PDDL Workshop

Quick Refresher: PDDL Basics

Domain File

```
(define (domain mydomain)
  (:requirements …)
  (:types …) ;; optional
  (:predicates …)
  (:action …)
  …
)
```

Problem File

```
(define (problem myproblem)
  (:domain mydomain)
  (:objects …)
  (:init …)
  (:goal …)
)
```

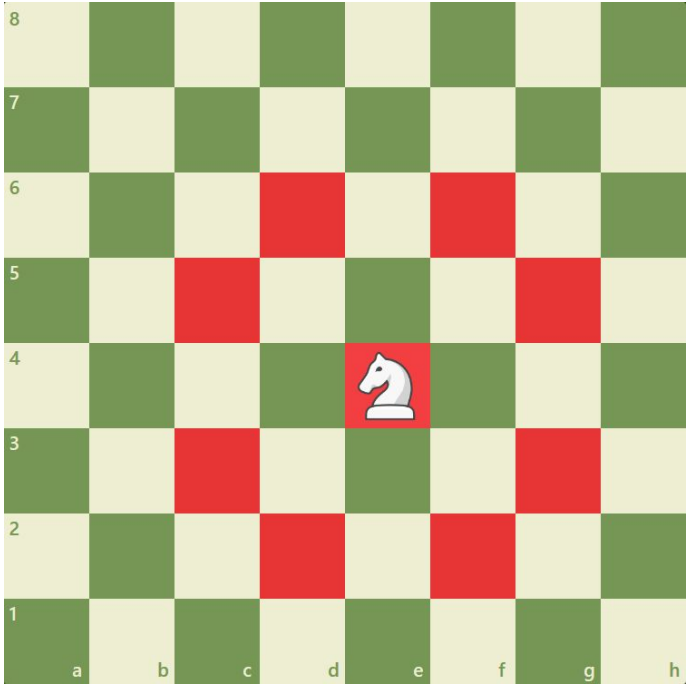
The Knight’s Tour Problem

Problem Overview:

- Chess puzzle
- **Goal**: Move knight to visit every square exactly once
- Special case of Hamiltonian Path

Knight’s Move:

- Moves in an L-shape
- “Jumps” over intermediate squares



Running Fast Downward

Basic command:

```
./fast-downward.py domain.pddl problem.pddl --search "<search config>"
```

Use `--search` followed by one of:

- `astar(...)` – A* search
- `lazy_greedy(...)` – Greedy best-first (lazy evaluation)
- `eager_greedy(...)` – Greedy best-first (eager evaluation)
- `lazy_wastar(...)` – Weighted A* search

Examples of supported heuristics:

- `blind()` – No heuristic
- `ff()` – Fast Forward heuristic (relaxed plan)
- `hmax()` – Max heuristic
- `add()` – Additive heuristic

Problem Description: Logistics Scenario

You are managing logistics between two cities: City-A and City-B.

Each city includes:

- Two locations (e.g., loc-a1, loc-a2 in City-A)
- One airport (airport-a in City-A, airport-b in City-B)

Vehicles available:

- 1 Truck in City-A, starting at loc-a1
- 1 Airplane, starting at airport-a

One package, initially at loc-a1, must be delivered to loc-b1.

Constraints:

- Trucks can only drive within a single city.
- Airplanes can fly between airports, connecting cities.

Exercise: Define a Typed PDDL Domain

Define a PDDL domain from scratch that:

- Uses typing (e.g., truck, airplane, location, etc.)
- Specifies relevant predicates (e.g., at, in, in-city)
- Models actions such as:
  - Loading/unloading packages into trucks or airplanes
  - Driving trucks within a city
  - Flying airplanes between airports

Bonus Challenge (optional): Consider the hierarchy of types (e.g., vehicles as a subtype of objects)

Linehaul Logistics – Problem Description

Context:

- A distributor must deliver goods to customers in one day
- Trucks start and end at a central depot
- Each truck can only make one round trip

Fleet – Trucks differ in:

- Refrigeration (some can carry chilled goods)
- Capacity (how many goods they can carry)

Goods:

- Chilled goods: must go on refrigerated trucks
- Ambient goods: can go on any truck

Customer Demands – Each location requests:

- X units of chilled goods
- Y units of ambient goods

Goal:

- Deliver all requested goods and return trucks to the depot

Linehaul Logistics – PDDL Modeling Tips

Object Types:

- truck, refrigerated_truck (subtype), location, quantity

Quantities as Objects:

- Represent integers as symbolic objects: n0, n1, ..., n40
- Define plus1(n2, n3) to encode: n3 = n2 + 1

✅ Key Predicates

- `(at ?t ?l)` – truck is at location
- `(free_capacity ?t ?q)` – truck has q free capacity
- `(demand_chilled_goods ?l ?q)` / `(demand_ambient_goods ?l ?q)`

Actions:

- drive – move a truck between locations
- deliver_chilled – deliver 1 chilled unit (refrigerated trucks only)
- deliver_ambient – deliver 1 ambient unit

Linehaul Logistics – Problem Instance

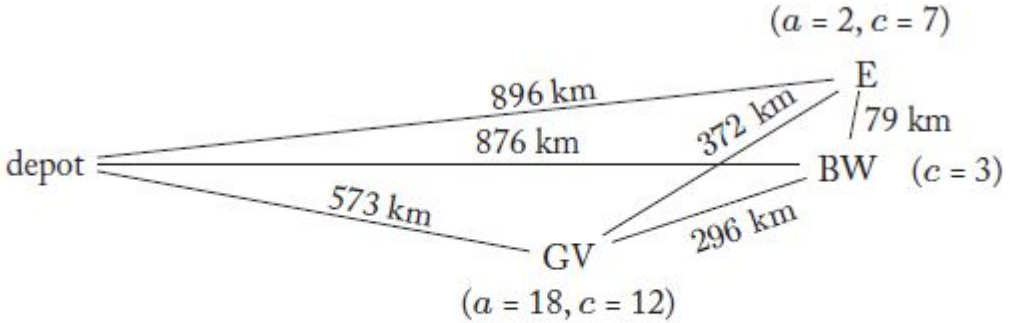


Figure 2.2: A small example instance of the Linehaul problem. The numbers next to customer locations show their demand for ambient and chilled goods.

(Image source: An Introduction to the Planning Domain Definition Language, AIPS-2000 Tutorial, McDermott et al.)

Introducing Action Costs in PDDL

- Purpose: Model and minimize total execution cost in a planning problem
- Based on numeric functions (fluents) – limited form for action costs only

3 Key Steps to Add Action Costs

Declare total-cost as a function in the domain:

```
(:functions (total-cost))
```

Modify actions to include cost effects using increase:

```
(increase (total-cost) <expression>)
```

Specify cost minimization in the problem:

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

Notes about costs

- Use `:action-costs` in `:requirements`
- Use prefix notation: `(* a b)`, `(+ a b)`, `etc.`
- Static function values initialized in `:init` with `(= ...)`
- `total-cost` must be initialized to 0

Understanding `either` in PDDL

- `either` defines a **union type**: a variable can belong to **multiple types**

- Example:
  - `?x - (either type1 type2 ...)`
  - `(at ?x - (either truck airplane package) ?loc - location)`

- Avoids defining separate predicates like:
  `(at-truck ...), (at-airplane ...), (at-package ...)`

Not all planners support `either` (check your planner's PDDL version)