UNIVERSITÀ DEGLI STUDI
DI GENOVA

Artificial Intelligence for Robotics 2

Assignment 1 Report

Ali Yousefi

Reza Taleshi

BOUAZZA EL MOUTAOUAKIL

Part1: Domain Model Description

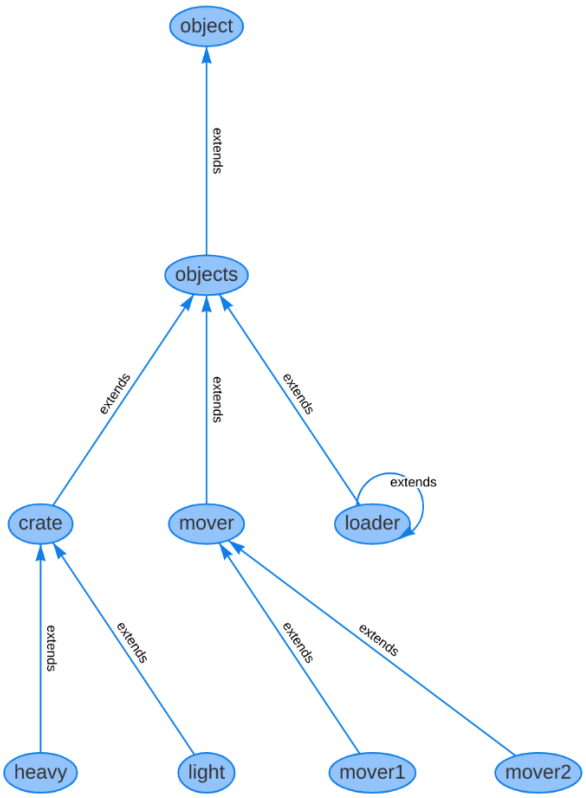
Considering the description of the automated warehouse scenario, we have numeric fluents such as crates weight and their distance from the loading bay, and also durative actions. So, we have to find a solution for this scenario using numeric and temporal planning.

1-1: Adding Requirements: *First step is to add the requirements for this kind of planning:*

```
(define (domain warehouse)
    (:requirements :strips :typing :negative-preconditions :durative-actions :numeric-fluents)
```

1-2: Defining Types: *We can define types for objects of the scenario to distinguish them, you can see the hierarchy of the objects used in this domain file: (generated by vscode)*

```
    (:types
        crate mover loader - objects
        heavy light - crate
        mover1 mover2 - mover
        loaderA  loaderB - loader
    )
```



1-3: Defining Predicates: *Same as classical planning, we have to define some predicates to specify the condition of our objects in each state:*

```
(:predicates
    ; defining some predicates based on our objects
    (carry-light-single ?m - mover ?c - light)
    (carry-light-double ?m1 - mover1 ?m2 - mover2 ?c - light)
    (carry-fragile ?m1 - mover1 ?m2 - mover2 ?c - fragile)
    (carry-heavy ?m1 ?m2 - mover ?c - heavy)
    (loader-free ?ld - loader)
    (mover-empty ?m - mover)
    (mover-at-loadingBay ?m - mover)
    (mover-at-StationCharge ?m - mover)
    (mover-near-crate ?m - mover ?c - crate)
    (crate-at-warehouse ?c - crate)
    (crate-at-loadingBay ?c - crate)
    (crate-at-conveyorBelt ?c - crate)
)
```

1-4: Adding Numeric Fluents: *Now we can add numeric fluents into our domain model:*

```
(:functions
    ; adding numeric fluents for crates
    (crate_distance ?c - crate)
    (crate_weight ?c - crate)
    (crate_fragile ?c - crate) ; must be set to 1 if IS NOT fragile and 1.5 if IS FRAGILE
    (mover_power ?m - mover)
)
```

1-5: Durative Actions:

1-5-1: move-to-loadingBay-heavy*: First durative action is for the state when mover1 and mover2 carry a heavy crate to the loading bay together, the required precondition for this action is to both movers carrying a heavy crate and get into the loading bay before the loader is free. It takes a specific time as described in the assignment specification for movers to get to the loading bay and finally the effect of this action is at end both movers will be at the loading bay:*

```
(:durative-action move-to-loadingBay-heavy
        ; mover1 and mover2 carry a heavy crate to the loadingBay together
        :parameters (?m1 - mover1 ?m2 - mover2 ?c - heavy ?ld - loader)
        :duration (= ?duration (/ (* (crate_distance ?c)(crate_weight ?c)) 100))
        :condition (and (at start (not (mover-at-loadingBay ?m1)))
                        (at start (not (mover-at-loadingBay ?m2)))
                        (over all (carry-heavy ?m1 ?m2 ?c))
                        (at end (loader-free ?ld))
                (at start (> (+(mover_power ?m1) 1)(/ (* (crate_distance ?c)(crate_weight ?c)) 100)))
                (at start (> (+(mover_power ?m2) 1)(/ (* (crate_distance ?c)(crate_weight ?c)) 100))))
```

```
    :effect (and (at end  (mover-at-loadingBay ?m1))
                 (at end  (mover-at-loadingBay ?m2))
                 (at end (not (mover-at-StationCharge ?m1)))
                 (at end (not (mover-at-StationCharge ?m2)))
            (at end (Decrease (mover_power ?m1)(/ (* (crate_distance ?c)(crate_weight ?c)) 100)))
            (at end (Decrease (mover_power ?m2)(/ (* (crate_distance ?c)(crate_weight ?c)) 100))))
)
```

1-5-2: move-to-loadingBay-light-single: *Second durative action is same as the first one but in this case we only have one mover carrying a light crate, the time which takes for this action can be derived with the same formula:*

```
(:durative-action move-to-loadingBay-light-single
    :parameters (?m - mover ?c - light ?ld - loader)
    :duration (= ?duration (/ (* (crate_distance ?c)(crate_weight ?c)) 100))
    :condition (and (at start (not (mover-at-loadingBay ?m)))
                    (over all (carry-light-single ?m ?c))
                    (at end (loader-free ?ld)))
             (at start (> (+(mover_power ?m) 1)(/ (* (crate_distance ?c)(crate_weight ?c)) 100))))
    :effect (and (at end  (mover-at-loadingBay ?m))
             (at end (Decrease (mover_power ?m)(/ (* (crate_distance ?c)(crate_weight ?c)) 100))))
)
```

1-5-3: move-to-loadingBay-light-double: *Next durative action is for when two movers carrying a light crate to get to the loading bay faster, so the formula is different from the previous ones.*

```
(:durative-action move-to-loadingBay-light-double
    :parameters (?m1 - mover1 ?m2 - mover2 ?c - light ?ld - loader)
    :duration (= ?duration (/ (* (crate_distance ?c)(crate_weight ?c)) 150))
    :condition (and (at start (not (mover-at-loadingBay ?m1)))
                    (at start (not (mover-at-loadingBay ?m2)))
                    (over all (carry-light-double ?m1 ?m2 ?c))
                    (at end (loader-free ?ld)))
             (at start (> (+(mover_power ?m1) 1)(/ (* (crate_distance ?c)(crate_weight ?c)) 150)))
             (at start (> (+(mover_power ?m2) 1)(/ (* (crate_distance ?c)(crate_weight ?c)) 150)))

    :effect (and (at end  (mover-at-loadingBay ?m1))
                 (at end  (mover-at-loadingBay ?m2)))
                  (at end (not (mover-at-StationCharge ?m1)))
                 (at end (not (mover-at-StationCharge ?m2)))
             (at end (> (+(mover_power ?m1) 1)(/ (* (crate_distance ?c)(crate_weight ?c)) 150)))
             (at end (> (+(mover_power ?m2) 1)(/ (* (crate_distance ?c)(crate_weight ?c)) 150)))

)
```

1-5-(Optional function): move-to-loadingBay-fragile*: Next durative action is for when two movers carrying a fragile crate to get to the loading bay with more care, so the formula is different from the previous ones.*

```
(:durative-action move-to-loadingBay-fragile
    :parameters (?m1 - mover1 ?m2 - mover2 ?c - fragile ?ld - loader)
    :duration (= ?duration (/ (* (crate_distance ?c)(crate_weight ?c)) 100))
    :condition (and (at start (not (mover-at-loadingBay ?m1)))
                    (at start (not (mover-at-loadingBay ?m2)))
                    (over all (carry-light-double ?m1 ?m2 ?c))
                    (at end (loader-free ?ld)))
               (at start (> (+(mover_power ?m1) 1)(/ (* (crate_distance ?c)(crate_weight ?c)) 100)))
               (at start (> (+(mover_power ?m2) 1)(/ (* (crate_distance ?c)(crate_weight ?c)) 100)))

    :effect (and (at end  (mover-at-loadingBay ?m1))
                 (at end  (mover-at-loadingBay ?m2)))
                  (at end (not (mover-at-StationCharge ?m1)))
                 (at end (not (mover-at-StationCharge ?m2)))
               (at end (> (+(mover_power ?m1) 1)(/ (* (crate_distance ?c)(crate_weight ?c)) 100)))
               (at end (> (+(mover_power ?m2) 1)(/ (* (crate_distance ?c)(crate_weight ?c)) 100)))

)
```

1-5-4: move-to-crate-light*: This action considers the situation when a mover is free and it is located in the loading bay, so it decides to move toward a light crate to pick it up, after the duration is passed the mover will be near the light crate.*

```
(:durative-action move-to-crate-light
    :parameters (?m - mover ?c - light)
    :duration (= ?duration (/ (crate_distance ?c) 10))
    :condition (and (at start (mover-empty ?m))
                    (at start (not (mover-near-crate ?m ?c)))
                    (over all (crate-at-warehouse ?c)))
                     (at start (> (+(mover_power ?m) 1)(/  (crate_distance ?c) 10)) ))
    :effect (and (at end (not (mover-at-loadingBay ?m)))
                 (at end (not (mover-at-StationCharge ?m)))
                (at end (mover-near-crate ?m ?c)))
                 (at end (Decrease (mover_power ?m) (/  (crate_distance ?c) 10))))
)
```

***1-5-5: move-to-crate-heavy**: This action is like the previous one but in this case both movers decide to move toward a heavy crate to pick it up together.*

```
 (:durative-action move-to-crate-heavy
        :parameters (?m1 - mover1 ?m2 - mover2 ?c - heavy)
        :duration (= ?duration (/ (crate_distance ?c) 10))
        :condition (and (at start (mover-empty ?m1))
                        (at start (mover-empty ?m2))
                        (at start (not (mover-near-crate ?m1 ?c)))
                        (at start (not (mover-near-crate ?m2 ?c)))
                         (at start (> (+(mover_power ?m1) 1)(/  (crate_distance ?c) 10)))
                        (at start (> (+(mover_power ?m2) 1)(/  (crate_distance ?c) 10))))
                        (over all (crate-at-warehouse ?c)))
        :effect (and (at end (not (mover-at-loadingBay ?m1)))
                     (at end (not (mover-at-StationCharge ?m1)))
                     (at end (mover-near-crate ?m1 ?c))
                     (at end (not (mover-at-loadingBay ?m2)))
                     (at end (not (mover-at-StationCharge ?m2)))
                     (at end (mover-near-crate ?m2 ?c)))
                     (at end (Decrease (mover_power ?m1) (/  (crate_distance ?c) 10 )) )
                     (at end (Decrease (mover_power ?m2) (/  (crate_distance ?c) 10 ))))
    )
```

***1-5-(optional function): move-to-crate-fragile**: This action is like the previous one but in this case both movers decide to move toward a fragile crate to pick it up together.*

```
 (:durative-action move-to-crate-fragile
        :parameters (?m1 - mover1 ?m2 - mover2 ?c - fragile)
        :duration (= ?duration (/ (crate_distance ?c) 10))
        :condition (and (at start (mover-empty ?m1))
                        (at start (mover-empty ?m2))
                        (at start (not (mover-near-crate ?m1 ?c)))
                        (at start (not (mover-near-crate ?m2 ?c)))
                         (at start (> (+(mover_power ?m1) 1)(/  (crate_distance ?c) 10)))
                        (at start (> (+(mover_power ?m2) 1)(/  (crate_distance ?c) 10))))
                        (over all (crate-at-warehouse ?c)))
        :effect (and (at end (not (mover-at-loadingBay ?m1)))
                     (at end (not (mover-at-StationCharge ?m1)))
                     (at end (mover-near-crate ?m1 ?c))
                     (at end (not (mover-at-loadingBay ?m2)))
                     (at end (not (mover-at-StationCharge ?m2)))
                     (at end (mover-near-crate ?m2 ?c)))
                     (at end (Decrease (mover_power ?m1) (/  (crate_distance ?c) 10 )) )
                     (at end (Decrease (mover_power ?m2) (/  (crate_distance ?c) 10 ))))
)
```

1-5-6: pick-light-single*: This action considers the situation in which a mover is near a light crate and it is empty, the effect of this action is mover at the end is not empty and carrying the crate, the duration for this action was not mentioned in the assignment description file so we decided it to be 1unit of time:*

```
(:durative-action pick-light-single
    :parameters (?m - mover ?c - light)
    :duration (= ?duration 1)
    :condition (and (at start (mover-empty ?m))
                    (over all (mover-near-crate ?m ?c))
                    (at start (crate-at-warehouse ?c)))
                     (at start (not (mover-at-loadingBay ?m)))
                    (at start (not (mover-at-StationCharge ?m)))
                    (at start (> (+(mover_power ?m) 1) 1)))

    :effect (and (at end (not (mover-empty ?m)))
                 (at end (not (crate-at-warehouse ?c)))
                 (at end (carry-light-single ?m ?c)))
                  (at end (Decrease (mover_power ?m)  1)))
)
```

1-5-7: pick-light-double: *Same as previous action we have a light crate but, in this case, the precondition is both movers have to be near the crate and empty, its effect is both of them won't be empty and carrying the crate:*

```
(:durative-action pick-light-double
    :parameters (?m1 - mover1 ?m2 - mover2 ?c - light)
    :duration (= ?duration 1)
    :condition (and (at start (mover-empty ?m1))
                    (at start (mover-empty ?m2))
                    (over all (mover-near-crate ?m1 ?c))
                    (over all (mover-near-crate ?m2 ?c))
                     (at start (not (mover-at-loadingBay ?m1)))
                    (at start (not (mover-at-loadingBay ?m2)))
                    (at start (not (mover-at-StationCharge ?m1)))
                    (at start (not (mover-at-StationCharge ?m2)))
                     (at start (crate-at-warehouse ?c)))
                     (at start (> (+(mover_power ?m1) 1) 1))
                    (at start (> (+(mover_power ?m2) 1) 1)))
    :effect (and (at end (not (mover-empty ?m1)))
                 (at end (not (mover-empty ?m2)))
                 (at end (not (crate-at-warehouse ?c)))
                 (at end (carry-light-double ?m1 ?m2 ?c)))
                  (at end (Decrease (mover_power ?m1)  1))
                 (at end (Decrease (mover_power ?m2)  1)))
)
```

1-5-(optional function): pick-fragile: *Same as previous action we have a fragile crate but:*

```
(:durative-action pick-fragile
    :parameters (?m1 - mover1 ?m2 - mover2 ?c - fragile)
    :duration (= ?duration 1)
    :condition (and (at start (mover-empty ?m1))
                    (at start (mover-empty ?m2))
                    (over all (mover-near-crate ?m1 ?c))
                    (over all (mover-near-crate ?m2 ?c))
                     (at start (not (mover-at-loadingBay ?m1)))
                    (at start (not (mover-at-loadingBay ?m2)))
                    (at start (not (mover-at-StationCharge ?m1)))
                    (at start (not (mover-at-StationCharge ?m2)))
                     (at start (crate-at-warehouse ?c)))
                     (at start (> (+(mover_power ?m1) 1) 1))
                    (at start (> (+(mover_power ?m2) 1) 1)))
    :effect (and (at end (not (mover-empty ?m1)))
                (at end (not (mover-empty ?m2)))
                (at end (not (crate-at-warehouse ?c)))
                (at end (carry-fragile ?m1 ?m2 ?c)))
                 (at end (Decrease (mover_power ?m1)  1))
                (at end (Decrease (mover_power ?m2)  1)))
)
```

1-5-8: drop-light-single: *It consider the situation in which a mover is carrying a light crate and it is located in the loading bay, the effect of this action is the mover will be empty at the end and the crate will be located at the loading bay, the duration for this action was not mentioned in the assignment description file so we decided it to be 1unit of time:*

```
(:durative-action drop-light-single
    :parameters (?m - mover ?c – light ?d -loader)
    :duration (= ?duration 1)
    :condition (and (at start (carry-light-single ?m ?c))
                    (over all (mover-at-loadingBay ?m)))
                     (at start (not (mover-empty ?m)))
                    (at start (loader-free ?d))
                    (at start (> (+(mover_power ?m) 1) 1)))

    :effect (and (at start (not (carry-light-single ?m ?c)))
                    (at end (crate-at-loadingBay ?c))
                    (at end (mover-empty ?m)))
                     (at end (not (loader-free ?d)))
                    (at end (Decrease (mover_power ?m)  1)))

)
```

1-5-9: drop-light-double: *Same as previous action, but in this case both movers are carrying the light crate:*

```
    (:durative-action drop-light-double
        :parameters (?m1 - mover1 ?m2 - mover2 ?c – light ?d - loader)
        :duration (= ?duration 1)
        :condition (and (at start (carry-light-double ?m1 ?m2 ?c))
                        (over all (mover-at-loadingBay ?m1))
                        (over all (mover-at-loadingBay ?m2)))
                        (at start (not (mover-empty ?m1)))
                        (at start (not (mover-empty ?m2)))
                        (at start  (loader-free ?d))
                        (at start (> (+(mover_power ?m1) 1) 1))
                        (at start (> (+(mover_power ?m2) 1) 1)))

        :effect (and (at start (not (carry-light-double ?m1 ?m2 ?c)))
                        (at end (crate-at-loadingBay ?c))
                        (at end (mover-empty ?m1))
                        (at end (mover-empty ?m2)))
                        (at end (not (loader-free ?d)))
                        (at end (Decrease (mover_power ?m1)  1))
                        (at end (Decrease (mover_power ?m2)  1))))
```

1-5-(optional function): drop-fragile: *Same as previous action, but in this case are carrying the fragile crate:*

```
    (:durative-action drop-fragile
        :parameters (?m1 - mover1 ?m2 - mover2 ?c – fragile ?d - loader)
        :duration (= ?duration 1)
        :condition (and (at start (carry-fragile ?m1 ?m2 ?c))
                        (over all (mover-at-loadingBay ?m1))
                        (over all (mover-at-loadingBay ?m2)))
                        (at start (not (mover-empty ?m1)))
                        (at start (not (mover-empty ?m2)))
                        (at start  (loader-free ?d))
                        (at start (> (+(mover_power ?m1) 1) 1))
                        (at start (> (+(mover_power ?m2) 1) 1)))

        :effect (and (at start (not (carry-fragile ?m1 ?m2 ?c)))
                        (at end (crate-at-loadingBay ?c))
                        (at end (mover-empty ?m1))
                        (at end (mover-empty ?m2)))
                        (at end (not (loader-free ?d)))
                        (at end (Decrease (mover_power ?m1)  1))
                        (at end (Decrease (mover_power ?m2)  1)))
)
```

1-5-10: pick-heavy: *The preconditions for this action is both movers to be near the heavy crate and they are empty, the effect is both movers won't be empty anymore and the crate won't be at the warehouse, the duration for this action was not mentioned in the assignment description file so we decided it to be 1unit of time:*

```
(:durative-action pick-heavy
    :parameters (?m1 - mover1 ?m2 - mover2 ?c - heavy)
    :duration (= ?duration 1)
    :condition (and (at start (crate-at-warehouse ?c))
                    (over all (mover-near-crate ?m1 ?c))
                    (over all (mover-near-crate ?m2 ?c))
                    (at start (mover-empty ?m1))
                    (at start (mover-empty ?m2)))
                     (at start (not (mover-at-loadingBay ?m1)))
                    (at start (not (mover-at-loadingBay ?m2)))
                    (at start (not (mover-at-StationCharge ?m1)))
                    (at start (not (mover-at-StationCharge ?m2)))
                    (at start (> (+(mover_power ?m1) 1) 1))
                    (at start (> (+(mover_power ?m2) 1) 1)))

    :effect (and (at end (not (mover-empty ?m1)))
                 (at end (not (mover-empty ?m2)))
                 (at end (carry-heavy ?m1 ?m2 ?c))
                 (at end (not (crate-at-warehouse ?c))))
                  (at end (Decrease (mover_power ?m1)  1))
                 (at end (Decrease (mover_power ?m2)  1))))
```

1-5-11: drop-heavy: *It consider the situation when the movers are carrying a heavy crate and they are located in the loading bay, effect of this action is both movers will be empty and heavy crate will be at the loadind bay, the duration for this action was not mentioned in the assignment description file so we decided it to be time 1 :*

```
(:durative-action drop-heavy
    :parameters (?m1 - mover1 ?m2 - mover2 ?c – heavy ?d - loader)
    :duration (= ?duration 1)
    :condition (and (at start (carry-heavy ?m1 ?m2 ?c))
                    (over all (mover-at-loadingBay ?m1))
                    (over all (mover-at-loadingBay ?m2)))
                     (at start (not (mover-empty ?m1)))
                    (at start (not (mover-empty ?m1)))
                    (at start (loader-free ?d))
                    (at start (> (+(mover_power ?m1) 1) 1))
                    (at start (> (+(mover_power ?m2) 1) 1)))
    :effect (and (at end (not (carry-heavy ?m1 ?m2 ?c)))
                 (at end (crate-at-loadingBay ?c))
                 (at end (mover-empty ?m1))
                 (at end (mover-empty ?m2)))
```

```
                    (at end (not (loader-free ?d)))
                    (at end (Decrease (mover_power ?m1)  1))
                    (at end (Decrease (mover_power ?m2)  1))))
```

1-5-12: load: *It consider the situation in which, a crate is located in the loading bay, it takes 4 (0r 6 if is fragile) units of time to load a crate, and during the load process, the loader is no longer free:*

```
    (:durative-action load_first_loader ; the principle one
        :parameters (?ld - loaderA ?c - crate)
        :duration (= ?duration (* (crate_fragile ?c )  4))
        :condition (and (at start (crate-at-loadingBay ?c))
                        (at start (not (crate-at-conveyorBelt ?c))))
        :effect (and (at end (not (crate-at-loadingBay ?c)))
                    (at end (crate-at-conveyorBelt ?c))
                    (at start (not (loader-free ?ld)))
                    (at end (loader-free ?ld))))
)
    (:durative-action load_second_loader ; the principle one ;OPTIONAL-FUNCTION
        :parameters (?ld - loaderB ?c - crate)
        :duration (= ?duration (* (crate_fragile ?c )  4))
        :condition (and (at start (crate-at-loadingBay ?c))
                        (at start (not (crate-at-conveyorBelt ?c)))
                        (at start (<(crate_weight ?c ) 50)))
        :effect (and (at end (not (crate-at-loadingBay ?c)))
                    (at end (crate-at-conveyorBelt ?c))
                    (at start (not (loader-free ?ld)))
                    (at end (loader-free ?ld))))
)
```

1-5-12: move-to-StationCharge: *This function let robot go charge station, that is necessary to charge the robot*

```
(:durative-action move-to-StationCharge ;OPTIONAL-FUNCTION
        :parameters (?m - mover  )
        :duration (= ?duration 1)
        :condition (and (at start (not (mover-at-StationCharge ?m))))
        :effect (and (at end  (mover-at-StationCharge ?m))
              (at end  (not(mover-at-loadingBay ?m)))
              (at end (Decrease (mover_power ?m) 1)))
    )
```

1-5-12: Charge: *This function let robot charge .*

```
(:durative-action Charge
        :parameters (?m - mover )
        :duration (= ?duration 8)
        :condition (and (at start  (mover-at-StationCharge ?m)))
        :effect (and (at end  (mover-at-StationCharge ?m))
              (at end (assign (mover_power ?m) 20))))
```

Part2: Analysis of the Planning Engine:

As it was mentioned in the assignment specification file, there are 4 problems, we used LPG to find solutions for these problems.

2-1: Problem1: *In the first problem we have three crates, one of them is heavy and the others are light and fragile; we are set the problem with the distance, the weight and if it is fragile or not for the crate, and we are set the power of the move robot; and in the end we are set goal. That is the solution:*

```
; Version LPG-td-1.4
; Seed 74872166
; Command line: ./lpg-td -o warehouse.pddl -f warehouse-prob1.pddl -n 3
; Problem warehouse-prob1.pddl
; Time 1.98
; Search time 1.94
; Parsing time 0.03
; Mutex time 0.01
; MakeSpan 39.00

0.0003:   (MOVE-TO-CRATE-FRAGILE MOVER1 MOVER2 FRAGILE) [2.0000]
2.0005:   (PICK-FRAGILE MOVER1 MOVER2 FRAGILE) [1.0000]
3.0008:   (MOVE-TO-LOADINGBAY-FRAGILE MOVER1 MOVER2 FRAGILE LOADER_B) [4.0000]
7.0010:   (DROP-FRAGILE MOVER1 MOVER2 FRAGILE LOADER_B) [1.0000]
8.0012:   (LOAD_SECONDE_LOUDER LOADER_B FRAGILE) [6.0000]
8.0012:   (MOVE-TO-CRATE-HEAVY MOVER1 MOVER2 HEAVY) [1.0000]
9.0015:   (PICK-HEAVY MOVER1 MOVER2 HEAVY) [1.0000]
10.0017:   (MOVE-TO-LOADINGBAY-HEAVY MOVER1 MOVER2 HEAVY LOADER_B) [7.0000]
17.0020:   (DROP-HEAVY MOVER1 MOVER2 HEAVY LOADER_A) [1.0000]
18.0022:   (LOAD_FIRST_LOADER LOADER_A HEAVY) [4.0000]
18.0025:   (MOVE-TO-SATIONCHARGE MOVER1) [1.0000]
19.0027:   (CHARGE MOVER1) [8.0000]
27.0032:   (MOVE-TO-CRATE-LIGHT MOVER1 LIGHT1) [2.0000]
29.0035:   (PICK-LIGHT-SINGLE MOVER1 LIGHT1) [1.0000]
30.0037:   (MOVE-TO-LOADINGBAY-LIGHT-SINGLE MOVER1 LIGHT1 LOADER_A) [4.0000]
34.0040:   (DROP-LIGHT-SINGLE MOVER1 LIGHT1 LOADER_A) [1.0000]
35.0043:   (LOAD_SECONDE_LOUDER LOADER_B LIGHT1) [4.0000]
```

2-2: Problem2: *In the second problem, we have four crates, two of them are light and the others are light and fragile, we are set the problem with the distance, the weight and if it is fragile or not for the crate, and we are set the power of the move robot; and in the end we are set goal. That is the solution:*

```
; Version LPG-td-1.4
; Seed 62928053
; Command line: ./lpg-td -o warehouse.pddl -f warehouse-prob2.pddl -n 4
; Problem warehouse-prob2.pddl
; Time 16.75
; Search time 16.71
; Parsing time 0.03
; Mutex time 0.01
; MakeSpan 66.00

0.0002:   (MOVE-TO-CRATE-LIGHT MOVER2 LIGHT2) [1.0000]
1.0005:   (PICK-LIGHT-SINGLE MOVER2 LIGHT2) [1.0000]
2.0007:   (MOVE-TO-LOADINGBAY-LIGHT-SINGLE MOVER2 LIGHT2 LOADER_A) [3.0000]
5.0010:   (DROP-LIGHT-SINGLE MOVER2 LIGHT2 LOADER_A) [1.0000]
6.0012:   (LOAD_FIRST_LOADER LOADER_A LIGHT2) [4.0000]
0.0003:   (MOVE-TO-CRATE-LIGHT MOVER1 LIGHT1) [2.0000]
2.0005:   (PICK-LIGHT-SINGLE MOVER1 LIGHT1) [1.0000]
3.0008:   (MOVE-TO-LOADINGBAY-LIGHT-SINGLE MOVER1 LIGHT1 LOADER_B) [4.0000]
7.0010:   (DROP-LIGHT-SINGLE MOVER1 LIGHT1 LOADER_B) [1.0000]
8.0012:   (MOVE-TO-CRATE-HEAVY MOVER1 MOVER2 HEAVY1) [1.0000]
9.0015:   (PICK-HEAVY MOVER1 MOVER2 HEAVY1) [1.0000]
10.0017:   (MOVE-TO-LOADINGBAY-HEAVY MOVER1 MOVER2 HEAVY1 LOADER_A) [7.0000]
17.0020:   (DROP-HEAVY MOVER1 MOVER2 HEAVY1 LOADER_A) [1.0000]
17.0022:   (LOAD_FIRST_LOADER LOADER_A LIGHT1) [4.0000]
18.0025:   (MOVE-TO-SATIONCHARGE MOVER1) [1.0000]
```

```
19.0027:   (CHARGE MOVER1) [8.0000]
27.0032:   (MOVE-TO-CRATE-FRAGILE MOVER1 MOVER2 FRAGILE) [2.0000]
29.0035:   (PICK-FRAGILE MOVER1 MOVER2 FRAGILE) [1.0000]
30.0040:   (MOVE-TO-SATIONCHARGE MOVER2) [1.0000]
31.0042:   (CHARGE MOVER2) [8.0000]
39.0048:   (MOVE-TO-LOADINGBAY-FRAGILE MOVER1 MOVER2 FRAGILE LOADER_A) [16.0000]
55.0050:   (DROP-FRAGILE MOVER1 MOVER2 FRAGILE LOADER_A) [1.0000]
56.0053:   (LOAD_FIRST_LOADER LOADER_A FRAGILE) [6.0000]
62.0058:   (LOAD_FIRST_LOADER LOADER_A HEAVY1) [4.0000]
```

2-3: Problem3: *In the third problem, we have one light, one fragile crate and two heavy crates we are set the problem with the distance, the weight and if it is fragile or not for the crate, and we are set the power of the move robot; and in the end we are set goal. That is the solution:*

```
; Version LPG-td-1.4
; Seed 49091372
; Command line: ./lpg-td -o warehouse.pddl -f warehouse-prob3.pddl -n 1
; Problem warehouse-prob3.pddl
; Time 1.69
; Search time 1.65
; Parsing time 0.03
; Mutex time 0.01
; MakeSpan 125.00

0.0003:   (MOVE-TO-CRATE-HEAVY MOVER1 MOVER2 HEAVY1) [2.0000]
2.0005:   (PICK-HEAVY MOVER1 MOVER2 HEAVY1) [1.0000]
3.0010:   (MOVE-TO-SATIONCHARGE MOVER2) [1.0000]
4.0013:   (CHARGE MOVER2) [8.0000]
3.0010:   (MOVE-TO-SATIONCHARGE MOVER1) [1.0000]
4.0013:   (CHARGE MOVER1) [8.0000]
4.0013:   (CHARGE MOVER2) [8.0000]
12.0017:   (MOVE-TO-LOADINGBAY-HEAVY MOVER1 MOVER2 HEAVY1 LOADER_B) [14.0000]
26.0020:   (DROP-HEAVY MOVER1 MOVER2 HEAVY1 LOADER_A) [1.0000]
27.0022:   (LOAD_FIRST_LOADER LOADER_A HEAVY1) [4.0000]
27.0025:   (MOVE-TO-SATIONCHARGE MOVER1) [1.0000]
28.0028:   (CHARGE MOVER1) [8.0000]
36.0033:   (MOVE-TO-CRATE-LIGHT MOVER1 LIGHT) [1.0000]
27.0025:   (MOVE-TO-SATIONCHARGE MOVER2) [1.0000]
28.0028:   (CHARGE MOVER2) [8.0000]
37.0035:   (PICK-LIGHT-SINGLE MOVER1 LIGHT) [1.0000]
38.0040:   (MOVE-TO-SATIONCHARGE MOVER1) [1.0000]
39.0043:   (CHARGE MOVER1) [8.0000]
47.0048:   (MOVE-TO-LOADINGBAY-LIGHT-SINGLE MOVER1 LIGHT LOADER_A) [3.0000]
50.0050:   (DROP-LIGHT-SINGLE MOVER1 LIGHT LOADER_A) [1.0000]
51.0055:   (MOVE-TO-SATIONCHARGE MOVER1) [1.0000]
52.0058:   (CHARGE MOVER1) [8.0000]
28.0028:   (CHARGE MOVER2) [8.0000]
28.0028:   (CHARGE MOVER2) [8.0000]
28.0028:   (CHARGE MOVER2) [8.0000]
28.0028:   (CHARGE MOVER2) [8.0000]
28.0028:   (CHARGE MOVER2) [8.0000]
28.0028:   (CHARGE MOVER2) [8.0000]
60.0063:   (MOVE-TO-CRATE-FRAGILE MOVER1 MOVER2 FRAGILE) [2.0000]
62.0065:   (PICK-FRAGILE MOVER1 MOVER2 FRAGILE) [1.0000]
63.0068:   (MOVE-TO-LOADINGBAY-FRAGILE MOVER1 MOVER2 FRAGILE LOADER_B) [16.0000]
79.0070:   (DROP-FRAGILE MOVER1 MOVER2 FRAGILE LOADER_B) [1.0000]
80.0075:   (MOVE-TO-SATIONCHARGE MOVER1) [1.0000]
81.0078:   (CHARGE MOVER1) [8.0000]
80.0075:   (MOVE-TO-SATIONCHARGE MOVER2) [1.0000]
81.0078:   (CHARGE MOVER2) [8.0000]
81.0078:   (CHARGE MOVER1) [8.0000]
81.0078:   (CHARGE MOVER2) [8.0000]
81.0078:   (CHARGE MOVER1) [8.0000]
81.0078:   (CHARGE MOVER2) [8.0000]
```

```
79.0073:   (LOAD_SECONDE_LOUDER LOADER_B LIGHT) [4.0000]
80.0073:   (LOAD_FIRST_LOADER LOADER_A FRAGILE) [6.0000]
89.0083:   (MOVE-TO-CRATE-HEAVY MOVER1 MOVER2 HEAVY2) [3.0000]
92.0085:   (PICK-HEAVY MOVER1 MOVER2 HEAVY2) [1.0000]
93.0090:   (MOVE-TO-SATIONCHARGE MOVER1) [1.0000]
94.0093:   (CHARGE MOVER1) [8.0000]
93.0090:   (MOVE-TO-SATIONCHARGE MOVER2) [1.0000]
94.0093:   (CHARGE MOVER2) [8.0000]
102.0098:   (MOVE-TO-LOADINGBAY-HEAVY MOVER1 MOVER2 HEAVY2 LOADER_B) [18.0000]
120.0101:   (DROP-HEAVY MOVER1 MOVER2 HEAVY2 LOADER_B) [1.0000]
121.0103:   (LOAD_FIRST_LOADER LOADER_A HEAVY2) [4.0000]
```

2-4: Problem4: *In the last problem we have two light crates, and four fragile crates we are set the problem with the distance, the weight and if it is fragile or not for the crate, and we are set the power of the move robot; and in the end we are set goal. That is the solution:*

```
; Version LPG-td-1.4
; Seed 79233237
; Command line: ./lpg-td -o warehouse.pddl -f warehouse-prob4.pddl -n 5
; Problem warehouse-prob4.pddl
; Time 5.28
; Search time 5.23
; Parsing time 0.04
; Mutex time 0.01
; MakeSpan 91.00

0.0003:   (MOVE-TO-CRATE-LIGHT MOVER1 LIGHT1) [2.0000]
2.0005:   (PICK-LIGHT-SINGLE MOVER1 LIGHT1) [1.0000]
3.0007:   (MOVE-TO-LOADINGBAY-LIGHT-SINGLE MOVER1 LIGHT1 LOADER_A) [6.0000]
9.0010:   (DROP-LIGHT-SINGLE MOVER1 LIGHT1 LOADER_A) [1.0000]
10.0012:   (LOAD_SECONDE_LOUDER LOADER_B LIGHT1) [4.0000]
10.0012:   (MOVE-TO-CRATE-FRAGILE MOVER1 MOVER2 FRAGILE4) [3.0000]
13.0015:   (PICK-FRAGILE MOVER1 MOVER2 FRAGILE4) [1.0000]
14.0020:   (MOVE-TO-SATIONCHARGE MOVER1) [1.0000]
15.0022:   (CHARGE MOVER1) [8.0000]
23.0028:   (MOVE-TO-LOADINGBAY-FRAGILE MOVER1 MOVER2 FRAGILE4 LOADER_B) [9.0000]
32.0030:   (DROP-FRAGILE MOVER1 MOVER2 FRAGILE4 LOADER_B) [1.0000]
33.0033:   (LOAD_SECONDE_LOUDER LOADER_B FRAGILE4) [6.0000]
33.0033:   (MOVE-TO-CRATE-FRAGILE MOVER1 MOVER2 FRAGILE2) [1.0000]
34.0035:   (PICK-FRAGILE MOVER1 MOVER2 FRAGILE2) [1.0000]
35.0040:   (MOVE-TO-SATIONCHARGE MOVER2) [1.0000]
36.0043:   (CHARGE MOVER2) [8.0000]
44.0048:   (MOVE-TO-LOADINGBAY-FRAGILE MOVER1 MOVER2 FRAGILE2 LOADER_B) [3.0000]
47.0050:   (DROP-FRAGILE MOVER1 MOVER2 FRAGILE2 LOADER_B) [1.0000]
48.0053:   (LOAD_SECONDE_LOUDER LOADER_B FRAGILE2) [6.0000]
48.0055:   (MOVE-TO-SATIONCHARGE MOVER1) [1.0000]
49.0058:   (CHARGE MOVER1) [8.0000]
57.0063:   (MOVE-TO-CRATE-FRAGILE MOVER1 MOVER2 FRAGILE1) [2.0000]
59.0065:   (PICK-FRAGILE MOVER1 MOVER2 FRAGILE1) [1.0000]
60.0068:   (MOVE-TO-LOADINGBAY-FRAGILE MOVER1 MOVER2 FRAGILE1 LOADER_B) [4.0000]
64.0070:   (DROP-FRAGILE MOVER1 MOVER2 FRAGILE1 LOADER_B) [1.0000]
65.0073:   (LOAD_FIRST_LOADER LOADER_A FRAGILE1) [6.0000]
65.0073:   (MOVE-TO-CRATE-FRAGILE MOVER1 MOVER2 FRAGILE3) [2.0000]
67.0075:   (PICK-FRAGILE MOVER1 MOVER2 FRAGILE3) [1.0000]
68.0078:   (MOVE-TO-LOADINGBAY-FRAGILE MOVER1 MOVER2 FRAGILE3 LOADER_A) [4.0000]
72.0080:   (DROP-FRAGILE MOVER1 MOVER2 FRAGILE3 LOADER_A) [1.0000]
73.0083:   (LOAD_FIRST_LOADER LOADER_A FRAGILE3) [6.0000]
73.0083:   (MOVE-TO-CRATE-LIGHT MOVER1 LIGHT6) [1.0000]
74.0085:   (PICK-LIGHT-SINGLE MOVER1 LIGHT6) [1.0000]
75.0090:   (MOVE-TO-SATIONCHARGE MOVER1) [1.0000]
76.0093:   (CHARGE MOVER1) [8.0000]
84.0098:   (MOVE-TO-LOADINGBAY-LIGHT-SINGLE MOVER1 LIGHT6 LOADER_A) [2.0000]
86.0101:   (DROP-LIGHT-SINGLE MOVER1 LIGHT6 LOADER_A) [1.0000]
87.0103:   (LOAD_SECONDE_LOUDER LOADER_B LIGHT6) [4.0000]
```