CS 395 A

PROJECT SUPERVISOR-
PROFESSOR SUBHAJIT ROY

PROJECT MENTOR-
NITESH TRIVEDI

PRESENTED BY-
ARYAN MAURYA

PDDL

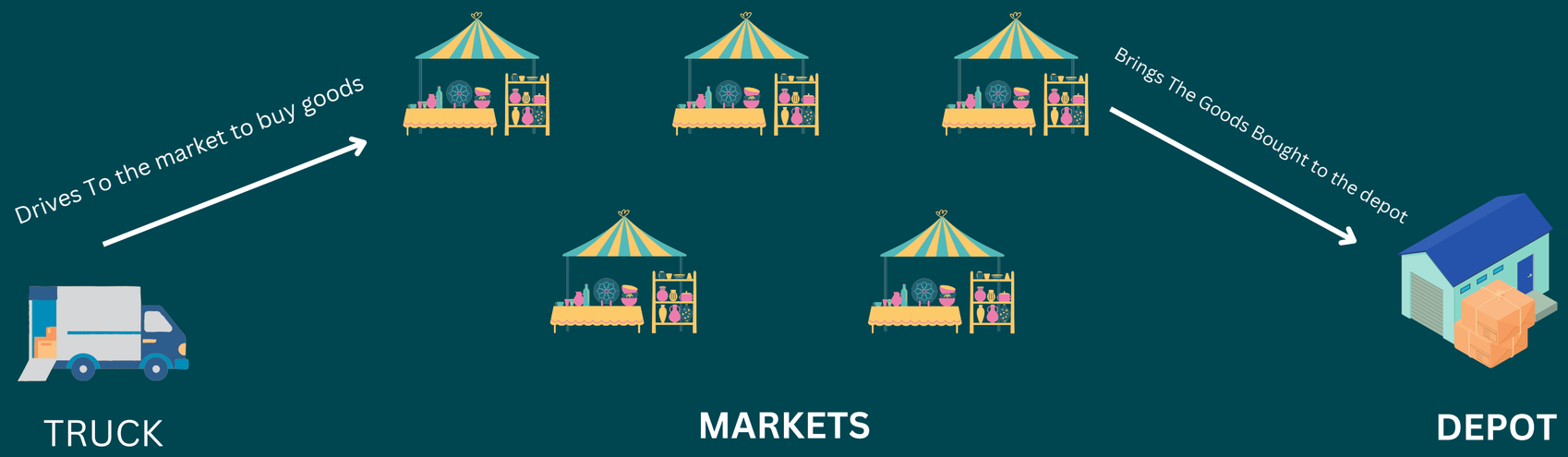
PDDL stands for "Planning Domain Definition Language." It is a formal language used for representing planning problems in artificial intelligence and automated planning.

PDDL consists of two main components: a domain description and a problem description. The domain description specifies the actions that can be taken in the planning problem, their preconditions, effects, and the types of objects that can be used.Describes the blueprint for a world. The problem description specifies the initial state of the world, the goal to be achieved, and any constraints or requirements that must be satisfied.

PROBLEM STATEMENT 1

BASIC PROBLEM

We have a set of products and a set of markets. Each market is provided with a limited amount of each product at a known price. The problem consists in selecting a subset of markets such that a given demand of each product can be purchased, minimizing the routing cost and the purchasing cost.



Source -https://ipc06.icaps-conference.org/deterministic/

PROBLEM STATEMENT 1

DOMAIN FILE

```
(define (domain TPP-Metric)
(:requirements :typing :fluents)
Show hierarchy
(:types depot market - place
    truck goods - locatable)

(:predicates (at ?t - truck ?p - place) )
```

The above piece of code specifies the domain file and 2 classes i.e place and locatable.

Here we will specify the subclasses ---depot,market,truck and the goods/product.

Predicates - The "at" predicate is used to check if the Truck is at the desired location or not

Source -https://ipc06.icaps-conference.org/deterministic/

PROBLEM STATEMENT 1

DOMAIN FILE

```
(:functions (on-sale ?g - goods ?m - market)
            (drive-cost ?p1 ?p2 - place)
            (price ?g - goods ?m - market)
            (bought ?g - goods)
            (request ?g - goods)
            (total-cost))
```

In PDDL 2.1 functions referred to as numeric fluents was introduced. A numeric fluent, similar to a predicate, is a variable which applies to zero or more objects and maintains a value throughout the duration of the plan.

It is declared with a name followed by the object type to which it applies. Default value=1.0 There are multiple operations possible on numeric fulents.(On Next Slide)

Source -https://ipc06.icaps-conference.org/deterministic/

PROBLEM STATEMENT 1

DOMAIN FILE

Increase

SUPPORT: HIGH USAGE: HIGH

```
(increase (battery-level ?r) 10)
```

An increase effect increases the value of a numeric variable by the given amount. It is possible to use another numeric variable as the increase value for example.

```
(increase (battery-level ?r) (charge-available - ?solarpanel))
```

Decrease

SUPPORT: HIGH USAGE: HIGH

```
(decrease (battery-level ?r) 10)
```

A decrease effect decreases the value of a numeric variable by the given amount. It is possible to use another numeric variable as the decrease value for example.

```
(decrease (battery-level ?r) (power-needed-for-work - ?task))
```

Assign

SUPPORT: HIGH USAGE: HIGH

```
(assign (battery-level ?r) 10)
```

Prefix Maths (Numeric Expressions)

SUPPORT: HIGH USAGE: HIGH

ADD

```
(+ (sample-capacity) (battery-capacity))
```

SUBTRACT

```
(- (sample-capacity) (battery-capacity))
```

DIVIDE

```
(/ (sample-capacity) (battery-capacity))
```

MULTIPLY

```
(* (sample-capacity) (battery-capacity))
```

Source -https://ipc06.icaps-conference.org/deterministic/

PROBLEM STATEMENT 1

DOMAIN FILE : DESCRIPTION OF EACH FUNCTION

```
(:functions (on-sale ?g - goods ?m - market)
            (drive-cost ?p1 ?p2 - place)
            (price ?g - goods ?m - market)
            (bought ?g - goods)
            (request ?g - goods)
            (total-cost))
```

1) on-sale specifies the number of goods on sale in the market.
2) Price-Specifies the price of good in that particular market.
3) bought- specifies the number of goods actually bought(till any particular state).
4) request-specifies the number of goods initially requested for.
5)total-cost-inculdes drive-cost and price of goods

Source -https://ipc06.icaps-conference.org/deterministic/

PROBLEM STATEMENT 1

DOMAIN FILE : ACTIONS

```
(:action drive
 :parameters (?t - truck ?from ?to - place)
 :precondition (and (at ?t ?from))
 :effect (and (not (at ?t ?from)) (at ?t ?to)
          (increase (total-cost) (drive-cost ?from ?to))))

(:action buy-allneeded
 :parameters (?t - truck ?g - goods ?m - market)
 :precondition (and (at ?t ?m) (> (on-sale ?g ?m) 0)
            (> (on-sale ?g ?m) (- (request ?g) (bought ?g))))
 :effect (and (decrease (on-sale ?g ?m) (- (request ?g) (bought ?g)))
          (increase (total-cost) (* (- (request ?g) (bought ?g))
                    (price ?g ?m)))
          (assign (bought ?g) (request ?g))))

(:action buy-all
 :parameters (?t - truck ?g - goods ?m - market)
 :precondition (and (at ?t ?m) (> (on-sale ?g ?m) 0)
            (<= (on-sale ?g ?m) (- (request ?g) (bought ?g))))
 :effect (and (assign (on-sale ?g ?m) 0)
          (increase (total-cost) (* (on-sale ?g ?m) (price ?g ?m)))
          (increase (bought ?g) (on-sale ?g ?m))))
)
```

An action represents a transformation from one state to another state in the world. An action is defined by its name, its parameters, preconditions, and effects.

1)Drive - Changes the location of truck and increments the the total cost by the drive cost .

2)Buy-allneeded - This action is executed when the current market has the number of goods we need.

3)Buy-all - This action is executed when the current market has less than the number of goods we need .In that case we buy all the goods from that market

Source -https://ipc06.icaps-conference.org/deterministic/

PROBLEM STATEMENT 1

PROBLEM FILE : OBJECTS AND INITIAL STATE

```
(define (problem pfile01)
(:domain TPP-Metric)
Show hierarchy
(:objects
    market1 market2 market3 market4 market5 - market
    depot0 - depot
    truck0 - truck
    goods0 - goods)
View
(:init
    (= (price goods0 market1) 17)
    (= (on-sale goods0 market1) 4)
    (= (price goods0 market2) 49)
    (= (on-sale goods0 market2) 9)
    (= (price goods0 market3) 33)
    (= (on-sale goods0 market3) 17)
    (= (price goods0 market4) 14)
    (= (on-sale goods0 market4) 9)
    (= (price goods0 market5) 40)
    (= (on-sale goods0 market5) 2)
    (at truck0 depot0)
    (= (drive-cost depot0 market1) 381.20)
    (= (drive-cost market1 depot0) 381.20)
    (= (drive-cost depot0 market2) 737.52)
```

The Problem set is defined on domain called TPP-Metric as .
Objects of the classes are created .

There are 5 Markets ,1 Depot ,1 Truck and 1 Type of product.

The Number of Goods in all 5 markets are specified in the (:init ...) (which specifies the initial state).

The Drive cost from any market to market and market to depot is also specified using drive-cost numeric fluent.
The bought goods and the total cost initially is zero and the requested goods are 39.
Initially the truct is at depot0

Source -https://ipc06.icaps-conference.org/deterministic/

PROBLEM STATEMENT 1

PROBLEM FILE : FOAL AND OPTIMIZATION

```
    (= (bought goods0) 0)
    (= (request goods0) 38)
    (= (total-cost) 0))

(:goal (and
    (>= (bought goods0) (request goods0))
    (at truck0 depot0)))

(:metric minimize (total-cost))
)
```

Our Goal Is To Start From depot0 and return back to depot0 after buying all the goods requested initially(more than requested is also acceptable).

And the planner has to minimize the total-cost.

A metric can be defined using some numeric operation on fluents, including using just a single fluent. We can choose to either minimize or maximize the value of the metric function.

Here we are minimizing the total cost of the entire plan.

Source -https://ipc06.icaps-conference.org/deterministic/

PROBLEM STATEMENT 1

TEMPORAL CONSTRAINTS

Now that we have defined the basic problem we will introduce some temporal constraints.

In PDDL the temporal constraints are of two types -

1)State Trajectory Constraints i.e Hard Contraints--- State trajectory constraints in PDDL (Planning Domain Definition Language) are used to impose constraints on the sequence of states that can be generated by a plan. These constraints specify conditions that must be satisfied by the states in a plan, as well as constraints on the order in which these states can occur.

2)Preferences (Soft Contraints) --allow a planner to take into account user-defined preferences when generating plans. Preferences can be used to indicate which plans are more desirable than others, based on criteria such as efficiency, cost, or user preferences.

Source -https://ipc06.icaps-conference.org/deterministic/

PROBLEM STATEMENT 1

TEMPORAL CONSTRAINTS

$$
\begin{array}{lll}
\langle (S_0,0),(S_1,t_1),...,(S_n,t_n)\rangle \models (\texttt{at end}\ \phi) & \text{iff} & S_n \models \phi \\
\langle (S_0,0),(S_1,t_1),...,(S_n,t_n)\rangle \models \phi & \text{iff} & S_n \models \phi \\
\langle (S_0,0),(S_1,t_1),...,(S_n,t_n)\rangle \models (\texttt{always}\ \phi) & \text{iff} & \forall i : 0 \le i \le n \cdot S_i \models \phi \\
\langle (S_0,0),(S_1,t_1),...,(S_n,t_n)\rangle \models (\texttt{sometime}\ \phi) & \text{iff} & \exists i : 0 \le i \le n \cdot S_j \models \phi \\
\langle (S_0,0),(S_1,t_1),...,(S_n,t_n)\rangle \models (\texttt{within}\ t\ \phi) & \text{iff} & \exists i : 0 \le i \le n \cdot S_i \models \phi \\
 & & \text{and } t_i \le t \\
\langle (S_0,0),(S_1,t_1),...,(S_n,t_n)\rangle \models (\texttt{at-most-once}\ \phi) & \text{iff} & \forall i : 0 \le i \le n \cdot \text{ if } S_i \models \phi \text{ then} \\
 & & \exists j : j \ge i \cdot \forall k : i \le k \le j \cdot S_k \models \phi \\
 & & \quad \text{and } \forall k : k > j \cdot S_k \models \neg\phi \\
\langle (S_0,0),(S_1,t_1),...,(S_n,t_n)\rangle \models & & \\
\quad (\texttt{sometime-after}\ \phi\ \psi) & \text{iff} & \forall i \cdot \text{if } S_i \models \phi \text{ then } \exists j : i \le j \le n \cdot S_j \models \psi \\
\langle (S_0,0),(S_1,t_1),...,(S_n,t_n)\rangle \models & & \\
\quad (\texttt{sometime-before}\ \phi\ \psi) & \text{iff} & \forall i \cdot \text{if } S_i \models \phi \text{ then } \exists j : 0 \le j < i \cdot S_j \models \psi \\
\langle (S_0,0),(S_1,t_1),...,(S_n,t_n)\rangle \models & & \\
\quad (\texttt{always-within}\ t\ \phi\ \psi) & \text{iff} & \forall i \cdot \text{if } S_i \models \phi \text{ then } \exists j : i \le j \le n \cdot S_j \models \psi \\
 & & \text{and } t_j - t_i \le t
\end{array}
$$

The hard constraints provided by PDDL ---

1)always
2)sometime
3)within
4)at-most-once
5)sometime-after
6)sometime-before
7)always-within

S0,S1,....Sn refers to the state of plane where Sn is the last state.
t0,t1....tn refers to the time at which the plan is in that state.

Source -http://www.cs.yale.edu/homes/dvm/papers/pddl-ipc5.pdf

PROBLEM STATEMENT 1

MODIFICATION TO PROBLEM STATEMENT 1

We will have multiple trucks and depots now.

Each action has a duration and the plan quality is a linear combination of total-time (makespan) and the total cost of traveling and purchasing.

The operator "buyall" has a "rebate" rate (if you buy the whole amount of a good that is sold at a market, then you have a discount).

2 New actions are Introduced — "load" truck and "unload" truck.

In our new problem we again will buy the goods from the markets but this time minimizing the total cost as well as the total time.

As there are more than 1 trucks ,we are adding a constraint that only one truck can be there in a market at one time and each truck must be used.All trucks should be empty at end of plan.

Source -https://ipc06.icaps-conference.org/deterministic/

PROBLEM STATEMENT 1

MODIFICATION TO PROBLEM STATEMENT 1: CONSTRAINTS

```
(:predicates (connected ?p1 ?p2 - place) 1👁
        (at ?t - truck ?p - place) )

(:functions (on-sale ?g - goods ?m - market) 6👁 1↘ 1≔
        (ready-to-load ?g -goods ?m - place) 3👁 2↗ 1≔ 1?
        (drive-cost ?p1 ?p2 - place) 1👁
        (drive-time ?p1 ?p2 - place) 1👁
        (loaded ?g - goods ?t - truck) 3👁 1↗ 1≔ 2?
        (stored ?g - goods) 1↗
        (price ?g - goods ?m - market) 2👁
        (bought ?g - goods) 4👁 1↗ 1≔
        (request ?g - goods) 5👁
        (total-cost) 3↗
        (rebate-rate ?m - market) 1👁)

(:constraints (and

        (forall (?m - market ?g - goods) (at end (= (ready-to-load ?g ?m) 0)))

        (forall (?t - truck ?g - goods) (at end (= (loaded ?g ?t) 0)))

        (forall (?m - market ?t1 ?t2 - truck)
           (always (imply (and (at ?t1 ?m) (at ?t2 ?m)) (= ?t1 ?t2))))

        (forall (?t - truck)
       (sometime (exists (?g - goods) (> (loaded ?g ?t) 0))))
))
```

Constraints:

1) At End There Should Be No Market Where There Are Godds that have to be Loaded in The Truck.

2)At End Of Plan No Truck shall have any goods in it.

3)In any market there should be only one truck at a given time.

4)Each Truck Must Be Loaded with Goods sometime during the Plan.

Source -https://ipc06.icaps-conference.org/deterministic/

PROBLEM STATEMENT 1

MODIFICATION TO PROBLEM STATEMENT 1: ACTIONS

```
(:durative-action drive
 :parameters (?t - truck ?from ?to - place)
 :duration (= ?duration (drive-time ?from ?to))
 :condition (and (at start (at ?t ?from)) (over all (connected ?from ?to)))
 :effect (and (at start (not (at ?t ?from))) (at end (at ?t ?to))
              (at end (increase (total-cost) (drive-cost ?from ?to)))))

(:durative-action load
 :parameters (?g - goods ?t - truck ?m - market)
 :duration (= ?duration (* 1.0 (ready-to-load ?g ?m)))
 :condition (and (at start (> (ready-to-load ?g ?m) 0)) (over all (at ?t ?m))
 :effect (and (at start (increase (loaded ?g ?t) (ready-to-load ?g ?m)))
              (at start (assign (ready-to-load ?g ?m) 0))))

(:durative-action unload
 :parameters (?t - truck ?g - goods ?d - depot)
 :duration (= ?duration (* 1.0 (loaded ?g ?t)))
 :condition (and (at start (> (loaded ?g ?t) 0)) (over all (at ?t ?d)))
 :effect (and (at start (increase (stored ?g) (loaded ?g ?t)))
              (at start (assign (loaded ?g ?t) 0))))
```

```
(:durative-action buy-allneeded
 :parameters (?t - truck ?g - goods ?m - market)
 :duration (= ?duration 1.0)
 :condition (and (at start (> (on-sale ?g ?m) (- (request ?g) (bought ?g))))
         (at start (> (on-sale ?g ?m) 0)) (over all (at ?t ?m)))
 :effect (and (at start (decrease (on-sale ?g ?m)
                        (- (request ?g) (bought ?g))))
         (at start (increase (ready-to-load ?g ?m)
                 (- (request ?g) (bought ?g))))
         (at start (increase (total-cost) (* (- (request ?g) (bought ?g))
                             (price ?g ?m))))
         (at start (assign (bought ?g) (request ?g)))))

(:durative-action buy-all
 :parameters (?t - truck ?g - goods ?m - market)
 :duration (= ?duration 1.0)
 :condition (and (at start (> (on-sale ?g ?m) 0)) (over all (at ?t ?m)))
 :effect (and (at start (assign (on-sale ?g ?m) 0))
          (at start (increase (ready-to-load ?g ?m) (on-sale ?g ?m)))
          (at start (increase (total-cost) (* (rebate-rate ?m)
                              (* (on-sale ?g ?m) (price ?g ?m)))))
          (at start (increase (bought ?g) (on-sale ?g ?m)))))
```

Source -https://ipc06.icaps-conference.org/deterministic/

PROBLEM STATEMENT 1

MODIFICATION TO PROBLEM STATEMENT 1: PROBLEM SET

1

```
(define (problem pfile01)
(:domain TPP-MetricTimeConstraints)
Show hierarchy
(:objects
    market1 - market
    depot1 - depot
    truck1 truck2 - truck
    goods1 goods2 goods3 - goods)

View
(:init
    (= (price goods1 market1) 49)
    (= (ready-to-load goods1 market1) 0)
    (= (on-sale goods1 market1) 17)
    (= (price goods2 market1) 4)
    (= (ready-to-load goods2 market1) 0)
    (= (on-sale goods2 market1) 5)
    (= (price goods3 market1) 50)
    (= (ready-to-load goods3 market1) 0)
    (= (on-sale goods3 market1) 15)
    (connected depot1 market1)
    (connected market1 depot1)
```

2

```
(:goal (and
    (>= (stored goods1) (request goods1))
    (>= (stored goods2) (request goods2))
    (>= (stored goods3) (request goods3))))

View
(:constraints (and))

(:metric minimize (+ (* 1.3 (total-cost))(total-time)))
)
```

The Goal is that the number of goods of each kind stored in the depot should be at least the number of that requested.

This piece of code also introduces prefence in optimization .

*1.3(totalcost) specifies that minimizing total-cost is preffered more than minimizing total time. (total-time) is equilvalent to (*1.0(total-time)).

Source -https://ipc06.icaps-conference.org/deterministic/

PROBLEM STATEMENT 1

PREFERENCES

Soft constraints are preferences that can be violated if necessary to achieve the goal. Soft constraints are often used to express preferences that are desirable, but not absolutely required for a plan to be considered valid.

For soft constraints, preferences are typically expressed as penalties that are added to the cost of a plan if the preference is violated. The planner will attempt to find a plan with the lowest possible cost, taking into account the penalties for violating preferences.

These penalties are imposed using :metic (is-violated)

Source -https://ipc06.icaps-conference.org/deterministic/

PROBLEM STATEMENT 1

MODIFIED PROBLEM 1: PREFERENCES

```
(:constraints (and
    (forall (?g - goods)
      (preference p5A (at end (forall (?m - market ?t - truck)
                    (and (= (ready-to-load ?g ?m) 0)
                     (= (loaded ?g ?t) 0))))))

    (forall (?t - truck ?g - goods)
      (preference p1A (at-most-once (> (loaded ?g ?t) 0))))

    (forall (?m - market ?t - truck)
      (preference p2A (at-most-once (at ?t ?m))))

    (forall (?m - market ?t1 ?t2 - truck)
      (preference p1B (always (imply (and (at ?t1 ?m) (at ?t2 ?m))
                    (= ?t1 ?t2)))))


    (forall (?t - truck)
      (preference p4A (always-within 2390.00 (> (loaded goods3 ?t) 0)
                (= (loaded goods3 ?t) 0))))

    (forall (?t - truck)
      (preference p4B (always-within 2390.00 (> (loaded goods2 ?t) 0)
                (= (loaded goods2 ?t) 0))))

    (forall (?t - truck)
      (preference p3A (sometime (exists (?g - goods) (> (loaded ?g ?t) 0)))))
    (preference p0A (sometime-before (> (stored goods2) 0)
              (>= (stored goods1) (request goods1))))))
```

1)P5A- At End Of Plan att all markets there should not be any good that has to beloaded.And all trucks must be empty

2)P1A- Each truck must be loaded at most once with each type of goods.

3)P2A- Each truck must be at a market at most once.

4)P1B- No Two Trucks are at a market at once .

5) P4A and P4B - Specified that the goods 2 and goods 3 must be loaded andunloaded within respected times.

6) P3A- All trucks must have goods in them sometime.

7)P0A - Before any amount of goods2 are stored in the depot,all goods1 requiredmust be stored.

Source -https://ipc06.icaps-conference.org/deterministic/

PROBLEM STATEMENT 1

MODIFIED PROBLEM 1:
PREFERENCES PENALTIES

```
(:metric minimize (+ (* 1 (is-violated p0A))
                     (* 2 (is-violated p1A))
                     (* 2 (is-violated p1B))
                     (* 3 (is-violated p2A))
                     (* 4 (is-violated p3A))
                     (* 5 (is-violated p4A))
                     (* 5 (is-violated p4B))
                     (* 6 (is-violated p5A))))
)
```

The Penalties are impoised in :metric .

PDDL3 extends the definition in PDDL2.1 to include a helper function is-violated which when given the name of preference, gives a count of the number of times the constraint has been violated.

Weight assigned to certain preferences as being more costly than others, using a * operator, we can multiply the number of violations by a violation cost, to give us total cost for violating that preference.

Here p5A has the most penalty followed by p4B, p4A, p3A, p2A, p1B, p1A.

Source -https://ipc06.icaps-conference.org/deterministic/

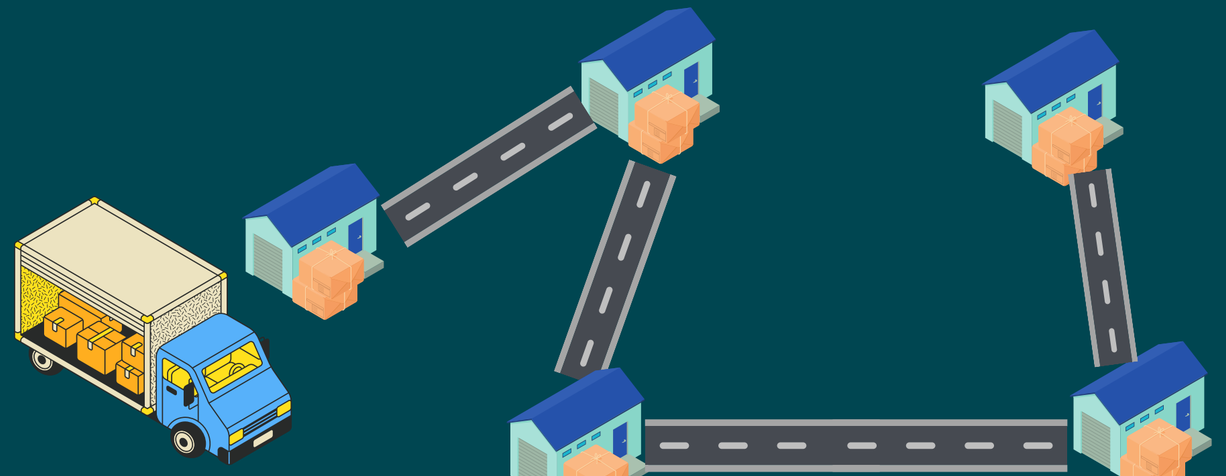
PROBLEM STATEMENT 2

PROBLEM STATEMENT

This is a logistics domain about moving packages between locations by trucks under certain constraints.

The loading space of each truck is organized by areas: a package can be (un)loaded onto an area of a truck only if the areas between the area under consideration and the truck door are free.
Moreover, some packages must be delivered within some deadlines.

Hence the planner needs to give the a plan containing sequential plan where each truck should go and when.



TRUCK:INITIALLY AT ONE OF THE LOCATION. 3 TRUCK AREAS (VISIBLE)

INTERCONNECTED LOCATIONS

Source -https://ipc06.icaps-conference.org/deterministic/

PROBLEM STATEMENT 2

DOMAIN FILE

```
(define (domain Trucks-ComplexPreferences)
(:requirements :typing :adl :durative-actions :fluents :constraints :preferences)
Show hierarchy
(:types truckarea location locatable - object
        truck package - locatable)

(:predicates (at ?x - locatable ?l - location)
             (in ?p - package ?t - truck ?a - truckarea) 1👁 1☑ 1☒
             (connected ?x ?y - location) 1👁
             (free ?a - truckarea ?t - truck) 5👁 1☑ 1☒
          (delivered ?p - package ?l - location) 1☑
          (closer ?a1 - truckarea ?a2 - truckarea) 4👁)

(:functions (drive-time ?from ?to - location) 1👁)
```

The types are truck area,land locations where the package have to be delivered.

Truck and packages are types which can be located using at predicate.

in predicate tells if package in given truck in given truckarea.connected tells if two location are connected and only then the truck can travel between the two locations.

Free tells if the given truck area in truck is free.

delivered is predicate which tells if given package is delivered to given location.

closer predicate tells if a1 truck area is closer to truct door than a2.

drive-time is the time for truck to travel between two locations.

Source -https://ipc06.icaps-conference.org/deterministic/

PROBLEM STATEMENT 2

DOMAIN FILE: ACTIONS

```
(:durative-action load
 :parameters (?p - package ?t - truck ?a1 - truckarea ?l - location)
 :duration (= ?duration 1)
 :condition (and (at start (at ?p ?l)) (at start (free ?a1 ?t))
        (at start (forall (?a2 - truckarea)
                  (imply (closer ?a2 ?a1) (free ?a2 ?t))))
            (over all (at ?t ?l))
        (over all (forall (?a2 - truckarea)
                  (imply (closer ?a2 ?a1) (free ?a2 ?t)))))
 :effect (and (at start (not (at ?p ?l))) (at start (not (free ?a1 ?t)))
    (at end (in ?p ?t ?a1))))

(:durative-action unload
 :parameters (?p - package ?t - truck ?a1 - truckarea ?l - location)
 :duration (= ?duration 1)
 :condition (and (at start (in ?p ?t ?a1))
        (at start (forall (?a2 - truckarea)
                  (imply (closer ?a2 ?a1) (free ?a2 ?t))))
            (over all (at ?t ?l))
        (over all (forall (?a2 - truckarea)
                  (imply (closer ?a2 ?a1) (free ?a2 ?t)))))
 :effect (and (at start (not (in ?p ?t ?a1))) (at end (free ?a1 ?t))
         (at end (at ?p ?l))))
```

Load- Truck starts filling from the truck area farthest from door .Hence a truck can store package only if truck area before it are empty at start of action and throughout the action. At end of Load .The Truck area a1 is not empty and package in is truck area.

Unload-Same Goes for unloading .For unloading from a truck area ,the truck atreas before the given truck area has to be free

Source -https://ipc06.icaps-conference.org/deterministic/

PROBLEM STATEMENT 2

DOMAIN FILE: ACTIONS

```
(:durative-action drive
 :parameters (?t - truck ?from ?to - location)
 :duration (= ?duration (drive-time ?from ?to))
 :condition (and (at start (at ?t ?from)) (over all (connected ?from ?to)))
 :effect (and (at start (not (at ?t ?from))) (at end (at ?t ?to))))

(:durative-action deliver
 :parameters (?p - package ?l - location)
 :duration (= ?duration 1)
 :condition (and (at start (at ?p ?l)) (over all (at ?p ?l)))
 :effect (and (at end (not (at ?p ?l))) (at end (delivered ?p ?l))))

)
```

Drive– Truck Drives from one location to another and the total time is increased with drive-time specified by the user.

Deliver – Marks The Package As Delivered

Source -https://ipc06.icaps-conference.org/deterministic/

PROBLEM STATEMENT 2

PROBLEM FILE

1

```
(free a3 truck1)
(closer a1 a2)
(closer a1 a3)
(closer a2 a3)
(at package1 l3)
(at package2 l3)
(at package3 l3)
(at package4 l2)
(at package5 l2)
(at package6 l2)
(at package7 l4)
(at package8 l4)
(at package9 l4)
(connected l1 l2)
(connected l1 l3)
(connected l1 l4)
(connected l2 l1)
(connected l2 l3)
(connected l2 l4)
(connected l3 l1)
(connected l3 l2)
(connected l3 l4)
(connected l4 l1)
(connected l4 l2)
(connected l4 l3)
(= (drive-time l1 l2) 545.6)
(= (drive-time l1 l3) 476.6)
(= (drive-time l1 l4) 270.1)
(= (drive-time l2 l1) 545.6)
```

Image On The Left Starts the problem file by specifying that there are 9 packages and 4 locations ,1 truck and 3 truck areas.

Image On The Right Specifies the Location of Trucks and packages initially .The order of truck areas in truck.The connectivity of locations with each other and the drive time between the connected locations

2

```
(define (problem truck-7)
(:domain Trucks-ComplexPreferences)
Show hierarchy
(:objects
    truck1 - truck
    package1 - package
    package2 - package
    package3 - package
    package4 - package
    package5 - package
    package6 - package
    package7 - package
    package8 - package
    package9 - package
    l1 - location
    l2 - location
    l3 - location
    l4 - location
    a1 - truckarea
    a2 - truckarea
    a3 - truckarea)

View
(:init
    (at truck1 l4)
    (free a1 truck1)
    (free a2 truck1)
    (free a3 truck1)
    (closer a1 a2)
    (closer a1 a3)
```

Source -https://ipc06.icaps-conference.org/deterministic/

PROBLEM STATEMENT 2

PROBLEM FILE

1

```
(free a3 truck1)
(closer a1 a2)
(closer a1 a3)
(closer a2 a3)
(at package1 l3)
(at package2 l3)
(at package3 l3)
(at package4 l2)
(at package5 l2)
(at package6 l2)
(at package7 l4)
(at package8 l4)
(at package9 l4)
(connected l1 l2)
(connected l1 l3)
(connected l1 l4)
(connected l2 l1)
(connected l2 l3)
(connected l2 l4)
(connected l3 l1)
(connected l3 l2)
(connected l3 l4)
(connected l4 l1)
(connected l4 l2)
(connected l4 l3)
(= (drive-time l1 l2) 545.6)
(= (drive-time l1 l3) 476.6)
(= (drive-time l1 l4) 270.1)
(= (drive-time l2 l1) 545.6)
```

Image On The Left Starts the problem file by specifying that there are 9 packages and 4 locations ,1 truck and 3 truck areas.

Image On The Right Specifies the Location of Trucks and packages initially .The order of truck areas in truck.The connectivity of locations with each other and the drive time between the connected locations

2

```
(define (problem truck-7)
(:domain Trucks-ComplexPreferences)
Show hierarchy
(:objects
    truck1 - truck
    package1 - package
    package2 - package
    package3 - package
    package4 - package
    package5 - package
    package6 - package
    package7 - package
    package8 - package
    package9 - package
    l1 - location
    l2 - location
    l3 - location
    l4 - location
    a1 - truckarea
    a2 - truckarea
    a3 - truckarea)

View
(:init
    (at truck1 l4)
    (free a1 truck1)
    (free a2 truck1)
    (free a3 truck1)
    (closer a1 a2)
    (closer a1 a3)
```

Source -https://ipc06.icaps-conference.org/deterministic/

PROBLEM STATEMENT 2

PROBLEM FILE: PREFERENCES

```
view
(:constraints (and

    (forall (?p - package ?t - truck)
       ((always (forall (?a - truckarea) (imply (in ?p ?t ?a) (closer ?a a2))))))
    (forall (?p - package ?t - truck)
       (preference p1B (always (forall (?a - truckarea) (imply (in ?p ?t ?a) (closer ?a a3))))))

    (preference p2A (sometime-before (delivered package5 l1)
                     (delivered package4 l1)))
    (preference p2B (sometime-before (delivered package6 l3)
                     (delivered package5 l1)))
    (preference p2C (sometime-before (delivered package9 l2)
                     (delivered package8 l3)))

    (preference p5A (within 1721.0 (delivered package3 l2)))
    (preference p5B (within 2845.4 (delivered package4 l1)))
    (preference p5C (within 2845.4 (delivered package5 l1)))
    (preference p5D (within 4555.8 (delivered package8 l3)))
    (preference p5E (within 4555.8 (delivered package9 l2)))

    (forall (?p - package)
       (preference p3A (at-most-once (exists (?t - truck ?a - truckarea)
        (in ?p ?t ?a)))))))
```

I1) P1B- Only truck areas a1 and a2 are used.

2)P2A - Package4 should be delivered before package5 at location 1.

3)P2B- Package5 should be delivered at location 1 before package6 at location 3.

4)P2C- Package8 should be delivered at location 3 before package9 at location 2.

5) P5A, P5B ,P5C ,P5D, P5E - Specifies the time before which the package should be delivered.

6) P3A- Each Package Should Be Once Once In A Truck.

Source -https://ipc06.icaps-conference.org/deterministic/

PROBLEM STATEMENT 2

PROBLEM FILE: PENALTIES

```
(:metric minimize (+
                (* 1 (is-violated p1B))
                (* 2 (is-violated p2A))
                (* 2 (is-violated p2B))
                (* 2 (is-violated p2C))
                (* 3 (is-violated p3A))
                (* 5 (is-violated p5A))
                (* 5 (is-violated p5B))
                (* 5 (is-violated p5C))
                (* 5 (is-violated p5D))
                (* 5 (is-violated p5E))))

)
```

The Penalties for violation of preferences are defined .

Source -https://ipc06.icaps-conference.org/deterministic/

RESOURCES

1) https://ipc06.icaps-conference.org/deterministic/

2)https://planning.wiki/ref/pddl3/problem

3)https://planning.wiki/ref/pddl3/domain

4)https://planning.wiki/ref

5)http://www.cs.yale.edu/homes/dvm/papers/pddl-ipc5.pdf

6)http://reports-archive.adm.cs.cmu.edu/anon/2004/CMU-CS-04-167.pdf

7)https://github.com/rpgoldman/pddl-tools

Thank you!