listing 1.1: dominio.pddl

```

1 (define (domain ESL)
2
3 ;remove requirements that are not needed
4 (:requirements :adl :universal-preconditions)
5
6 (:types
7   location locatable carrier_space - object
8   person box robot carrier content - locatable
9 )
10
11 (:predicates
12   ;todo: define predicates here
13   (at ?locatable -locatable ?loc -location)
14
15   (isEmpty ?box -box)

```

```

16     (isFilledBy ?box -box ?content -content)
17
18     (needs ?person - person ?content -content)
19
20     (isFree ?carrier -carrier ?carrier_space -carrier_space )
21     (isOccupiedBy ?carrier -carrier ?carrier_space
22         -carrier_space ?box -box)
23
24     (are_same_location ?location1 -location
25         ?location2 -location)
26
27 )
28
29 ;azione con cui un robot raccoglie il contenuto che si trova
30 nella sua posizione e lo ripone in un box
31 (:action fill_box
32     :parameters (?loc -location ?box -box ?content -content
33         ?robot -robot)
34     :precondition (and
35         (at ?box ?loc)
36         (at ?robot ?loc)
37         (at ?content ?loc)
38         (isEmpty ?box)
39     )
40     :effect (and
41         (not(isEmpty ?box))
42         (isFilledBy ?box ?content)
43     )
44 )
45
46 azione con cui il robot svuota il contenuto da un box e lo
47 consegna ad una persona che ne ha bisogno nella sua
48 posizione
49 (:action empty_box_and_deliver
50     :parameters (?loc -location ?box -box ?content -content
51         ?person -person ?robot -robot)
52     :precondition (and
53         (at ?box ?loc)
54         (at ?robot ?loc)
55         (at ?person ?loc)
56         (not(isEmpty ?box))
57         (isFilledBy ?box ?content)
58         (needs ?person ?content)
59     )
60     :effect (and
61         (isEmpty ?box)
62         (not(isFilledBy ?box ?content))
63         (not(needs ?person ?content))
64     )
65 )

```

```

58 )
59
60 ;azione tramite cui il robot pu  caricare un box su un
    carrello al fine di spostarlo successivamente
61 (:action load_carrier
62     :parameters (?loc -location ?box -box ?carrier -carrier
        ?space -carrier_space ?robot -robot )
63     :precondition (and
64         (at ?box ?loc)
65         (at ?robot ?loc)
66         (at ?carrier ?loc)
67         (isFree ?carrier ?space)
68     )
69     :effect (and
70         (not(at ?box ?loc))
71         (not(isFree ?carrier ?space))
72         (isOccupiedBy ?carrier ?space ?box)
73     )
74 )
75
76
77 azione tramite cui un robot scarica una box da un carrello
78 (:action unload_carrier
79     :parameters (?loc -location ?box -box ?carrier -carrier
        ?space - carrier_space ?robot -robot)
80     :precondition (and
81         (at ?robot ?loc)
82         (at ?carrier ?loc)
83         (isOccupiedBy ?carrier ?space ?box)
84     )
85     :effect (and
86         (isFree ?carrier ?space)
87         (not(isOccupiedBy ?carrier ?space ?box))
88         (at ?box ?loc)
89     )
90 )
91
92 (:action move_carrier
93     :parameters (?initial_loc - location ?final_loc -
        location ?carrier - carrier ?robot -robot)
94     :precondition (and
95         (at ?robot ?initial_loc)
96         (at ?carrier ?initial_loc)
97         (not(are_same_location ?initial_loc ?final_loc))
98     )
99     :effect (and
100         (not(at ?robot ?initial_loc))
101         (not(at ?carrier ?initial_loc))
102         (at ?robot ?final_loc)

```



```

103     (at ?carrier ?final_loc)
104   )
105 )
106
107 (:action simply_move
108   :parameters (?initial_loc - location ?final_loc -
109     location ?robot - robot)
110   :precondition (and
111     (at ?robot ?initial_loc)
112     (not(are_same_location ?initial_loc ?final_loc))
113   )
114   :effect (and
115     (not (at ?robot ?initial_loc))
116     (at ?robot ?final_loc)
117   )
118 )

```

- Definizione dei Tipi :types

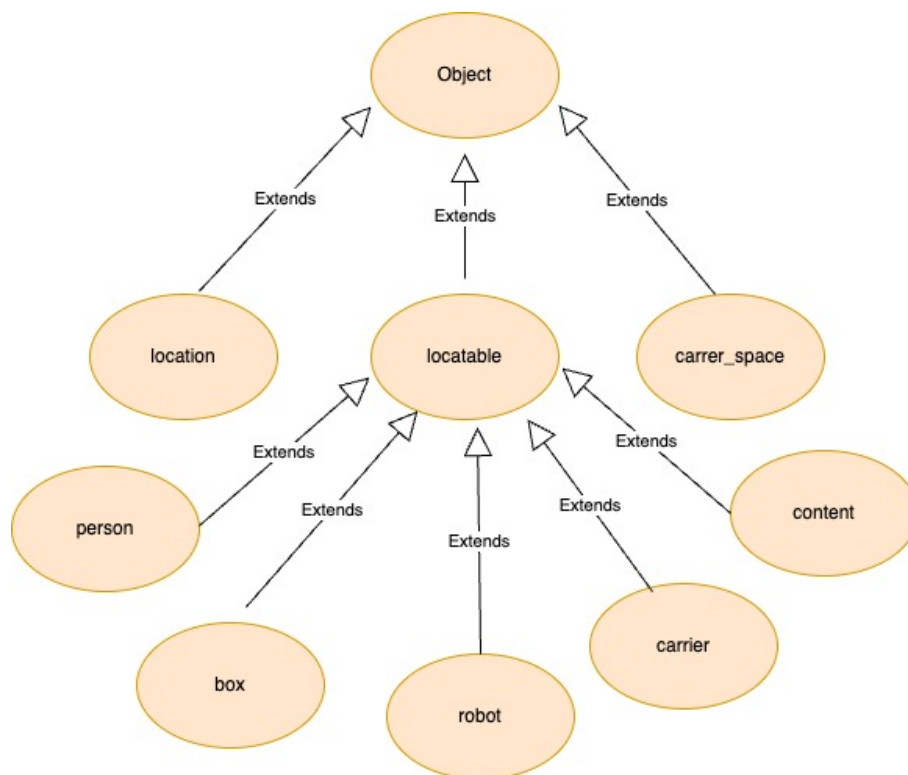


Figure 1.1: Gerarchia dei tipi

Nel diagramma UML mostrato in Figura 1.1, vengono mostrati i tipi e la gerarchia usate nel nostro dominio PDDL. E' stata utilizzata l'astrazione chiamata **locatable** in modo tale da raggruppare tutti gli oggetti che si possono posizionare in una **location** ovvero:

- **person**
- **box**
- **robot**
- **carrier**
- **content**

Il tipo **carrier-space**, rappresenta un posto all'interno di un carrello, inoltre il tipo **content** è un tipo generico usato per indicare un qualsiasi contenuto richiesto dalle persone.

- **Definizione dei Predicati :predicates**

Vengono definiti vari predicati che rappresentano le relazioni tra gli oggetti all'interno del dominio. Alcuni esempi includono:

- **at ?locatable - locatable ?loc - location:** Indica che un oggetto locatable si trova in una determinata location.
- **isEmpty ?box - box:** Indica che una scatola è vuota.
- **isFilledBy ?box - box ?content - content:** Indica che una scatola è riempita da un certo contenuto.
- **needs ?person - person ?content - content** Indica che una persona ha bisogno di un certo contenuto.
- **isFree ?carrier - carrier ?carrier-space - carrier-space:** Indica che un carrello ha lo spazio carrier-space libero.
- **isOccupiedBy ?carrier - carrier ?carrier-space - carrier-space ?box - box:** Indica che un carrello ha lo spazio carrier-space occupato dalla scatola.
- **are-same-location ?location1 - location ?location2 - location:** Indica che location1 e location2 coincidono.

- **Definizione delle Azioni :action:** Sono definite diverse azioni che i robot possono compiere all'interno del dominio. Ecco alcune di queste azioni:

- **fill-box:** Un robot può raccogliere il contenuto che si trova nella sua posizione e riporlo in una scatola vuota.
- **empty-box-and-deliver:** Un robot può svuotare il contenuto da una scatola e consegnarlo a una persona che ne ha bisogno nella sua posizione.
- **load-carrier:** Un robot può caricare una scatola su un carrello.
- **unload-carrier:** Un robot può scaricare una scatola da un carrello.
- **move-carrier:** Un robot può spostare un carrello da una posizione iniziale a una posizione finale.
- **simply-move:** Un robot può spostarsi da una posizione iniziale a una posizione finale senza carrello.

Queste azioni sono descritte con parametri, condizioni pre-esecuzione (**:precondition**) ed effetti post-esecuzione (**:effect**) che specificano le situazioni in cui un'azione può essere eseguita e come essa modifica lo stato del mondo.

1.2.5 Istanze del Problema

Qui di seguito verranno descritte le 3 istanze del problema:

- **Istanza 1:**
 - **Deliveries:** 4
 - **n° Slots:** 4
 - **n° Box:** 5

- **Istanza 2:**
 - Deliveries: 11
 - n° Slots: 2
 - n° Box: 3
- **Istanza 3:**
 - Deliveries: 17
 - n° Slots: 2
 - n° Box: 4

1.2.6 Istanza 1 del Problema

Listing 1.2: istanza1.pddl

```

1 (define (problem instance1)
2 (:domain ESL)
3 (:objects
4   depot l1 l2 - location
5   b1 b2 b3 b4 b5 - box
6   p1 p2 p3 - person
7   food medicine - content
8   robot - robot
9   carrier - carrier
10  space1 space2 space3 space4 - carrier_space
11 )
12
13 (:init (at b1 depot)(at b2 depot)(at b3 depot)(at b4
14   depot)(at b5 depot)
15   (at food depot)(at medicine depot)
16   (at p1 l1)(at p2 l1)(at p3 l2)
17   (at robot depot)
18   (at carrier depot)
19   (isEmpty b1)(isEmpty b2)(isEmpty b3)(isEmpty b4)(isEmpty
20     b5)
21   (needs p1 medicine)(needs p1 food)(needs p2
22     medicine)(needs p3 food)
23   (isFree carrier space1)(isFree carrier space2)(isFree
24     carrier space3)(isFree carrier space4)
25   (are_same_location depot depot)(are_same_location l1
26     l1)(are_same_location l2 l2)
27 )
28
29 (:goal (and (not(needs p1 medicine))(not(needs p1
30   food))(not(needs p2 medicine))(not(needs p3 food))))
31 )

```

- **Condizione Iniziale:** Inizialmente, tutte e cinque le box si trovano in un'unica posizione chiamata deposito. Tutti i contenuti da caricare nelle box si trovano inizialmente nel deposito. Tali contenuti sono sempre sufficienti per soddisfare le esigenze delle persone ferite.
- **Le persone che richiedono assistenza sono tre:** P1, P2 e P3; P1 e P2 si trovano nella stessa posizione; P1 ha bisogno di cibo e medicine, P2 ha bisogno di medicine, P3 ha bisogno di

cibo.

- **Un singolo agente robotico e un carrier si trovano nel deposito.**
- **Ciascun carrier può contenere fino a 4 box**
- **Obiettivo:** Tutte le persone ferite ricevano gli articoli di cui hanno bisogno.

1.2.7 Istanza 2 del Problema

Listing 1.3: istanza2.pddl

```

1  (define (problem instance2)
2  (:domain ESL)
3  (:objects
4    depot l1 l2 l3 l4 l5 - location
5    b1 b2 b3 - box
6    p1 p2 p3 p4 p5 p6 - person
7    food medicine tools foodORtools - content
8    r1 r2 - robot
9    c1 c2 - carrier
10   space1 space2 - carrier_space
11 )
12
13
14 (:init (at b1 depot)(at b2 depot)(at b3 depot)
15         (at food depot)(at medicine depot)(at tools depot)(at
16         foodORtools depot)
17         (at p1 l1)(at p2 l1)(at p3 l2)(at p4 l3)(at p5 l4)(at p6
18         l5)
19         (at r1 depot)(at r2 depot)
20         (at c1 depot)(at c2 depot)
21         (isEmpty b1)(isEmpty b2)(isEmpty b3)
22
23         (needs p1 foodORtools)
24         (needs p2 medicine)
25         (needs p3 medicine)
26         (needs p4 food)
27         (needs p4 medicine)
28         (needs p5 food)
29         (needs p5 medicine)
30         (needs p5 tools)
31         (needs p6 food)
32         (needs p6 medicine)
33         (needs p6 tools)
34
35         (isFree c1 space1)(isFree c1 space2)(isFree c2
36         space1)(isFree c2 space2)
37         (are_same_location depot depot)(are_same_location l1
38         l1)(are_same_location l2 l2)
39         (are_same_location l3 l3)(are_same_location l4
40         l4)(are_same_location l5 l5)

```

```

38      )
39
40  (:goal
41      (and
42          (not(needs p1 foodORtools))
43          (not(needs p2 medicine))
44          (not(needs p3 medicine))
45          (not(needs p4 food))
46          (not(needs p4 medicine))
47          (not(needs p5 food))
48          (not(needs p5 medicine))
49          (not(needs p5 tools))
50          (not(needs p6 food))
51          (not(needs p6 medicine))
52          (not(needs p6 tools))
53      )
54  )
55  )
56  )

```

- **Condizione Iniziale:**

- 2 Robot
- 2 Carrier
- 3 Box
- La capacità dei Carrier è di 2
- 6 People

p1 ha bisogno di cibo o attrezzi; p2 ha bisogno di medicine; P3 ha bisogno di medicine; P4 ha bisogno di medicine e cibo; P5 e P6 hanno bisogno di tutto. P1 e P2 si trovano nella stessa posizione; le altre persone sono in posizioni diverse.

- **Obiettivo:** Tutte le persone ferite ricevano gli articoli di cui hanno bisogno.

1.2.8 Istanza 3 del Problema

Listing 1.4: istanza3.pddl

```

1  (define (problem instance2)
2  (:domain ESL)
3  (:objects
4      depot 11 12 13 14 15 16 17 - location
5      b1 b2 b3 b4 - box
6      p1 p2 p3 p4 p5 p6 p7 p8 - person
7      food medicine tools foodORtools - content
8      r1 r2 - robot
9      c1 c2 - carrier
10     space1 space2 - carrier_space
11  )
12  )
13
14  (:init (at b1 depot)(at b2 depot)(at b3 depot)
15         (at food depot)(at medicine depot)(at tools depot)(at
16             foodORtools depot)

```

```

16      (at p1 l1)(at p2 l1)(at p3 l2)(at p4 l3)(at p5 l4)(at p6
17          15)(at p7 l6)(at p8 l7)
18      (at r1 depot)(at r2 depot)
19      (at c1 depot)(at c2 depot)
20      (isEmpty b1)(isEmpty b2)(isEmpty b3)
21
22      (needs p1 foodORtools)
23      (needs p2 medicine)
24      (needs p3 medicine)
25      (needs p4 food)
26      (needs p4 medicine)
27      (needs p5 food)
28      (needs p5 medicine)
29      (needs p5 tools)
30      (needs p6 food)
31      (needs p6 medicine)
32      (needs p6 tools)
33      (needs p7 food)
34      (needs p7 medicine)
35      (needs p7 tools)
36      (needs p8 food)
37      (needs p8 medicine)
38      (needs p8 tools)
39
40      (isFree c1 space1)(isFree c1 space2)(isFree c2
41          space1)(isFree c2 space2)
42
43      (are_same_location depot depot)(are_same_location l1
44          l1)(are_same_location l2 l2)
45      (are_same_location l3 l3)(are_same_location l4
46          l4)(are_same_location l5 l5)
47      (are_same_location l6 l6)(are_same_location l7 l7)
48
49      )
50
51      (:goal (and
52          (not(needs p1 foodORtools))
53          (not(needs p2 medicine))
54          (not(needs p3 medicine))
55          (not(needs p4 food))
56          (not(needs p4 medicine))
57          (not(needs p5 food))
58          (not(needs p5 medicine))
59          (not(needs p5 tools))
60          (not(needs p6 food))
61          (not(needs p6 medicine))
62          (not(needs p6 tools))
63          (not(needs p7 food))
64          (not(needs p7 medicine))

```

```

61         (not (needs p7 tools))
62         (not (needs p8 food))
63         (not (needs p8 medicine))
64         (not (needs p8 tools))
65     )
66 )
67 )

```

- **Condizione Iniziale:**

- 2 Robot
- 2 Carrier
- 4 Box
- La capacità dei Carrier è di 2
- 8 People

P1 ha bisogno di cibo o attrezzi; P2 ha bisogno di medicine; P3 ha bisogno di medicine; P4 ha bisogno di medicine e cibo; P5, P6, P7 e p8 hanno bisogno di tutto. P1 e P2 si trovano nella stessa posizione; le altre persone sono in posizioni diverse.

- **Obiettivo:** Tutte le persone ferite ricevano gli articoli di cui hanno bisogno.

1.2.9 Output delle tre istanze

Vorremmo porgere l'attenzione non al tempo di esecuzione (in quanto si tratta di un planner online orientato al *Temporal Planning*), ma alla qualità dei piani in termini di numero di azioni necessarie. Partendo dal dominio "completo" senza rilassamenti alcuni, si è ottenuto per le tre istanze, un numero di mosse mostrato nelle Figura 1.2, Figura 1.3, Figura 1.4. Come vedremo nel paragrafo successivo, usando una nostra euristica domain specific, siamo riusciti ad ottenere piani molto più soddisfacenti.

```

Plan computed:
Time: (ACTION) [action Duration; action Cost]
0.0003: (FILL_BOX DEPOT B3 FOOD ROBOT) [D:1.0000; C:1.0000]
1.0005: (LOAD_CARRIER DEPOT B3 CARRIER SPACE1 ROBOT) [D:1.0000; C:1.0000]
0.0008: (FILL_BOX DEPOT B4 MEDICINE ROBOT) [D:1.0000; C:1.0000]
1.0010: (LOAD_CARRIER DEPOT B4 CARRIER SPACE4 ROBOT) [D:1.0000; C:1.0000]
0.0012: (FILL_BOX DEPOT B2 FOOD ROBOT) [D:1.0000; C:1.0000]
1.0015: (LOAD_CARRIER DEPOT B2 CARRIER SPACE2 ROBOT) [D:1.0000; C:1.0000]
0.0018: (FILL_BOX DEPOT B5 MEDICINE ROBOT) [D:1.0000; C:1.0000]
1.0020: (LOAD_CARRIER DEPOT B5 CARRIER SPACE3 ROBOT) [D:1.0000; C:1.0000]
2.0022: (MOVE_CARRIER DEPOT L2 CARRIER ROBOT) [D:1.0000; C:1.0000]
3.0025: (UNLOAD_CARRIER L2 B2 CARRIER SPACE2 ROBOT) [D:1.0000; C:1.0000]
4.0027: (EMPTY_BOX_AND_DELIVER L2 B2 FOOD P3 ROBOT) [D:1.0000; C:1.0000]
5.0030: (MOVE_CARRIER L2 L1 CARRIER ROBOT) [D:1.0000; C:1.0000]
6.0033: (UNLOAD_CARRIER L1 B3 CARRIER SPACE1 ROBOT) [D:1.0000; C:1.0000]
7.0035: (EMPTY_BOX_AND_DELIVER L1 B3 FOOD P1 ROBOT) [D:1.0000; C:1.0000]
6.0037: (UNLOAD_CARRIER L1 B4 CARRIER SPACE4 ROBOT) [D:1.0000; C:1.0000]
7.0040: (EMPTY_BOX_AND_DELIVER L1 B4 MEDICINE P2 ROBOT) [D:1.0000; C:1.0000]
6.0043: (UNLOAD_CARRIER L1 B5 CARRIER SPACE3 ROBOT) [D:1.0000; C:1.0000]
7.0045: (EMPTY_BOX_AND_DELIVER L1 B5 MEDICINE P1 ROBOT) [D:1.0000; C:1.0000]

Solution found:
Total time: 0.51
Search time: 0.14
Actions: 18
Execution cost: 18.00
Duration: 9.000
Plan quality: 18.000
Plan file: plan/tmp/problem-264-E2wJNg9dJlJL-.pddl_1.SOL

```

Figure 1.2: Output Istanza 1

```
69.0240: (MOVE CARRIER L3 L4 C2 R1) [D:1.0000; C:1.0000]
70.0242: (UNLOAD CARRIER L4 B2 C1 SPACE2 R1) [D:1.0000; C:1.0000]
71.0245: (EMPTY_BOX_AND_DELIVER L4 B2 FOOD P5 R1) [D:1.0000; C:1.0000]

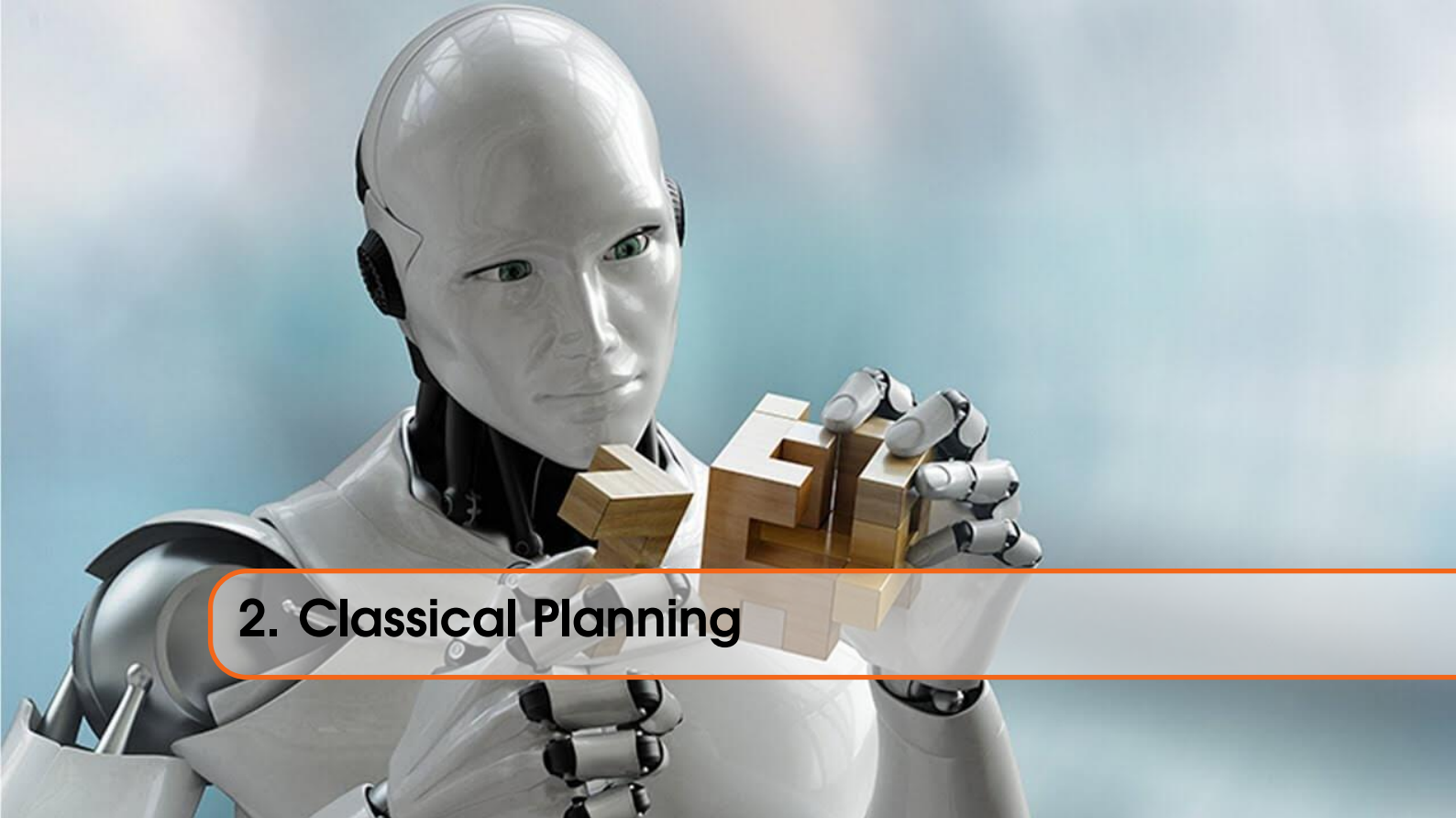
Solution found:
Total time:      1.04
Search time:     0.24
Actions:         98
Execution cost:  98.00
Duration:        72.000
Plan quality:    98.000
Plan file:       plan_/tmp/problem--264-61bmujyWxb16-.pddl_1.SOL
```

Figure 1.3: Output Istanza 2

```
128.0417: (MOVE CARRIER L1 L4 C1 R2) [D:1.0000; C:1.0000]
129.0420: (UNLOAD CARRIER L4 B2 C1 SPACE2 R2) [D:1.0000; C:1.0000]
127.0423: (SIMPLY MOVE DEPOT L4 R1) [D:1.0000; C:1.0000]
130.0425: (EMPTY_BOX_AND_DELIVER L4 B2 FOOD P5 R1) [D:1.0000; C:1.0000]

Solution found:
Total time:      1.93
Search time:     1.74
Actions:         158
Execution cost:  158.00
Duration:        124.000
Plan quality:    158.000
Plan file:       plan_/tmp_1.SOL
```

Figure 1.4: Output Istanza 3



2. Classical Planning

2.1 Obiettivi

Per rispondere al punto 2 della traccia, che richiedeva lo sviluppo di un planner personalizzato utilizzando le librerie fornite dal framework *PDDL4J*, abbiamo intrapreso un processo di personalizzazione dalla classe ASP discussa a lezione. Partendo da questa classe, abbiamo effettuato modifiche mirate per adattarla alle nostre esigenze. Il passo successivo è stato l'integrazione di un'euristica specifica per il nostro dominio, da noi ideata e sviluppata, con l'obiettivo di identificare soluzioni efficienti e, spesso, ottimali in tempi brevi.

La necessità di questa euristica domain-specific è emersa chiaramente dal confronto con il dominio da noi definito. Questo dominio, caratterizzato da un'elevata densità di azioni possibili e da una formalizzazione poco rilassata, presentava sfide notevoli, anche nell'istanza più semplice. Lo spazio degli stati da esplorare era ampio e complesso, rendendo l'approccio standard poco efficace. L'introduzione dell'euristica domain-specific ha permesso di guidare l'algoritmo in maniera più mirata e precisa verso l'obiettivo, sfruttando la conoscenza specifica del dominio per ottimizzare la ricerca di soluzioni. Questo approccio si è rivelato particolarmente fruttuoso, permettendo di superare i limiti delle tecniche più generali e di raggiungere risultati di alto livello in termini di efficienza e qualità delle soluzioni trovate.

La soluzione che abbiamo ottenuto è sostanzialmente basata su 3 classi:

2.2 Struttura delle Classi

- **myASP:** E' una versione leggermente modificata della classe ASP che implementa l'algoritmo A^* . Le modifiche che abbiamo effettuato alla classe sono necessarie a far funzionare al meglio l'euristica definita da noi.

- **myNode::** Questa classe costituisce il nucleo fondamentale della nostra soluzione, rappresentando una versione estesa della classe *Node* originariamente presente in PDDL. Essenzialmente, essa svolge la medesima funzione, ma aggiunge ulteriori funzionalità necessarie per i calcoli dell'euristica. Ad esempio mantiene le varie informazioni sullo stato corrispondente nell'albero di ricerca. Ad esempio questa classe tiene traccia del numero di consegne ancora da effettuare in uno stato specifico. Approfondiremo ulteriormente questa classe nel capitolo successivo.
- **myHeuristic** Questa classe è un'estensione della classe *RelaxedGraphEuristic* di PDDL4J. Essenzialmente, questa classe sfrutta le informazioni accumulate all'interno dell'oggetto *myNode* associato a uno stato specifico per calcolare una valutazione della distanza di tale stato dall'obiettivo. Inoltre, l'euristica può applicare *penalità* strategiche per eliminare interi rami dell'albero di ricerca, allo scopo di accelerare la ricerca o evitare di intraprendere percorsi eccessivamente lunghi o complessi che coinvolgerebbero più azioni del necessario. Ne approfondiremo ulteriormente il funzionamento nei prossimi capitoli.

2.2.1 myASP

La classe *MyASP* è l'implementazione di un planner di ricerca basato sull'algoritmo *A** per la risoluzione di problemi di pianificazione. La classe gestisce la configurazione del planner, consentendo la personalizzazione di parametri chiave come il peso dell'euristica. La classe fornisce diversi metodi per l'inizializzazione del planner, la gestione delle chiamate all'euristica che, nel nostro caso, è quella implementata da noi e la gestione di eventuali problemi nei file PDDL.

```

1  /*
2  /*
3  * Copyright (c) 2021 by Damien Pellier <Damien.Pellier@imag
4  * .fr>.
5  *
6  * This file is part of PDDL4J library.
7  *
8  * PDDL4J is free software: you can redistribute it and/or
9  * modify * it under the terms of the GNU General Public
10 * License
11 * as published by the Free Software Foundation, either
12 * version 3 of the License, or (at your option) any later
13 * version.
14 *
15 * PDDL4J is distributed in the hope that it will be useful,
16 * but WITHOUT ANY WARRANTY; without even the implied
17 * warranty
18 * of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
19 * See the GNU General Public License for more details.
20 *
21 * You should have received a copy of the GNU General Public
22 * License * along with PDDL4J. If not,
23 * see <http://www.gnu.org/licenses/>
24 */
25
26 package fr.uga.pddl4j.examples.asp;

```

```

19 import fr.uga.pddl4j.heuristics.state.StateHeuristic;
20 import fr.uga.pddl4j.parser.DefaultParsedProblem;
21 import fr.uga.pddl4j.parser.RequireKey;
22 import fr.uga.pddl4j.plan.Plan;
23 import fr.uga.pddl4j.plan.SequentialPlan;
24 import fr.uga.pddl4j.planners.LogLevel;
25 import fr.uga.pddl4j.planners.AbstractPlanner;
26 import fr.uga.pddl4j.planners.InvalidConfigurationException
    ;
27 import fr.uga.pddl4j.planners.Planner;
28 import fr.uga.pddl4j.planners.PlannerConfiguration;
29 import fr.uga.pddl4j.planners.ProblemNotSupportedException;
30 import fr.uga.pddl4j.planners.statespace.search.Node;
31 import fr.uga.pddl4j.problem.DefaultProblem;
32 import fr.uga.pddl4j.problem.Problem;
33 import fr.uga.pddl4j.problem.State;
34 import fr.uga.pddl4j.problem.operator.Action;
35 import fr.uga.pddl4j.problem.operator.ConditionalEffect;
36 import org.apache.logging.log4j.LogManager;
37 import org.apache.logging.log4j.Logger;
38 import picocli.CommandLine;
39
40 import java.util.Comparator;
41 import java.util.HashSet;
42 import java.util.List;
43 import java.util.PriorityQueue;
44 import java.util.Set;
45 /**
46  * The class is an example. It shows how to create a simple
47  * A* search planner able to
48  * solve an ADL problem by choosing the heuristic to used
49  * and its weight.
50  *
51  * @author D. Pellier
52  * @version 4.0 - 30.11.2021
53  */
54 @CommandLine.Command(name = "MyASP",
55     version = "MyASP 1.0",
56     description = "Solves a specified planning problem
57         using A* search strategy.",
58     sortOptions = false,
59     mixinStandardHelpOptions = true,
60     headerHeading = "Usage:%n",
61     synopsisHeading = "%n",
62     descriptionHeading = "%nDescription:%n%n",
63     parameterListHeading = "%nParameters:%n",
64     optionListHeading = "%nOptions:%n")
65 public class MyASP extends AbstractPlanner {

```

```
64     /**
65      * The class logger.
66      */
67     private static final Logger LOGGER = LogManager.
        getLogger(MyASP.class.getName());
68
69     /**
70      * The HEURISTIC property used for planner
71      * configuration.
72      */
73     public static final String HEURISTIC_SETTING = "
        HEURISTIC";
74
75     /**
76      * The default value of the HEURISTIC property used for
77      * planner configuration.
78      */
79     public static final StateHeuristic.Name
        DEFAULT_HEURISTIC = StateHeuristic.Name.MAX;
80
81     /**
82      * The WEIGHT_HEURISTIC property used for planner
83      * configuration.
84      */
85     public static final String WEIGHT_HEURISTIC_SETTING = "
        WEIGHT_HEURISTIC";
86
87     /**
88      * The default value of the WEIGHT_HEURISTIC property
89      * used for planner configuration.
90      */
91     public static final double DEFAULT_WEIGHT_HEURISTIC =
        1.0;
92
93     /**
94      * The weight of the heuristic.
95      */
96     private double heuristicWeight;
97
98     /**
99      * The name of the heuristic used by the planner.
100     */
101     private StateHeuristic.Name heuristic;
102
103     /**
104      * Creates a new A* search planner with the default
105      * configuration.
106      */
107     public MyASP() {
```

```

103         this(MyASP.getDefaultConfiguration());
104
105     }
106
107     /**
108      * Creates a new A* search planner with a specified
109      * configuration.
110      *
111      * @param configuration the configuration of the
112      * planner.
113      */
114     public MyASP(final PlannerConfiguration configuration)
115     {
116         super();
117         this.setConfiguration(configuration);
118     }
119
120     /**
121      * Sets the weight of the heuristic.
122      *
123      * @param weight the weight of the heuristic. The
124      * weight must be greater than 0.
125      * @throws IllegalArgumentException if the weight is
126      * strictly less than 0.
127      */
128     @CommandLine.Option(names = {"-w", "--weight"},
129         defaultValue = "1.0",
130         paramLabel = "<weight>", description = "Set the
131         weight of the heuristic (preset 1.0).")
132     public void setHeuristicWeight(final double weight) {
133         if (weight <= 0) {
134             throw new IllegalArgumentException("Weight <= 0
135             ");
136         }
137         this.heuristicWeight = weight;
138     }
139
140     /**
141      * Set the name of heuristic used by the planner to the
142      * solve a planning problem.
143      *
144      * @param heuristic the name of the heuristic.
145      */
146     @CommandLine.Option(names = {"-e", "--heuristic"},
147         defaultValue = "FAST_FORWARD",
148         description = "Set the heuristic : AJUSTED_SUM,
149         AJUSTED_SUM2, AJUSTED_SUM2M, COMBO, "
150         + "MAX, FAST_FORWARD SET_LEVEL, SUM, SUM_MUTEX
151         (preset: FAST_FORWARD)")

```

```

140     public void setHeuristic(StateHeuristic.Name heuristic)
141     {
142         this.heuristic = heuristic;
143     }
144
145     /**
146     * Returns the name of the heuristic used by the
147     * planner to solve a planning problem.
148     *
149     * @return the name of the heuristic used by the
150     * planner to solve a planning problem.
151     */
152     public final StateHeuristic.Name getHeuristic() {
153         return this.heuristic;
154     }
155
156     /**
157     * Returns the weight of the heuristic.
158     *
159     * @return the weight of the heuristic.
160     */
161     public final double getHeuristicWeight() {
162         return this.heuristicWeight;
163     }
164
165     /**
166     * Instantiates the planning problem from a parsed
167     * problem.
168     *
169     * @param problem the problem to instantiate.
170     * @return the instantiated planning problem or null if
171     * the problem cannot be instantiated.
172     */
173     @Override
174     public Problem instantiate(DefaultParsedProblem problem
175     ) {
176         final Problem pb = new DefaultProblem(problem);
177         pb.instantiate();
178         return pb;
179     }
180
181     /**
182     * Search a solution plan to a specified domain and
183     * problem using A*.
184     *
185     * @param problem the problem to solve.
186     * @return the plan found or null if no plan was found.
187     */

```

```

182  @Override
183  public Plan solve(final Problem problem) {
184      LOGGER.info("* Starting My A* seaaarch \n");
185      Plan plan = null;
186      // Search a solution
187      try {
188          final long begin = System.currentTimeMillis();
189
190          plan = this.astar(problem);
191          final long end = System.currentTimeMillis();
192          // If a plan is found update the statistics of
              the planner
193          // and log search information
194          if (plan != null) {
195              LOGGER.info("* My A* search succeeded\n");
196              this.getStatistics().setTimeToSearch(end -
                  begin);
197          } else {
198              LOGGER.info("* My A* search failed\n");
199          }
200
201      } catch (ProblemNotSupportedException e) {
202          LOGGER.error("not supported problem");
203          e.printStackTrace();
204      }
205      finally {
206          // Return the plan found or null if the search
              fails.
207          return plan;
208      }
209  }
210
211  /**
212   * Checks the planner configuration and returns if the
              configuration is valid.
213   * A configuration is valid if (1) the domain and the
              problem files exist and
214   * can be read, (2) the timeout is greater than 0, (3)
              the weight of the
215   * heuristic is greater than 0 and (4) the heuristic is
              a not null.
216   *
217   * @return <code>true</code> if the configuration is
              valid <code>false</code> otherwise.
218   */
219  public boolean isValidConfiguration() {
220      return super.isValidConfiguration()
221          && this.getHeuristicWeight() > 0.0
222          && this.getHeuristic() != null;

```

```
223     }
224
225     /**
226      * This method return the default arguments of the
227      * planner.
228      *
229      * @return the default arguments of the planner.
230      * @see PlannerConfiguration
231      */
232     public static PlannerConfiguration
233     getDefaultConfiguration() {
234         PlannerConfiguration config = Planner.
235             getDefaultConfiguration();
236         config.setProperty(MyASP.HEURISTIC_SETTING, MyASP.
237             DEFAULT_HEURISTIC.toString());
238         config.setProperty(MyASP.WEIGHT_HEURISTIC_SETTING,
239             Double.toString(MyASP.DEFAULT_WEIGHT_HEURISTIC)
240             );
241         return config;
242     }
243
244     /**
245      * Returns the configuration of the planner.
246      *
247      * @return the configuration of the planner.
248      */
249     @Override
250     public PlannerConfiguration getConfiguration() {
251         final PlannerConfiguration config = super.
252             getConfiguration();
253         config.setProperty(MyASP.HEURISTIC_SETTING, this.
254             getHeuristic().toString());
255         config.setProperty(MyASP.WEIGHT_HEURISTIC_SETTING,
256             Double.toString(this.getHeuristicWeight()));
257         return config;
258     }
259
260     /**
261      * Sets the configuration of the planner. If a planner
262      * setting is not defined in
263      *
264      * the specified configuration, the setting is
265      * initialized with its default value.
266      *
267      * @param configuration the configuration to set.
268      */
269     @Override
270     public void setConfiguration(final PlannerConfiguration
271         configuration) {
272         super.setConfiguration(configuration);
273     }
```



```

261         if (configuration.getProperty(MyASP.
262             WEIGHT_HEURISTIC_SETTING) == null) {
263             this.setHeuristicWeight(MyASP.
264                 DEFAULT_WEIGHT_HEURISTIC);
265         } else {
266             this.setHeuristicWeight(Double.parseDouble(
267                 configuration.getProperty(
268                     MyASP.WEIGHT_HEURISTIC_SETTING)));
269         }
270         if (configuration.getProperty(MyASP.
271             HEURISTIC_SETTING) == null) {
272             this.setHeuristic(MyASP.DEFAULT_HEURISTIC);
273         } else {
274             this.setHeuristic(StateHeuristic.Name.valueOf(
275                 configuration.getProperty(
276                     MyASP.HEURISTIC_SETTING)));
277         }
278     }
279
280     /**
281     * Search a solution plan for a planning problem using
282     * an A* search strategy.
283     *
284     * @param problem the problem to solve.
285     * @return a plan solution for the problem or null if
286     *         there is no solution
287     * @throws ProblemNotSupportedException if the problem
288     *         to solve is not supported by the planner.
289     */
290     public Plan astar(Problem problem) throws
291         ProblemNotSupportedException {
292         // Check if the problem is supported by the planner
293         if (!this.isSupported(problem)) {
294             throw new ProblemNotSupportedException("Problem
295                 not supported");
296         }
297
298         final MyHeuristic heuristic = new MyHeuristic(
299             problem);
300
301         // We get the initial state from the planning
302         // problem
303         final State init = new State(problem.
304             getInitialState());
305
306         // We initialize the closed list of nodes (store
307         // the nodes explored)

```

```

296     final Set<MyNode> close = new HashSet<>();
297
298     // We initialize the opened list to store the
299     // pending node according to function f
300     final double weight = this.getHeuristicWeight();
301     final PriorityQueue<MyNode> open = new
302     PriorityQueue<>(100, new Comparator<MyNode>() {
303         public int compare(MyNode n1, MyNode n2) {
304             double f1 = weight * n1.getHeuristic() + n1
305             .getCost();
306             double f2 = weight * n2.getHeuristic() + n2
307             .getCost();
308             return Double.compare(f1, f2);
309         }
310     });
311
312     // We create the root node of the tree search
313
314     final MyNode root = new MyNode(init, null, -1, 0,
315     heuristic.estimate(init, problem.getGoal())
316     ,17,2,4);
317
318     // We add the root to the list of pending nodes
319     open.add(root);
320     Plan plan = null;
321
322     // We set the timeout in ms allocated to the search
323     final int timeout = this.getTimeout() * 1000;
324     long time = 0;
325
326     // We start the search
327     while (!open.isEmpty() && plan == null && time <
328     timeout) {
329
330         // We pop the first node in the pending list
331         open
332         final MyNode current = open.poll();
333         close.add(current);
334
335         // If the goal is satisfied in the current node
336         // then extract the search and return it
337         if (current.satisfy(problem.getGoal())) {
338             return this.extractPlan(current, problem);
339         } else { // Else we try to apply the actions of
340             // the problem to the current node
341             for (int i = 0; i < problem.getActions().
342             size(); i++) {
343                 // We get the actions of the problem
344                 Action a = problem.getActions().get(i);

```

```

334         // If the action is applicable in the
335         current node
336         if (a.isApplicable(current)) { //
337             limitare azioni considerate
338
339             MyNode next = new MyNode(current,a)
340             ;
341             // We apply the effect of the
342             action
343             final List<ConditionalEffect>
344             effects = a.
345             getConditionalEffects();
346             for (ConditionalEffect ce : effects
347             ) {
348                 if (current.satisfy(ce.
349                 getCondition())) {
350                     next.apply(ce.getEffect());
351                 }
352             }
353
354             // We set the new child node
355             information
356             final double g = current.getCost()
357             + 1;
358             if (!close.contains(next)) {
359                 next.setCost(g);
360                 next.setParent(current);
361                 next.setAction(i);
362                 next.setHeuristic(heuristic.
363                 estimate(next, problem.
364                 getGoal()));
365                 open.add(next);
366             }
367         }
368     }
369 }
370
371 // Finally, we return the search computed or null
372 // if no search was found
373 return plan;
374 }
375
376 /**
377  * Extracts a search from a specified node.
378  *
379  * @param node    the node.
380  * @param problem the problem.

```

```

370     * @return the search extracted from the specified node
371     */
372     private Plan extractPlan(final MyNode node, final
373         Problem problem) {
374         MyNode n = node;
375         final Plan plan = new SequentialPlan();
376         while (n.getAction() != -1) {
377             final Action a = problem.getActions().get(n.
378                 getAction());
379             plan.add(0, a);
380             n = n.getParent();
381         }
382         return plan;
383     }
384
385     /**
386     * Returns if a specified problem is supported by the
387     * planner. Just ADL problem can be solved by this
388     * planner.
389     *
390     * @param problem the problem to test.
391     * @return <code>true</code> if the problem is
392     *         supported <code>false</code> otherwise.
393     */
394     @Override
395     public boolean isSupported(Problem problem) {
396         return (problem.getRequirements().contains(
397             RequireKey.ACTION_COSTS)
398             || problem.getRequirements().contains(
399                 RequireKey.CONSTRAINTS)
400             || problem.getRequirements().contains(
401                 RequireKey.CONTINUOUS_EFFECTS)
402             || problem.getRequirements().contains(
403                 RequireKey.DERIVED_PREDICATES)
404             || problem.getRequirements().contains(
405                 RequireKey.DURATIVE_ACTIONS)
406             || problem.getRequirements().contains(
407                 RequireKey.DURATION_INEQUALITIES)
408             || problem.getRequirements().contains(
409                 RequireKey.FLUENTS)
410             || problem.getRequirements().contains(
411                 RequireKey.GOAL_UTILITIES)
412             || problem.getRequirements().contains(
413                 RequireKey.METHOD_CONSTRAINTS)
414             || problem.getRequirements().contains(
415                 RequireKey.NUMERIC_FLUENTS)
416             || problem.getRequirements().contains(
417                 RequireKey.OBJECT_FLUENTS)

```

```

402         || problem.getRequirements().contains(
           RequireKey.PREFERENCES)
403         || problem.getRequirements().contains(
           RequireKey.TIMED_INITIAL_LITERALS)
404         || problem.getRequirements().contains(
           RequireKey.HIERARCHY))
405         ? false
406         : true;
407     }
408
409     public static void main(String[] args) {
410         // The path to the benchmarks directory
411         final String benchmarks = "/home/oem/ASP/filePDDL/";
412         // Gets the default configuration from the planner
413         fr.uga.pddl4j.planners.PlannerConfiguration config =
           MyASP.getDefaultConfiguration();
414         // Sets the domain of the problem to solve
415         config.setProperty(MyASP.DOMAIN_SETTING, benchmarks
           + "domain.pddl");
416         // Sets the problem to solve
417         config.setProperty(MyASP.PROBLEM_SETTING, benchmarks
           + "instance3.pddl");
418         // Sets the timeout allocated to the search.
419         config.setProperty(MyASP.TIME_OUT_SETTING, 1000);
420         // Sets the log level
421         config.setProperty(MyASP.LOG_LEVEL_SETTING, LogLevel
           .INFO);
422         // Sets the weight of the heuristic
423         config.setProperty(MyASP.WEIGHT_HEURISTIC_SETTING
           ,3.4);
424         //instance 1 :3.4
425         // Creates an instance of the MyASP planner with the
           specified configuration
426         final MyASP planner = new MyASP(config);
427         // Runs the planner and print the solution
428         try {
429             planner.solve();
430         } catch (InvalidConfigurationException e) {
431             e.printStackTrace();
432         } }

```

Listing 2.1: MyASP

- **solve(final Problem problem):** Questo metodo ha come unico obiettivo quello di chiamare il metodo `astar` che implementa la logica dell'algoritmo A*, gestire eventuali eccezioni dovute ad errori presenti nei file PDDL e/o errori legati al non ritrovamento di un piano, ed infine restituire il piano se trovato. Conta anche il tempo impiegato per il ritrovamento della soluzione.

- **astar(final Problem problem):** Questo metodo implementa la logica vera e propria di A* per cercare una soluzione al problema di pianificazione specificato. Si avvale di un'euristica per guidare la ricerca attraverso lo spazio degli stati, cercando una sequenza di azioni che conduca dalla situazione iniziale al goal. Nel nostro caso l'euristica utilizzata è proprio *myHeuristic*, definita da noi.
- **isSupported(Problem problem):** Verifica se il problema di pianificazione fornito come argomento è supportato dal planner. In base ai requisiti del problema, come la presenza di azioni continue o predicati derivati, decide se il planner può affrontare la pianificazione.
- **extractPlan(final MyNode node, final Problem problem):** Estrae il piano dalla soluzione trovata rappresentata da un nodo nella ricerca A*. Il metodo risale il percorso dall'obiettivo alla situazione iniziale, registrando le azioni necessarie nel piano risultante.
- **getDefaultConfiguration():** Restituisce la configurazione predefinita del planner, che può essere utilizzata come punto di partenza per personalizzare i parametri di esecuzione.
- **main(String[] args):** Rappresenta il main del nostro planner. In questo metodo, vengono impostati alcuni parametri di configurazione predefiniti, e il planner viene eseguito per risolvere una delle istanze del nostro problema, quindi vengono stampati i risultati e le tempistiche.

2.2.2 MyNode e MyHeuristic

Entriamo ora nel cuore del nostro planner. La nostra euristica è specifica per il dominio, il che significa che è cruciale che essa abbia accesso a informazioni riguardanti lo stato attuale del nostro scenario e le azioni precedentemente eseguite. La nostra euristica, come vedremo, si basa principalmente sulla seconda idea: cerca di stimare la distanza dall'obiettivo considerando quante azioni di ogni tipo sono ancora necessarie per raggiungerlo. Ora procediamo a illustrare la logica sottostante a questa euristica.

Sia N il numero di consegne da effettuare, S il numero di slot disponibili per il carrello e B il totale dei box a disposizione.

```

1 package fr.uga.pddl4j.examples.asp;
2
3 import fr.uga.pddl4j.heuristics.state.RelaxedGraphHeuristic;
4 import fr.uga.pddl4j.planners.statespace.search.Node;
5 import fr.uga.pddl4j.problem.Problem;
6 import fr.uga.pddl4j.problem.State;
7 import fr.uga.pddl4j.problem.numeric.NumericConstraint;
8 import fr.uga.pddl4j.problem.operator.Action;
9 import fr.uga.pddl4j.problem.operator.Condition;
10 import fr.uga.pddl4j.problem.operator.ConditionalEffect;
11
12 import java.util.List;
13
14 public final class MyHeuristic extends RelaxedGraphHeuristic
15 {
16     public MyHeuristic(Problem problem) {
17         super(problem);
18         super.setAdmissible(true);
19     }

```

```

20     public int estimate(State state, Condition goal) {
21         super.setGoal(goal);
22         super.expandRelaxedPlanningGraph(state);
23         return super.isGoalReachable() ? super.
            getRelaxedPlanValue() : Integer.MAX_VALUE;
24     }
25
26
27     public double estimate(Node node, Condition goal) {
28         return (double) this.estimate((State) node, goal);
29     }
30
31
32     public double estimate(MyNode node, Condition goal){
33         super.setGoal(goal);
34         super.expandRelaxedPlanningGraph(node);
35
36         double h=0;
37
38         if(node.getMin_fill_box_to_do() >=0) h+=node.
            getMin_fill_box_to_do()*0.5;
39         else h+=1000;
40
41         if(node.getMin_load_carrier_to_do() >=0) h+=node.
            getMin_load_carrier_to_do()*0.3;
42         else h+=1000;
43
44         if(node.getMin_back_load_carrier_to_do() >=0) h+=node.
            getMin_back_load_carrier_to_do()*0.2;
45         else h+=1000;
46
47         if(node.getMin_move_carrier_to_do() >=0) h+=node.
            getMin_move_carrier_to_do()*0.2;
48         else h+=(-node.getMin_move_carrier_to_do())*0.2;
49
50         if(node.getMin_back_move_carrier_to_do() >=0) h+=node.
            getMin_back_move_carrier_to_do()*0.1;
51         else h+=(-node.getMin_back_move_carrier_to_do())
            *0.1;
52
53         if(node.getMin_unload_carrier_to_do() >=0) h+=node.
            getMin_unload_carrier_to_do()*0.2; //per instance
            1 mettere a 0
54         else h+=1000;
55
56         if(node.getMin_back_unload_carrier_to_do() >=0) h+=
            node.getMin_back_unload_carrier_to_do()*0.3;
57         else h+=1000;
58         if(node.getMinSimply_move_to_do() >=0) h+=node.

```

```

    getMinSimply_move_to_do()*0;
59     else h+=(-node.getMinSimply_move_to_do())*0.8;
60
61     if(node.load_unload())h+=1000;
62     //penalizza il fare unload subito dopo una load(
        un comportamento che non ha senso)
63     //dopo che ho fatto una load dovrei muovermi da
        altre parti prima di scaricare
64     if(node.getMin_fill_box_to_do()>node.
        getMin_load_carrier_to_do())h+=1000;
65     //il numero di load che faccio deve sempre stare
        sopra al numero di fill
66     //altrimenti vuol dire che avr  caricato qualche
        scatola vuota dal deposito
67     //ma dal deposito non  mai utile caricare scatole
        vuote
68
69     if(node.getMin_back_load_carrier_to_do()<node.
        getDeliveries_to_do()- node.getN_boxes())h+=1000;
70     //if(node.getMin_back_unload_carrier_to_do()<node.
        getMin_back_load_carrier_to_do())h+=100;
71
72     h=h + node.getDeliveries_to_do()*4;
73
74     if(node.getDeliveries_to_do()==0)h= 0;
75
76     //System.out.println(h+" deliveries: "+node.
        getDeliveries_to_do()+" fill: "+node.
        getMin_fill_box_to_do()+" load: "+node.
        getMin_load_carrier_to_do()+" move: "+node.
        getMin_move_carrier_to_do()+
77     //      " unload: "+node.
        getMin_unload_carrier_to_do()+" back load: "+node
        .getMin_back_load_carrier_to_do()+" back unload:
        "+node.getMin_back_unload_carrier_to_do()
78     //      +" back move: "+node.
        getMin_back_move_carrier_to_do()+" simply move:
        "+node.getMinSimply_move_to_do());
79     return super.isGoalReachable() ? h : Integer.
        MAX_VALUE; }}

```

Listing 2.2: MyHeuristic

```

1
2 package fr.uga.pddl4j.examples.asp;
3 import fr.uga.pddl4j.problem.State;
4 import fr.uga.pddl4j.problem.operator.Action;
5
6 import java.util.HashMap;
7 import java.util.HashSet;

```



```
8
9 public class MyNode extends State{
10
11     private int min_fill_box_to_do;
12
13     private int min_load_carrier_to_do;
14     private int min_back_load_carrier_to_do;
15
16     private int min_move_carrier_to_do;
17     private int min_back_move_carrier_to_do;
18
19     private int min_unload_carrier_to_do;
20     private int min_back_unload_carrier_to_do;
21
22     private int min_simply_move_to_do;
23
24     private int deliveries_to_do;
25
26     private boolean load_da_poco;
27
28     private boolean load_unload;
29
30     private HashSet<Integer> filled_boxes= new HashSet<>();
31
32     private MyNode parent;
33     private int action;
34     private double cost;
35     private double heuristic;
36     private int depth;
37
38     private int n_boxes ;
39
40     public MyNode(MyNode n, Action a) {
41         super(n);
42
43         this.min_fill_box_to_do=n.min_fill_box_to_do;
44
45         this.min_load_carrier_to_do=n.min_load_carrier_to_do
46         ;
47         this.min_back_load_carrier_to_do=n.
48         min_back_load_carrier_to_do;
49
50         this.min_move_carrier_to_do=n.min_move_carrier_to_do
51         ;
52         this.min_back_move_carrier_to_do=n.
53         min_back_move_carrier_to_do;
54
55         this.min_unload_carrier_to_do=n.
56         min_unload_carrier_to_do;
```

```

52         this.min_back_unload_carrier_to_do=n.
           min_back_unload_carrier_to_do;
53
54         this.min_simply_move_to_do=n.min_simply_move_to_do;
55
56         this.deliveries_to_do=n.deliveries_to_do;
57
58         this.n_boxes=n.n_boxes;
59
60         this.load_da_poco=n.load_da_poco;
61         load_unload=false;
62
63         this.filled_boxes=n.filled_boxes;
64
65         if (a.getName().equals("fill_box")){
66             load_da_poco=false;
67             min_fill_box_to_do--;
68             filled_boxes.add(a.getValueOfParameter(1));
69         }
70         else if (a.getName().equals("load_carrier")){
71             load_da_poco=true; //non ha senso fare unload
              subito dopo
72
73             if(a.getValueOfParameter(0)==0 && filled_boxes.
              contains(a.getValueOfParameter(1))){
74                 //System.out.println(a.getValueOfParameter
              (1)+" "+filled_boxes);
75                 min_load_carrier_to_do--;
76             }
77             else if (a.getValueOfParameter(0)!=0 && !
              filled_boxes.contains(a.getValueOfParameter
              (1))){
78                 min_back_load_carrier_to_do--;
79             }
80         }
81     }
82     else if (a.getName().equals("move_carrier")){
83         load_da_poco=false;
84         if(a.getValueOfParameter(1)!=0)
85             min_move_carrier_to_do--;
86         else if (a.getValueOfParameter(1)==0)
87             min_back_move_carrier_to_do--;
88     }
89     else if (a.getName().equals("unload_carrier")){
90         if (load_da_poco)load_unload=true;
91         load_da_poco=false;
92         if(a.getValueOfParameter(0)==0 && !filled_boxes.
93             contains(a.getValueOfParameter(1)))
94             min_back_unload_carrier_to_do--;

```

```

91         else if (a.getValueOfParameter(0) != 0 &&
92                 filled_boxes.contains(a.getValueOfParameter
93                 (1))) min_unload_carrier_to_do--;
94     }
95     else if (a.getName().equals("simply_move")){
96         load_da_poco=false;
97         min_simply_move_to_do--;
98     }
99     else if (a.getName().equals("empty_box_and_deliver")
100             &&deliveries_to_do>0){
101         load_da_poco=false;
102         deliveries_to_do--;
103         filled_boxes.remove(a.getValueOfParameter(1));
104
105         //min_fill_box_to_do=deliveries_to_do;
106
107         //min_load_carrier_to_do=deliveries_to_do;
108         //this.min_back_load_carrier_to_do=
109         deliveries_to_do-2;
110
111         //min_move_carrier_to_do=deliveries_to_do/2;
112         //this.min_back_move_carrier_to_do=(
113         deliveries_to_do-2)/2;
114
115         //min_unload_carrier_to_do=deliveries_to_do;
116         //this.min_back_unload_carrier_to_do=
117         deliveries_to_do-2;
118
119         //min_simply_move_to_do=0;
120     }
121 }
122
123 public MyNode(State state, MyNode parent, int action,
124 double cost, double heuristic, int deliveries_to_do,
125 int num_slots, int num_boxes) {
126     super(state);
127     this.parent = parent;
128     this.action = action;
129     this.cost = cost;
130     this.heuristic = heuristic;
131     this.depth = -1;
132
133     if(parent==null){
134
135         this.min_fill_box_to_do=deliveries_to_do;
136
137         this.min_load_carrier_to_do=deliveries_to_do;
138         this.min_back_load_carrier_to_do=

```

```

        deliveries_to_do=num_boxes+1;
132
        this.min_move_carrier_to_do=(deliveries_to_do)/
133        num_slots;
134        this.min_back_move_carrier_to_do=(
        deliveries_to_do-num_boxes+1)/num_slots;
135
136        this.min_unload_carrier_to_do=deliveries_to_do;
137        this.min_back_unload_carrier_to_do=
        deliveries_to_do-num_boxes+1;
138
139        this.deliveries_to_do=deliveries_to_do;
140        this.min_simply_move_to_do=0;
141
142        this.n_boxes=num_boxes;
143
144    }
145}
146public MyNode(State state, MyNode parent, int action,
        double cost, int depth, double heuristic, int
        deliveries_to_do, int num_slots, int num_boxes) {
147    super(state);
148    this.parent = parent;
149    this.action = action;
150    this.cost = cost;
151    this.depth = depth;
152    this.heuristic = heuristic;
153
154    if(parent==null){
155        this.min_fill_box_to_do=deliveries_to_do;
156
157        this.min_load_carrier_to_do=deliveries_to_do;
158        this.min_back_load_carrier_to_do=
        deliveries_to_do-num_boxes;
159
160        this.min_move_carrier_to_do=deliveries_to_do/
        num_slots;
161        this.min_back_move_carrier_to_do=(
        deliveries_to_do-num_boxes)/num_slots;
162
163        this.min_unload_carrier_to_do=deliveries_to_do;
164        this.min_back_unload_carrier_to_do=
        deliveries_to_do-num_boxes;
165
166        this.deliveries_to_do=deliveries_to_do;
167        this.min_simply_move_to_do=0;
168
169        this.n_boxes=num_boxes;
170    }

```

```
171     }
172     public int getMin_fill_box_to_do() {
173         return min_fill_box_to_do;
174     }
175
176     public int getMin_load_carrier_to_do() {
177         return min_load_carrier_to_do;
178     }
179
180     public int getMin_move_carrier_to_do() {
181         return min_move_carrier_to_do;
182     }
183
184     public int getMin_unload_carrier_to_do() {
185         return min_unload_carrier_to_do;
186     }
187
188     public int getMinSimply_move_to_do() {
189         return min_simply_move_to_do;
190     }
191
192     public int getDeliveries_to_do() {
193         return deliveries_to_do;
194     }
195
196     public int getMin_back_load_carrier_to_do() {
197         return min_back_load_carrier_to_do;
198     }
199
200     public int getMin_back_move_carrier_to_do() {
201         return min_back_move_carrier_to_do;
202     }
203
204     public int getMin_back_unload_carrier_to_do() {
205         return min_back_unload_carrier_to_do;
206     }
207
208     public boolean load_unload(){
209         return load_unload;
210     }
211
212     public final int getAction() {
213         return this.action;
214     }
215
216     public final void setAction(int action) {
217         this.action = action;
218     }
219
```

```
220     public final MyNode getParent() {
221         return this.parent;
222     }
223
224     public final void setParent(MyNode parent) {
225         this.parent = parent;
226     }
227
228     public final double getCost() {
229         return this.cost;
230     }
231
232     public final void setCost(double cost) {
233         this.cost = cost;
234     }
235
236     public final double getHeuristic() {
237         return this.heuristic;
238     }
239
240     public final void setHeuristic(double estimates) {
241         this.heuristic = estimates;
242     }
243
244     public int getDepth() {
245         return this.depth;
246     }
247
248     public void setDepth(int depth) {
249         this.depth = depth;
250     }
251
252     public final double getValueF(double weight) {
253         return weight * this.heuristic + this.cost;
254     }
255     public boolean equals(Object obj) {
256         return super.equals(obj);
257     }
258     public int hashCode() {
259         return super.hashCode();
260     }
261     public int getN_boxes() {
262         return n_boxes;
263     }}
```

Listing 2.3: MyNode

In particolare, l'idea base è che all'inizio dello stato siano necessarie:

- **n deliveries** tante quante sono le clausole in *AND* nel goal dell'istanza, ovvero pari al numero di richieste dei clienti da soddisfare.

- almeno **n fill-box** fatte nel deposito: dobbiamo effettuare almeno una *fill box* per ogni articolo da consegnare. Tuttavia, eseguire più *fill box* di quelle necessarie non solo sarebbe superfluo dal punto di vista del piano, ma potrebbe anche risultare controproducente in termini di ricerca, poiché potrebbe comportare una perdita di tempo nella ricerca di una soluzione per via della creazione di loop.
- almeno **n load-carrier** di box situate nel deposito: anche qui bisogna farne almeno una per ogni contenuto da consegnare e farne di più sarebbe inutile e controproducente.
- almeno **n/s move-carrier** per trasportare le box riempite nei depositi alle locazioni dove si trovano le persone.
- almeno **n unload-carrier** in locazioni diverse dal deposito per scaricare scatole piene di cui consegnare il contenuto.
- almeno **n-b back-load-carrier** per ricaricare sul carretto le scatole vuote strettamente necessarie ad accontentare le richieste rimanenti delle persone. Se n è minore di b , il numero di back-load da fare va considerato nullo e quindi non è necessario ricaricare nessuna scatola.
- almeno **(n-b)/s back-move-carrier** dalle persone al deposito per riportare al deposito le scatole vuote da riutilizzare. Se n è minore di b , il numero di back-move-carrier da fare va considerato nullo e quindi non è necessario riportare nessuna scatola al deposito.
- almeno **n-b back-unload-carrier** nel deposito per scaricare le scatole vuote da riempire nuovamente. Se n è minore di b , il numero di back-unload-carrier da fare va considerato nullo e quindi non è necessario scaricare nessuna scatola nel deposito.

Tutte queste informazioni vengono memorizzate nelle corrispondenti variabili di *myNode*. Durante l'esplorazione dell'albero di ricerca, quando si passa da un nodo padre a un nodo figlio eseguendo una specifica azione, nel nodo figlio verrà decrementata solo la variabile associata a quella particolare azione, mentre le altre variabili rimarranno invariate. Per esempio, se l'azione che sto eseguendo è una *fill-box*, si procederà a decrementare la variabile *minNum-fill-box* che tiene traccia del numero minimo di *fill-box* richieste per raggiungere l'obiettivo. Le altre variabili rimarranno invariate.

L'euristica, a questo punto, è calcolata come la somma pesata dei valori contenuti in tutte queste variabili, che rappresentano il numero minimo di azioni necessarie per raggiungere l'obiettivo. Dopo averla testata su diversi esempi, abbiamo visto che questa funzione si comporta efficacemente come una buona misura di distanza dal goal.

In aggiunta a quanto sopra descritto, nell'euristica sono inclusi ulteriori termini che mirano a eliminare interi rami dell'albero di ricerca, allo scopo di guidare il planner verso una soluzione ottimale o altamente efficiente il più rapidamente possibile. Questi termini aggiuntivi sono progettati per ridurre i tempi di esecuzione complessivi del planner. Ad esempio andiamo a penalizzare:

- le **load** fatte subito dopo aver fatto una **unload** (che sono a tutti gli effetti mosse inutili)
- i piani in cui si fanno più **back-load** rispetto alle consegne effettuate. Infatti si dovrebbe fare una operazione di **back load** solo dopo che si è consegnato qualcosa in modo tale da prendere una scatola vuota. In altre parole il numero di *deliveries* deve essere sempre un *lower bound* del numero di *back-load* che restano da fare.
- allo stesso modo il numero di **fill box** deve essere sempre maggiore o uguale al numero di **load box**, altrimenti si andrebbe a caricare scatole dal deposito non ancora riempite, che si traducono in mosse inutili da effettuare.

L'unione di questa funzione distanza che abbiamo ottenuto e delle diverse penalità per velocizzare il tutto, ci consente di ottenere soluzioni ottime o simil-ottime in tutte e tre le istanze nel minor tempo possibile.

2.3 Risultati euristica sulle tre istanze

```
problem instantiation done successfully (162 actions, 72 fluents)

* Starting My A* seaaarch
* My A* search succeeded

found plan as follows:

00: (      fill_box depot b1 food robot) [0]
01: (      fill_box depot b2 food robot) [0]
02: (      fill_box depot b5 medicine robot) [0]
03: (      fill_box depot b4 medicine robot) [0]
04: ( load_carrier depot b1 carrier space3 robot) [0]
05: ( load_carrier depot b2 carrier space2 robot) [0]
06: ( load_carrier depot b4 carrier space1 robot) [0]
07: ( load_carrier depot b5 carrier space4 robot) [0]
08: (      move_carrier depot l1 carrier robot) [0]
09: ( unload_carrier l1 b4 carrier space1 robot) [0]
10: (empty_box_and_deliver l1 b4 medicine p1 robot) [0]
11: ( unload_carrier l1 b1 carrier space3 robot) [0]
12: ( empty_box_and_deliver l1 b1 food p1 robot) [0]
13: ( unload_carrier l1 b5 carrier space4 robot) [0]
14: (empty_box_and_deliver l1 b5 medicine p2 robot) [0]
15: (      move_carrier l1 l2 carrier robot) [0]
16: ( unload_carrier l2 b2 carrier space2 robot) [0]
17: ( empty_box_and_deliver l2 b2 food p3 robot) [0]

time spent:      0,04 seconds parsing
                 0,41 seconds encoding
                 3,32 seconds searching
                 3,76 seconds total time

memory used:      0,90 MBytes for problem representation
                  0,00 MBytes for searching
                  0,90 MBytes total
```

Figure 2.1: Risultati euristica su istanza 1

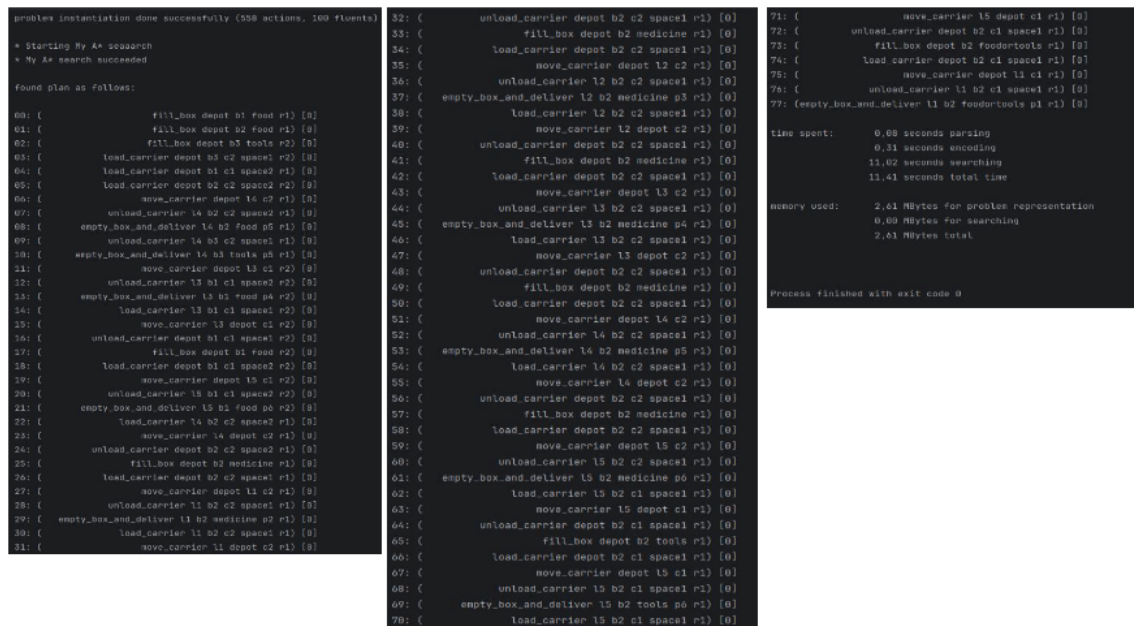


Figure 2.2: Risultati euristica su istanza 2

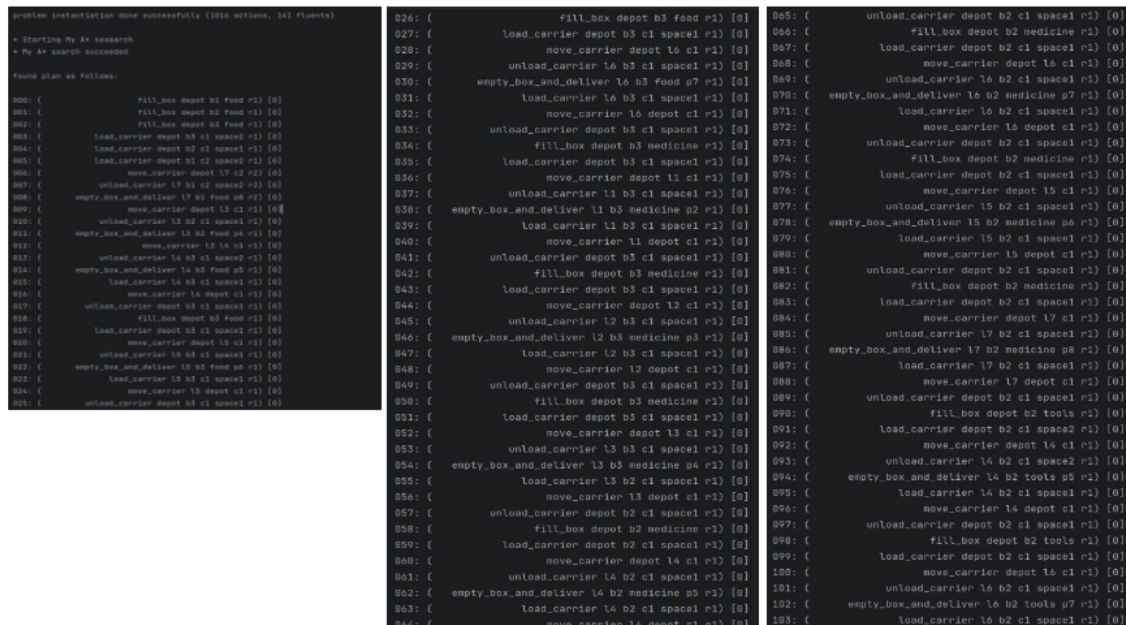


Figure 2.3: Risultati euristica su istanza 3

```

104: (      move_carrier l6 depot c1 r1) [0]
105: (      unload_carrier depot b2 c1 space1 r1) [0]
106: (      fill_box depot b2 tools r1) [0]
107: (      load_carrier depot b2 c1 space1 r1) [0]
108: (      move_carrier depot l5 c1 r1) [0]
109: (      unload_carrier l5 b2 c1 space1 r1) [0]
110: (      empty_box_and_deliver l5 b2 tools p6 r1) [0]
111: (      load_carrier l5 b2 c1 space1 r1) [0]
112: (      move_carrier l5 depot c1 r1) [0]
113: (      unload_carrier depot b2 c1 space1 r1) [0]
114: (      fill_box depot b2 tools r1) [0]
115: (      load_carrier depot b2 c1 space1 r1) [0]
116: (      move_carrier depot l7 c1 r1) [0]
117: (      unload_carrier l7 b2 c1 space1 r1) [0]
118: (      empty_box_and_deliver l7 b2 tools p8 r1) [0]
119: (      load_carrier l7 b2 c1 space1 r1) [0]
120: (      move_carrier l7 depot c1 r1) [0]
121: (      unload_carrier depot b2 c1 space1 r1) [0]
122: (      fill_box depot b2 foodortools r1) [0]
123: (      load_carrier depot b2 c1 space1 r1) [0]
124: (      move_carrier depot l1 c1 r1) [0]
125: (      unload_carrier l1 b2 c1 space1 r1) [0]
126: (empty_box_and_deliver l1 b2 foodortools p1 r1) [0]

time spent:      0,05 seconds parsing
                0,48 seconds encoding
                30,78 seconds searching
                39,31 seconds total time

memory used:     4,65 MBytes for problem representation
                0,00 MBytes for searching
                4,65 MBytes total

Process finished with exit code 0

```

Figure 2.4: Risultati euristica su istanza 3



3. Temporal Planning and Robotic

3.1 Temporal Planning

In questa fase viene inclusa nell'istanza 1 del problema, la gestione del tempo, utilizzando la versione 2.1 del linguaggio PDDL. Questo coinvolge l'introduzione delle **durative actions**, che consentono di rappresentare azioni che hanno una durata nel tempo, e dei **fluents**, che sono variabili numeriche che possono variare nel tempo.

Le **durative actions** sono un'estensione del linguaggio PDDL che permette di rappresentare azioni che richiedono un certo periodo di tempo per essere completate. Questo è particolarmente utile per modellare scenari in cui le azioni hanno una durata. Ad esempio, un'azione potrebbe richiedere un certo tempo per spostarsi da una posizione all'altra o per eseguire una determinata operazione.

I **fluents** sono variabili numeriche che possono cambiare nel tempo. Questo è importante perché consente di rappresentare concetti che variano continuamente, come la quantità di risorse disponibili, la posizione dei robot o il tempo trascorso. I fluents sono utili per tenere traccia di informazioni dinamiche nel dominio del problema, consentendo di effettuare pianificazioni basate su dati numerici che cambiano nel corso del tempo.

L'aggiunta del supporto per le durative actions e i fluents rende il modello PDDL più potente e flessibile, in quanto può ora gestire scenari in cui il tempo e le variabili numeriche giocano un ruolo cruciale nella pianificazione delle azioni. Questa estensione consente di affrontare problemi più complessi e realistici che coinvolgono il tempo come un fattore chiave nella soluzione del problema.

Listing 3.1: domainTemporal.pddl

```
1
2 (define
3   (domain ESLD)
4
5   (:requirements
6     :strips :typing
```

```

7      :negative-preconditions :universal-preconditions
8      :durative-actions
9      :fluents
10     )
11
12
13  (:types
14     location locatable carrier_space - object
15     person box robot carrier content - locatable
16  )
17
18  (:predicates
19     ;todo: define predicates here
20     (at ?locatable - locatable ?loc - location)
21
22     (is_empty ?box -box)
23     (is_filled_by ?box - box ?content - content)
24
25     (needs ?person - person ?content - content)
26
27     (is_free ?carrier - carrier ?carrier_space -
28         carrier_space )
29     (is_occupied_by ?carrier - carrier ?carrier_space -
30         carrier_space ?box - box)
31
32     (are_same_location ?location1 - location ?location2 -
33         location)
34
35     (occupied ?robot - robot)
36  )
37
38  (:functions
39     (box_weight ?b - box)
40     (content_weight ?c - content)
41     (carrier_weight ?c - carrier)
42  )
43
44  ;azione con cui un robot raccoglie il contenuto che si trova
45  ;nella sua posizione e lo ripone in un box
46  (:durative-action fill_box
47     :parameters (?loc - location ?box - box ?content -
48         content ?robot - robot)
49     :duration (= ?duration 1)
50     :condition (and
51         (at start (not(occupied ?robot)))
52         (over all (at ?box ?loc))
53         (over all (at ?robot ?loc))
54         (over all (at ?content ?loc))

```

```

51     (over all (is_empty ?box))
52   )
53   :effect (and
54     (at start (occupied ?robot))
55     (at end (assign (box_weight ?box) (content_weight
56       ?content)))
57     (at end (not(is_empty ?box)))
58     (at end (is_filled_by ?box ?content))
59     (at end (not(occupied ?robot)))
60   )
61 )
62 ;azione con cui il robot svuota il contenuto da un box e lo
    consegna ad una persona che ne ha bisogno nella sua
    posizione
63 (:durative-action empty_box_and_deliver
64   :parameters (?loc - location ?box - box ?content -
65     content ?person - person ?robot - robot)
66   :duration (= ?duration 1)
67   :condition (and
68     (at start (not(occupied ?robot)))
69     (over all (at ?box ?loc))
70     (over all (at ?robot ?loc))
71     (over all (at ?person ?loc))
72     (over all (is_filled_by ?box ?content))
73     (over all (needs ?person ?content))
74   )
75   :effect (and
76     (at start (occupied ?robot))
77     (at end (is_empty ?box))
78     (at end (not(is_filled_by ?box ?content)))
79     (at end (not(needs ?person ?content)))
80     (at end (assign (box_weight ?box) 0))
81     (at end (not(occupied ?robot)))
82   )
83 )
84 ;azione tramite cui il robot pu caricare un box su un
    carrello al fine di spostarlo successivamente
85 (:durative-action load_carrier
86   :parameters (?loc - location ?box - box ?carrier -
87     carrier ?space - carrier_space ?robot - robot )
88   :duration (= ?duration 1)
89   :condition (and
90     (at start (not(occupied ?robot)))
91     (at start (at ?box ?loc))
92     (over all (at ?robot ?loc))
93     (over all (at ?carrier ?loc))
94     (over all (is_free ?carrier ?space))

```

```

94   )
95   :effect (and
96     (at start (occupied ?robot))
97     (at start (not(at ?box ?loc)))
98     (at end (not(is_free ?carrier ?space)))
99     (at end (is_occupied_by ?carrier ?space ?box))
100    (at end (increase (carrier_weight ?carrier)
101      (box_weight ?box)))
102    (at end (not(occupied ?robot)))
103  )
104 )
105
106 ;azione tramite cui un robot scarica una box da un carretto
107 (:durative-action unload_carrier
108   :parameters (?loc - location ?box - box ?carrier -
109     carrier ?space - carrier_space ?robot - robot)
110   :duration (= ?duration 1)
111   :condition (and
112     (at start (not(occupied ?robot)))
113     (at start (is_occupied_by ?carrier ?space ?box))
114     (over all (at ?robot ?loc))
115     (over all (at ?carrier ?loc))
116   )
117   :effect (and
118     (at start (occupied ?robot))
119     (at start (not(is_occupied_by ?carrier ?space ?box)))
120     (at end (is_free ?carrier ?space))
121     (at end (at ?box ?loc))
122     (at end (decrease (carrier_weight ?carrier)
123       (box_weight ?box)))
124     (at end (not(occupied ?robot)))
125   )
126 )
127 (:durative-action move_carrier
128   :parameters (?initial_loc - location ?final_loc -
129     location ?carrier - carrier ?robot - robot)
130   :duration (= ?duration (* (carrier_weight ?carrier) 2))
131   :condition (and
132     (at start (not(occupied ?robot)))
133     (at start (not(are_same_location ?initial_loc
134       ?final_loc)))
135     (at start (at ?robot ?initial_loc))
136     (at start (at ?carrier ?initial_loc))
137   )
138   :effect (and
139     (at start (occupied ?robot))

```

```

138         (at start (not(at ?robot ?initial_loc)))
139         (at start (not(at ?carrier ?initial_loc)))
140         (at end (at ?robot ?final_loc))
141         (at end (at ?carrier ?final_loc))
142         (at end (not(occupied ?robot)))
143     )
144 )
145
146 (:durative-action simply_move
147   :parameters (?initial_loc - location ?final_loc -
148     location ?robot - robot)
149   :duration (= ?duration 2)
150   :condition (and
151     (at start (not(occupied ?robot)))
152     (at start (not(are_same_location ?initial_loc
153       ?final_loc)))
154     (at start (at ?robot ?initial_loc))
155   )
156   :effect (and
157     (at start (occupied ?robot))
158     (at start (not (at ?robot ?initial_loc)))
159     (at end (at ?robot ?final_loc))
160     (at end (not(occupied ?robot)))
161   )
162 )

```

Di seguito verranno descritti i nuovi predicati introdotti:

- **Azioni Durative (:durative-action):**

- Le azioni definite in questo dominio sono ora **durative actions**, il che significa che hanno una durata nel tempo. Ad esempio, l'azione **fill-box** rappresenta il riempimento di una scatola e ha una durata di **1 unità di tempo** (:duration (= ?duration 1)). Le condizioni e gli effetti sono specificati per l'inizio (*at start*) e la fine (*at end*) dell'azione durativa.
- Le azioni *empty-box-and-deliver*, *load-carrier*, *unload-carrier*, *move-carrier* e *simply-move* sono anch'esse durative e hanno specifiche condizioni e effetti in termini di variabili numeriche **fluents** come il peso delle scatole e dei carrelli, nonché lo stato di occupazione del robot.

Listing 3.2: instance1Temporal.pddl

```

1
2 (define (problem instance1d)
3   (:domain ESLD)
4   (:objects
5     depot l1 l2 - location
6     b1 b2 b3 b4 b5 - box
7     p1 p2 p3 - person
8     food medicine - content
9     robot - robot
10    carrier - carrier

```



```

11 space1 space2 space3 space4 - carrier_space
12 )
13
14 (:init (at b1 depot)(at b2 depot)(at b3 depot)(at b4
    depot)(at b5 depot)
15     (at food depot)(at medicine depot)
16     (at p1 l1)(at p2 l1)(at p3 l2)
17     (at robot depot)
18     (at carrier depot)
19     (isEmpty b1)(isEmpty b2)(isEmpty b3)(isEmpty b4)(isEmpty
    b5)
20     (needs p1 medicine)(needs p1 food)(needs p2
    medicine)(needs p3 food)
21     (isFree carrier space1)(isFree carrier space2)(isFree
    carrier space3)(isFree carrier space4)
22     (are_same_location depot depot)(are_same_location l1
    l1)(are_same_location l2 l2)
23
24     (= (content_weight medicine) 1)
25     (= (content_weight food) 2)
26 )
27
28 (:goal (and (not(needs p1 medicine))(not(needs p1
    food))(not(needs p2 medicine))(not(needs p3 food))))
29 )

```

- È stata introdotta l'inizializzazione delle variabili numeriche **fluents** con le funzioni *content-weight*. Ad esempio, il peso del contenuto medicine è impostato a 1, mentre il peso del contenuto food è impostato a 2. Queste funzioni rappresentano il peso dei contenuti e possono variare nel tempo.

```

first_solution_cpu_time: 0.12

plan computed:
  Time: (ACTION) [action Duration; action Cost]
  0.0000: (FILL_BOX DEPOT B1 MEDICINE ROBOT) [D:1.00; C:1.00]
  1.0000: (LOAD_CARRIER DEPOT B1 CARRIER SPACE3 ROBOT) [D:1.00; C:1.00]
  2.0000: (FILL_BOX DEPOT B3 MEDICINE ROBOT) [D:1.00; C:1.00]
  3.0000: (LOAD_CARRIER DEPOT B3 CARRIER SPACE4 ROBOT) [D:1.00; C:1.00]
  4.0000: (FILL_BOX DEPOT B2 FOOD ROBOT) [D:1.00; C:1.00]
  5.0000: (LOAD_CARRIER DEPOT B2 CARRIER SPACE1 ROBOT) [D:1.00; C:1.00]
  6.0000: (FILL_BOX DEPOT B5 FOOD ROBOT) [D:1.00; C:1.00]
  7.0000: (LOAD_CARRIER DEPOT B5 CARRIER SPACE2 ROBOT) [D:1.00; C:1.00]
  8.0000: (MOVE_CARRIER DEPOT L1 CARRIER ROBOT) [D:18.00; C:1.00]
  26.0000: (UNLOAD_CARRIER L1 B1 CARRIER SPACE3 ROBOT) [D:1.00; C:1.00]
  27.0000: (EMPTY_BOX_AND_DELIVER L1 B1 MEDICINE P1 ROBOT) [D:1.00; C:1.00]
  28.0000: (UNLOAD_CARRIER L1 B3 CARRIER SPACE4 ROBOT) [D:1.00; C:1.00]
  29.0000: (EMPTY_BOX_AND_DELIVER L1 B3 MEDICINE P2 ROBOT) [D:1.00; C:1.00]
  30.0000: (UNLOAD_CARRIER L1 B2 CARRIER SPACE1 ROBOT) [D:1.00; C:1.00]
  31.0000: (EMPTY_BOX_AND_DELIVER L1 B2 FOOD P1 ROBOT) [D:1.00; C:1.00]
  32.0000: (MOVE_CARRIER L1 L2 CARRIER ROBOT) [D:10.00; C:1.00]
  42.0000: (UNLOAD_CARRIER L2 B5 CARRIER SPACE2 ROBOT) [D:1.00; C:1.00]
  43.0000: (EMPTY_BOX_AND_DELIVER L2 B5 FOOD P3 ROBOT) [D:1.00; C:1.00]

Solution number: 1
Total time:      0.12
Search time:     0.12
Actions:         18
Duration:        44.000
Plan quality:    18.000
Total Num Flips: 1590
Plan file:       plan/tmp_1.SOL

Planner found 1 plan(s) in 0.269secs.

```

Figure 3.1: Piano Durativo


```

fill_box depot b1 medicine robot
load_carrier depot b1 carrier space3 robot
fill_box depot b3 medicine robot
load_carrier depot b3 carrier space4 robot
fill_box depot b2 food robot
load_carrier depot b2 carrier space1 robot
fill_box depot b5 food robot
load_carrier depot b5 carrier space2 robot
    move_carrier depot l1 carrier robot
        unload_carrier l1 b1 carrier space3 robot
        empty_box_and_deliver l1 b1 medicine p1 robot
        unload_carrier l1 b3 carrier space4 robot
        empty_box_and_deliver l1 b3 medicine p2 robot
        unload_carrier l1 b2 carrier space1 robot
        empty_box_and_deliver l1 b2 food p1 robot
            move_carrier l1 l2 carrier robot
                unload_carrier l2 b5 carrier space2 robot
                empty_box_and_deliver l2 b5 food p3 robot

```

Figure 3.2: Piano Durativo

3.2 Robotics Planning

Questo punto deve essere eseguito utilizzando il framework ROS insieme alla libreria PlanSys2 e consiste nell'esecuzione del piano temporizzato ottenuto nel punto 3.1. attraverso l'interazione di nodi ROS. Una parte importante del processo è l'adattamento del piano generato per renderlo compatibile con la sintassi del framework, formattata come segue:

Listing 3.3: Operazioni

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

Il piano deve tener conto dei tempi necessari per l'esecuzione delle azioni da parte dell'agente robotico. Inoltre, sono state definite le **fake actions** in C++, azioni fittizie che simulano le azioni temporizzate, descritte nel piano. Di seguito è mostrato il codice relativo a una singola fake action che riguarda l'azione di riempire una scatola (*fill-box*). Questo codice definirà il comportamento simulato dell'agente robotico per questa azione, tenendo conto della durata specificata nel piano.

Listing 3.4: Fake Actions - fill-box

```

1 // Copyright 2019 Intelligent Robotics Lab
2 //
3 // Licensed under the Apache License, Version 2.0 (the "
  License");
4 // you may not use this file except in compliance with the
  License.
5 // You may obtain a copy of the License at
6 //
7 //      http://www.apache.org/licenses/LICENSE-2.0

```

```

8 //
9 // Unless required by applicable law or agreed to in writing,
  software
10 // distributed under the License is distributed on an "AS IS"
  BASIS,
11 // WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
  express or implied.
12 // See the License for the specific language governing
  permissions and
13 // limitations under the License.
14
15 #include <memory>
16 #include <algorithm>
17 #include <vector>
18
19 #include "plansys2_executor/ActionExecutorClient.hpp"
20
21 #include "rclcpp/rclcpp.hpp"
22 #include "rclcpp_action/rclcpp_action.hpp"
23
24 using namespace std::chrono_literals;
25
26 class FillBoxAction : public plansys2::ActionExecutorClient{
27     float progress_;
28     public: FillBoxAction(): plansys2::ActionExecutorClient("
        fill_box", 200ms){
29         progress_ = 0.0;
30     }
31
32     private: void do_work(){
33         std::vector<std::string> arg = get_arguments();
34
35         if (progress_ < 1.0) {
36             progress_ += 0.2;
37             send_feedback(progress_, "At_location" + arg[0]+ "
                robot" + arg[3] + "is_filling_box" +
                arg[1] + "with_content" + arg[2]+\n");
38         } else {
39             finish(true, 1.0, "At_location" + arg[0]+ "robot"
                + arg[3] + "filled_box" + arg[1] + "
                with_content" + arg[2]+\n");
40
41             progress_ = 0.0;
42             std::cout << std::endl;
43         }
44
45         std::cout << "\r\e[K" << std::flush;
46         std::cout << "At_location" + arg[0]+ "robot" + arg
            [3] + "filling_box" + arg[1] + "with_content

```

```

    □" + arg[2]+ "...□[" << std::min(100.0,
    progress_ * 100.0) << "%]\n"<<
47     std::flush;
48     }
49 };
50
51 int main(int argc, char ** argv)
52 {
53     rclcpp::init(argc, argv);
54     auto node = std::make_shared<FillBoxAction>();
55
56     node->set_parameter(rclcpp::Parameter("action_name", "
        fill_box"));
57     node->trigger_transition(lifecycle_msgs::msg::Transition::
        TRANSITION_CONFIGURE);
58
59     rclcpp::spin(node->get_node_base_interface());
60
61     rclcpp::shutdown();
62
63     return 0;
64 }

```

In modo analogo sono state realizzate le seguenti Fake Actions:

- empty-box-and-deliver
- load-carrier
- move-carrier
- simply-move
- unload-carrier

3.2.1 Launch-File

Per definire un file di lancio (launch file) in Python per un sistema ROS 2 (Robot Operating System 2), è stato utilizzato il framework ROS 2 Launch. Questo file permette di configurare il lancio di vari nodi, specificare i parametri associati e impostare il namespace.

Listing 3.5: Launch.py

```

1
2 import os
3
4 from ament_index_python.packages import
    get_package_share_directory
5
6 from launch import LaunchDescription
7 from launch.actions import DeclareLaunchArgument,
    IncludeLaunchDescription
8 from launch.launch_description_sources import
    PythonLaunchDescriptionSource
9 from launch.substitutions import LaunchConfiguration
10 from launch_ros.actions import Node
11

```

```

12
13 def generate_launch_description():
14     # Get the launch directory
15     example_dir = get_package_share_directory('progettoai3')
16     namespace = LaunchConfiguration('namespace')
17
18     declare_namespace_cmd = DeclareLaunchArgument(
19         'namespace',
20         default_value='',
21         description='Namespace')
22
23     plansys2_cmd = IncludeLaunchDescription(
24         PythonLaunchDescriptionSource(os.path.join(
25             get_package_share_directory('plansys2_bringup'),
26             'launch',
27             'plansys2_bringup_launch_monolithic.py')),
28         launch_arguments={
29             'model_file': example_dir + '/pddl/domainD.pddl',
30             'namespace': namespace
31         }.items())
32
33     # Specify the actions
34
35     fill_box_cmd = Node(
36         package='progettoai3',
37         executable='fill_box',
38         name='fill_box',
39         namespace=namespace,
40         output='screen',
41         parameters=[])
42
43     load_carrier_cmd = Node(
44         package='progettoai3',
45         executable='load_carrier',
46         name='load_carrier',
47         namespace=namespace,
48         output='screen',
49         parameters=[])
50
51     simply_move_cmd = Node(
52         package='progettoai3',
53         executable='simply_move',
54         name='simply_move',
55         namespace=namespace,
56         output='screen',
57         parameters=[])
58
59     move_carrier_cmd = Node(
60         package='progettoai3',

```

```

61     executable='move_carrier',
62     name='move_carrier',
63     namespace=namespace,
64     output='screen',
65     parameters=[])
66
67     empty_box_and_deliver_cmd = Node(
68         package='progettoai3',
69         executable='empty_box_and_deliver',
70         name='empty_box_and_deliver',
71         namespace=namespace,
72         output='screen',
73         parameters=[])
74
75
76     unload_carrier_cmd = Node(
77         package='progettoai3',
78         executable='unload_carrier',
79         name='unload_carrier',
80         namespace=namespace,
81         output='screen',
82         parameters=[])
83
84     ld = LaunchDescription()
85
86     ld.add_action(declare_namespace_cmd)
87
88     # Declare the launch options
89     ld.add_action(plansys2_cmd)
90
91     ld.add_action(fill_box_cmd)
92     ld.add_action(load_carrier_cmd)
93     ld.add_action(simply_move_cmd)
94     ld.add_action(move_carrier_cmd)
95     ld.add_action(empty_box_and_deliver_cmd)
96     ld.add_action(unload_carrier_cmd)
97
98     return ld

```

3.2.2 Command File

A questo punto è stato creato un file contenente una serie di comandi da eseguire direttamente nel terminale di PlanSys2. Questo file traduce le informazioni dell'istanza del problema PDDL in un formato compatibile con la libreria di PlanSys2.

Listing 3.6: Command File

```

1 set instance R robot
2 set instance C carrier
3 set instance B1 box
4 set instance B2 box

```

```

5 set instance B3 box
6 set instance B4 box
7 set instance B5 box
8 set instance DEPOT location
9 set instance L1 location
10 set instance L2 location
11 set instance SPACE1 carrier_space
12 set instance SPACE2 carrier_space
13 set instance SPACE3 carrier_space
14 set instance SPACE4 carrier_space
15 set instance MEDICINE content
16 set instance FOOD content
17 set instance P1 person
18 set instance P2 person
19 set instance P3 person
20
21 set function (= (content_weight MEDICINE) 1)
22 set function (= (content_weight FOOD) 2)
23
24 set function (= (box_weight B1) 0)
25 set function (= (box_weight B2) 0)
26 set function (= (box_weight B3) 0)
27 set function (= (box_weight B4) 0)
28 set function (= (box_weight B5) 0)
29
30 set function (= (carrier_weight C) 3)
31
32 set predicate (at B1 DEPOT)
33 set predicate (at B2 DEPOT)
34 set predicate (at B3 DEPOT)
35 set predicate (at B4 DEPOT)
36 set predicate (at B5 DEPOT)
37 set predicate (at R DEPOT)
38 set predicate (at C DEPOT)
39 set predicate (at FOOD DEPOT)
40 set predicate (at MEDICINE DEPOT)
41 set predicate (at P1 L1)
42 set predicate (at P2 L1)
43 set predicate (at P3 L2)
44
45 set predicate (needs P1 FOOD)
46 set predicate (needs P1 MEDICINE)
47 set predicate (needs P2 MEDICINE)
48 set predicate (needs P3 FOOD)
49
50 set predicate (is_empty B1)
51 set predicate (is_empty B2)
52 set predicate (is_empty B3)
53 set predicate (is_empty B4)

```

```

54 set predicate (is_empty B5)
55
56 set predicate (is_free C SPACE1)
57 set predicate (is_free C SPACE2)
58 set predicate (is_free C SPACE3)
59 set predicate (is_free C SPACE4)
60
61 set predicate (are_same_location DEPOT DEPOT)
62 set predicate (are_same_location L1 L1)
63 set predicate (are_same_location L2 L2)
64
65 set goal (and (not(needs P1 MEDICINE))(not(needs P1
    FOOD))(not(needs P2 MEDICINE))(not(needs P3 FOOD)))

```

3.2.3 Plan.txt

Listing 3.7: Plan.txt

```

1 0.000: (fill_box DEPOT B2 MEDICINE R) [1.000]
2 1.000: (load_carrier DEPOT B2 C SPACE1 R) [1.000]
3 2.000: (fill_box DEPOT B4 MEDICINE R) [1.000]
4 3.000: (load_carrier DEPOT B4 C SPACE3 R) [1.000]
5 4.000: (fill_box DEPOT B5 FOOD R) [1.000]
6 5.000: (load_carrier DEPOT B5 C SPACE4 R) [1.000]
7 6.000: (fill_box DEPOT B3 FOOD R) [1.000]
8 7.000: (load_carrier DEPOT B3 C SPACE2 R) [1.000]
9 8.000: (move_carrier DEPOT L1 C R) [18.000]
10 26.000: (unload_carrier L1 B2 C SPACE1 R) [1.000]
11 27.000: (empty_box_and_deliver L1 B2 MEDICINE P2 R) [1.000]
12 28.000: (unload_carrier L1 B4 C SPACE3 R) [1.000]
13 29.000: (empty_box_and_deliver L1 B4 MEDICINE P1 R) [1.000]
14 30.000: (unload_carrier L1 B5 C SPACE4 R) [1.000]
15 31.000: (empty_box_and_deliver L1 B5 FOOD P1 R) [1.000]
16 32.000: (move_carrier L1 L2 C R) [10.000]
17 42.000: (unload_carrier L2 B3 C SPACE2 R) [1.000]
18 43.000: (empty_box_and_deliver L2 B3 FOOD P3 R) [1.000]

```

3.2.4 CMakeList File

Il file *CMakeLists* è un componente essenziale nell'ambiente di sviluppo di ROS2. Serve a definire le dipendenze del progetto e a fornire istruzioni su come i file sorgente C++ relativi alle *fake actions* devono essere compilati e distribuiti nel sistema.

Listing 3.8: CMakeList File

```

1
2 cmake_minimum_required(VERSION 3.5)
3 project(progettoai3)
4
5
6 find_package(ament_cmake REQUIRED)

```

```
7 find_package(rclcpp REQUIRED)
8 find_package(rclcpp_action REQUIRED)
9 find_package(plansys2_msgs REQUIRED)
10 find_package(plansys2_executor REQUIRED)
11 set(CMAKE_CXX_STANDARD 17)
12
13 set(dependencies
14     rclcpp
15     rclcpp_action
16     plansys2_msgs
17     plansys2_executor
18 )
19
20
21 add_executable(fill_box src/fill_box.cpp)
22ament_target_dependencies(fill_box ${dependencies})
23
24 add_executable(empty_box_and_deliver
25     src/empty_box_and_deliver.cpp)
26ament_target_dependencies(empty_box_and_deliver
27     ${dependencies})
28
29 add_executable(load_carrier src/load_carrier.cpp)
30ament_target_dependencies(load_carrier ${dependencies})
31
32 add_executable(simply_move src/simply_move.cpp)
33ament_target_dependencies(simply_move ${dependencies})
34
35 add_executable(move_carrier src/move_carrier.cpp)
36ament_target_dependencies(move_carrier ${dependencies})
37
38 add_executable(unload_carrier src/unload_carrier.cpp)
39ament_target_dependencies(unload_carrier ${dependencies})
40
41
42 install(DIRECTORY launch pddl DESTINATION
43     share/${PROJECT_NAME})
44
45 install(TARGETS
46     fill_box
47     empty_box_and_deliver
48     load_carrier
49     move_carrier
50     simply_move
51     unload_carrier
52
53     ARCHIVE DESTINATION lib
54     LIBRARY DESTINATION lib
55     RUNTIME DESTINATION lib/${PROJECT_NAME})
```



```

53 )
54
55 if (BUILD_TESTING)
56     find_package(ament_lint_auto REQUIRED)
57     ament_lint_auto_find_test_dependencies()
58
59     find_package(ament_cmake_gtest REQUIRED)
60 endif()
61
62 ament_export_dependencies(${dependencies})
63
64 ament_package()

```

3.2.5 Organizzazione del Workspace

Una volta creati i file di configurazione necessari per eseguire il progetto, otterremo una cartella con una struttura organizzata in questo modo:

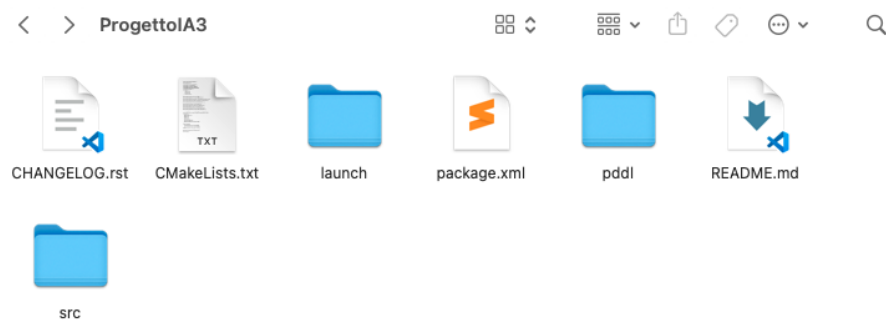


Figure 3.3: Workspace

- Nella cartella **launch** si trovano i file relativi all'avvio del programma, *file commands* e *plansys2-project-launch.py*
- La cartella **src** contenente i file che descrivono le *fake actions* in C++
- Nella cartella **pddl** si trovano i file *domainD.pddl* e *instanceID.pddl*, che sono rispettivamente il file di *dominio* e il file contenente il piano, definiti nella sezione 3.1.

3.2.6 Risultati

Listing 3.9: Terminale 1

```

2 colcon build --symlink-install
3 source install/local_setup.bash
4 ros2 launch progetto_ai3 progettoIA_launch.py

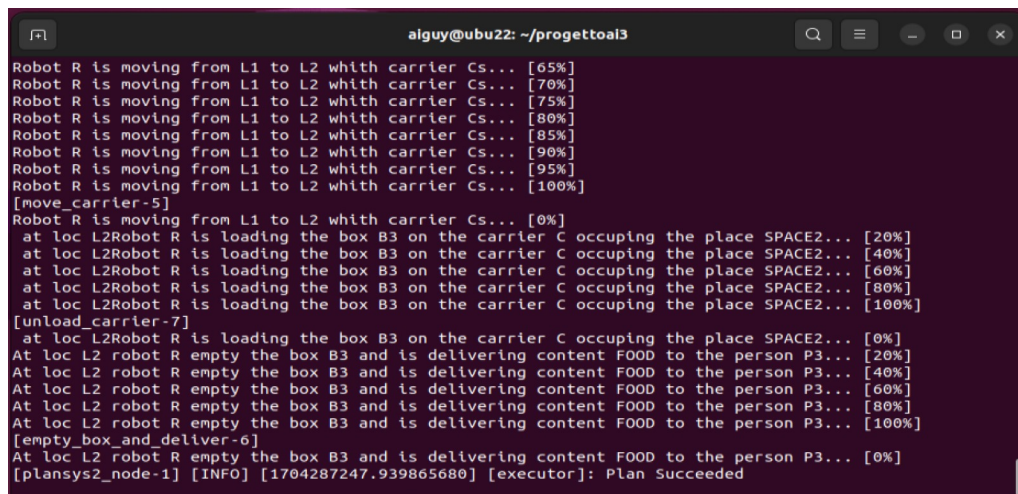
```

Listing 3.10: Terminale 2

```

1
2 ros2 run plansys2_terminal plansys2_terminal
3
4 Una volta avviato il terminale di plansys 2 si sono usati i
  seguenti comandi per caricare i vari comandi ed eseguire
  il plan:
5
6 source /home/aiguy/Desktop/progettoai3/launch/command
7 run plan-file /home/aiguy/Desktop/progettoai3/launch/plan.txt

```

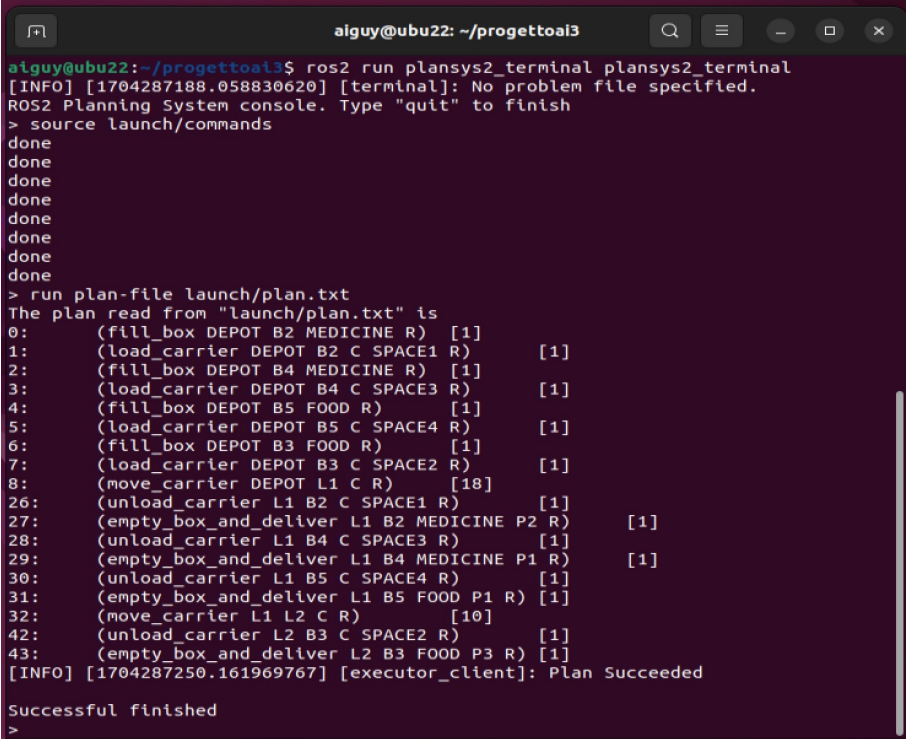


```

aiguy@ubu22: ~/progettoai3
Robot R is moving from L1 to L2 with carrier Cs... [65%]
Robot R is moving from L1 to L2 with carrier Cs... [70%]
Robot R is moving from L1 to L2 with carrier Cs... [75%]
Robot R is moving from L1 to L2 with carrier Cs... [80%]
Robot R is moving from L1 to L2 with carrier Cs... [85%]
Robot R is moving from L1 to L2 with carrier Cs... [90%]
Robot R is moving from L1 to L2 with carrier Cs... [95%]
Robot R is moving from L1 to L2 with carrier Cs... [100%]
[move_carrier-5]
Robot R is moving from L1 to L2 with carrier Cs... [0%]
  at loc L2Robot R is loading the box B3 on the carrier C occupying the place SPACE2... [20%]
  at loc L2Robot R is loading the box B3 on the carrier C occupying the place SPACE2... [40%]
  at loc L2Robot R is loading the box B3 on the carrier C occupying the place SPACE2... [60%]
  at loc L2Robot R is loading the box B3 on the carrier C occupying the place SPACE2... [80%]
  at loc L2Robot R is loading the box B3 on the carrier C occupying the place SPACE2... [100%]
[unload_carrier-7]
  at loc L2Robot R is loading the box B3 on the carrier C occupying the place SPACE2... [0%]
At loc L2 robot R empty the box B3 and is delivering content FOOD to the person P3... [20%]
At loc L2 robot R empty the box B3 and is delivering content FOOD to the person P3... [40%]
At loc L2 robot R empty the box B3 and is delivering content FOOD to the person P3... [60%]
At loc L2 robot R empty the box B3 and is delivering content FOOD to the person P3... [80%]
At loc L2 robot R empty the box B3 and is delivering content FOOD to the person P3... [100%]
[empty_box_and_deliver-6]
At loc L2 robot R empty the box B3 and is delivering content FOOD to the person P3... [0%]
[plansys2_node-1] [INFO] [1704287247.939865680] [executor]: Plan Succeeded

```

Figure 3.4: Terminale 1

A terminal window titled 'alguy@ubu22: ~/progettoal3' with standard Ubuntu window controls. The terminal shows the execution of 'ros2 run plansys2_terminal plansys2_terminal'. It displays several 'done' messages, then prompts to source 'launch/commands' and run 'plan-file launch/plan.txt'. The plan is read from 'launch/plan.txt' and lists 44 steps (0-43) including actions like 'fill_box', 'load_carrier', 'move_carrier', 'unload_carrier', and 'empty_box_and_deliver' with their durations in brackets. The terminal concludes with '[INFO] [1704287250.161969767] [executor_client]: Plan Succeeded' and 'Successful finished'.

```
alguy@ubu22:~/progettoal3$ ros2 run plansys2_terminal plansys2_terminal
[INFO] [1704287188.058830620] [terminal]: No problem file specified.
ROS2 Planning System console. Type "quit" to finish
> source launch/commands
done
done
done
done
done
done
done
done
done
> run plan-file launch/plan.txt
The plan read from "launch/plan.txt" is
0:      (fill_box DEPOT B2 MEDICINE R) [1]
1:      (load_carrier DEPOT B2 C SPACE1 R) [1]
2:      (fill_box DEPOT B4 MEDICINE R) [1]
3:      (load_carrier DEPOT B4 C SPACE3 R) [1]
4:      (fill_box DEPOT B5 FOOD R) [1]
5:      (load_carrier DEPOT B5 C SPACE4 R) [1]
6:      (fill_box DEPOT B3 FOOD R) [1]
7:      (load_carrier DEPOT B3 C SPACE2 R) [1]
8:      (move_carrier DEPOT L1 C R) [18]
26:     (unload_carrier L1 B2 C SPACE1 R) [1]
27:     (empty_box_and_deliver L1 B2 MEDICINE P2 R) [1]
28:     (unload_carrier L1 B4 C SPACE3 R) [1]
29:     (empty_box_and_deliver L1 B4 MEDICINE P1 R) [1]
30:     (unload_carrier L1 B5 C SPACE4 R) [1]
31:     (empty_box_and_deliver L1 B5 FOOD P1 R) [1]
32:     (move_carrier L1 L2 C R) [10]
42:     (unload_carrier L2 B3 C SPACE2 R) [1]
43:     (empty_box_and_deliver L2 B3 FOOD P3 R) [1]
[INFO] [1704287250.161969767] [executor_client]: Plan Succeeded

Successful finished
>
```

Figure 3.5: Terminale 2

A futuristic robot with a metallic, segmented head and torso is shown from the chest up. It is holding a large, white, rectangular document or screen with both hands. The robot's head is tilted slightly to the right, and its eyes are visible. The background is a dark, textured grey.

4. Deliverables

4.1 Organizzazione cartella progetto

La cartella principale del progetto, racchiude i vari punti che sono stati sviluppati. Per quanto riguarda il Punto 3, si rimanda al capitolo 3.2.5 della relazione dove è descritto in dettaglio il workspace.

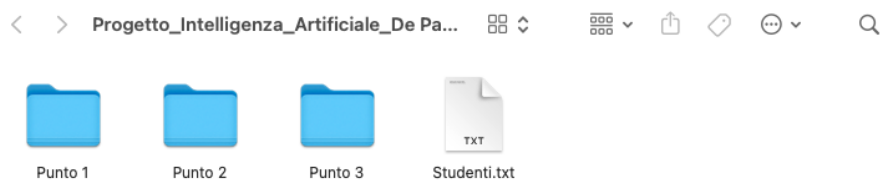


Figure 4.1: Organizzazione progetto

Nella Cartella Punto 1 si trovano i file PDDL, riguardanti il dominio e le tre istanze.

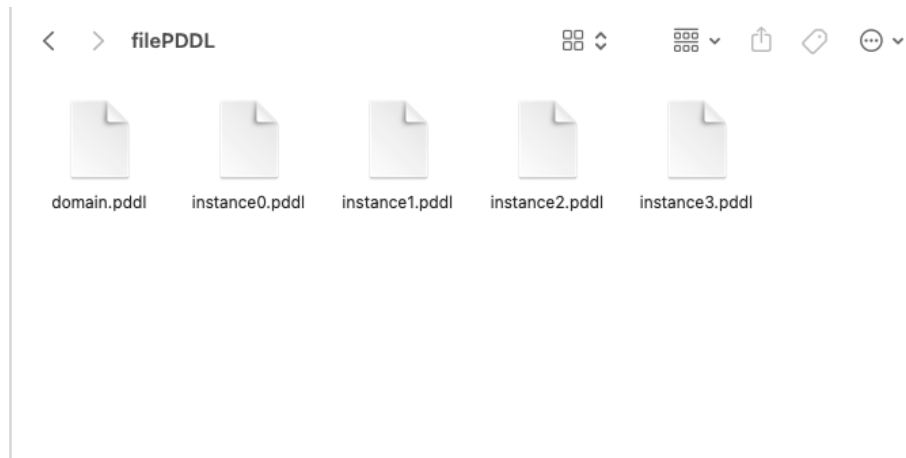


Figure 4.2: Punto 1

La cartella Punto 2 racchiude al suo interno il workspace creato con il Software IntelliJ IDEA. Si è usato la libreria PDDL4J per l'implementazione di un algoritmo A* che usa una euristica **MyHeuristic** sviluppata da noi.



Figure 4.3: Punto 2

I 3 file principali del Punto 2, si trovano nella cartella src.



Figure 4.4: Punto 2 - File PDDL4J