Hospital Logistics: Robot Delivery System

This presentation outlines a system for autonomous robots to deliver medicines within a multi-floor hospital, focusing on different PDDL implementations.

Scenario Overview

Hospital Structure

- 4 floors, 4 patient rooms each (16 patients total)
- 4 autonomous robots (one per floor)
- Elevator for inter-floor medicine transport

Key Locations

- Ground floor storage room (all medicines start here)
- Ground floor robot room (robot departure/return point)

The objective is to deliver all 16 medicines to the correct patients and ensure robots return to their room.

Boolean Implementation (PDDL 1.2)

Core Predicates

- **(robot_load_x ?r - robot)**: Assigns medicine slots (simulates capacity).
- **(connected_floor ?f1 - floor ?f2 - floor)**: Defines logical floor sequence.

Planner Used: LAMA

Used for satisficing (good results, less time).

Robot Limitations

- **can_use**, **can_move**: Robots use elevator only when full, avoiding redundant steps.

Boolean Priority: Adding Urgency

To simulate real-world urgency, we introduced medicine priority.

Medicine Priority

Identifies urgent medicines.

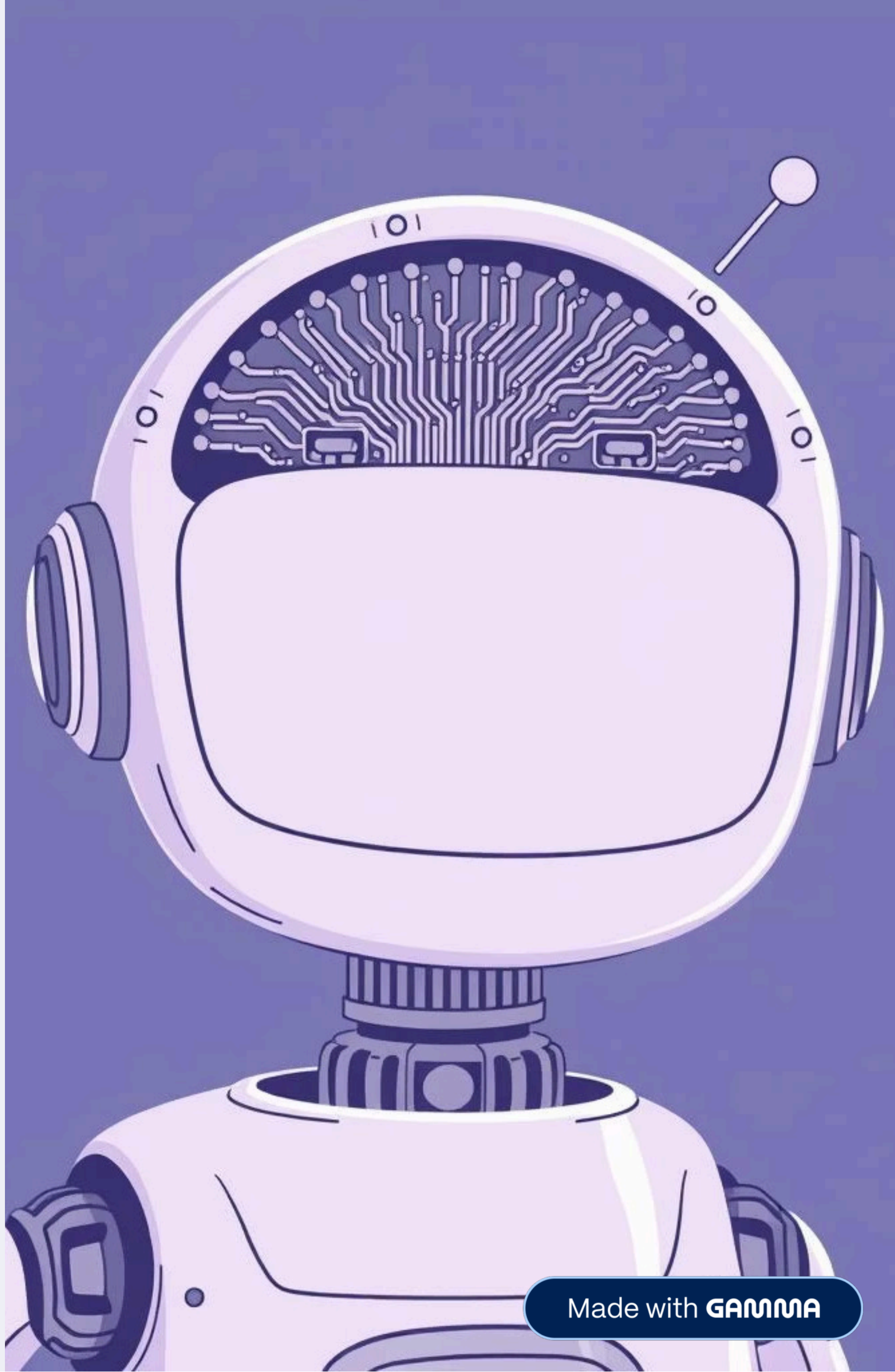
Robot Priority Status

Indicates if a robot carries a priority medicine.

Priority Handled

Confirms delivery of urgent medicine.

Specialized actions (**load_priority_med**, **deliver_priority_med**) override standard ones for urgent drugs, ensuring priority deliveries are handled first.



Numeric Implementation (PDDL 2.1)

PDDL 2.1 introduces numeric fluents for more expressive modeling.

Robot Capacity

Replaced **(robot_load_x ?r - robot)** predicates with **(load-count ?r - robot)**, simplifying tracking of medicines carried.

Simplified Control

Replaced **(can_move ?r - robot)** and **(can_use ?r - robot)** with **(robot_ready_for_delivery ?r - robot)**.

Elevator Capacity

Handled with **(elevator-load ?e - elevator)** numeric fluent, no longer requiring multiple predicates.

Planner Used: ENHSP

- Supports numeric fluents.
- Efficiently handles domains without temporal or priority features.
- Generated correct plans with fast grounding and planning times.
- Demonstrated solid performance and compatibility for PDDL 2.1.

This approach makes the PDDL simpler, cleaner, and easier to read.

Temporal Implementation (PDDL 2.1)

Key Durative Actions

- **load_med**: Takes **1 second**
- **start_move**: At least **5 seconds**
- **enter_elevator**: **10 seconds** duration

Planner: LPG++

Generates optimal plan in **2.39 seconds**

Efficiently handles temporal constraints while minimizing total delivery time

Core Fluents

- **robot_load**: Tracks medicine capacity
- **elevator_load**: Monitors elevator usage
- **total-cost**: Measures operational efficiency

Speaker Notes: PDDL 2.1 temporal implementation introduces explicit time durations. Example:

```
(:durative-action load_med
  :parameters (?r - robot ?m - medicine)
  :duration (= ?duration 1)
  :condition (at start (and (at ?r storage) (at ?m storage)))
  :effect (and (at end (carrying ?r ?m)) (at end (not (at ?m storage)))))
```

Battery-Aware Extension

New Predicates

- **low_battery** when charge falls **below 25%**
- **in_charging_room** tracking robot location
- **recharged** indicating full battery status

Energy Actions

- **navigate_to_charger** finds nearest charging station
- **recharge_battery** restores **100%** capacity

The hybrid discrete-continuous model significantly increases operational robustness by preventing battery depletion during critical medicine deliveries.

Base vs Battery-Aware: Quick Comparison

Concurrency & Cost

Base model handles parallel actions efficiently but lacks energy awareness, potentially leading to unrealistic scheduling in real-world deployment.

Energy Realism

Battery extension introduces critical constraints that mirror real-world limitations, preventing scenarios where robots would realistically fail due to power depletion.

Planner Impact

Base model: **2.39s** planning time

Battery variant: **~15%** slower

Small performance trade-off for significantly enhanced operational reliability.

Key Takeaway: The battery-aware extension substantially increases model realism with only modest computational overhead, creating a more robust and deployable solution for real-world hospital environments.

The battery extension represents a crucial step toward real-world deployment, addressing a critical gap in the base model. The 15% performance penalty is negligible compared to the operational safety gained by preventing mid-delivery battery failures.

Planner Spotlight: LPG++

Heuristic Forward-Chaining

Supports PDDL 2.1 durative actions, numeric fluents, and **cost optimization**

Advanced Search Techniques

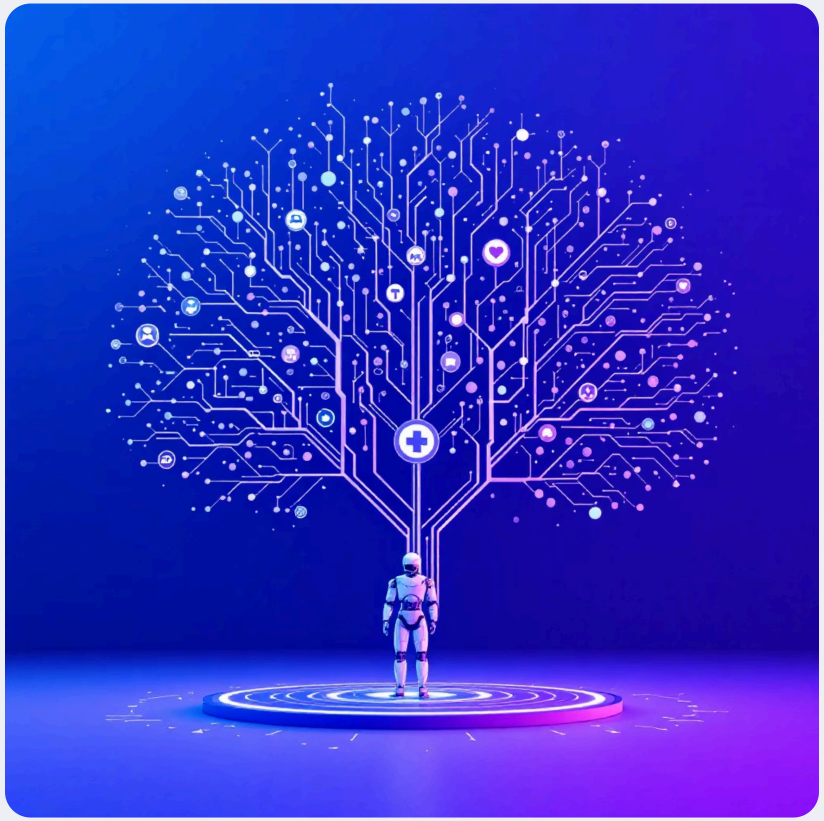
Weighted relaxed-plan heuristic with anytime incremental search

Concurrency Optimization

Exploits parallel actions to reduce makespan while maintaining **total-cost** minimization

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

// Generated optimal plan in 2.39s
```



Command-line example :

```
./lpg++ -o domain.pddl -f problem.pddl -out plan.txt
```

PDDL+ Implementation

Processes & Events

Extends PDDL 2.1, integrating dynamic processes and discrete events for richer state modeling.

PDDL+ is used for (mixed discrete-continuous) planning

Continuous Dynamics

Enables modeling of continuous changes and unforeseen exogenous events in real-time environments.

Hybrid Modeling

Ideal for complex domains that blend discrete actions with continuous processes, like robot movement.

Realistic Robotics

Applied to simulate precise, real-world robot navigation within the hospital logistics scenario.

PDDL+ Framework: Technical Overview

1

Framework Basis

Extends PDDL 2.1 with processes and events for continuous and hybrid dynamics.

2

Predicates

robot_at, **medicine_at**, **elevator_at**, **robot_moving**, **elevator_moving**, **robot_available**, **elevator_available**.

3

Numeric Fluents:

- **robot_load** (0-4 medicines), elevator_load (0-4 robots).
- **move_progress** (robot movement, 0.0-1.0), elevator_progress (elevator movement).
- **robot_speed** (0.2, 5 time units/move), elevator_speed (0.1, 10 time units/floor).
- **total-cost** (minimized metric).

4

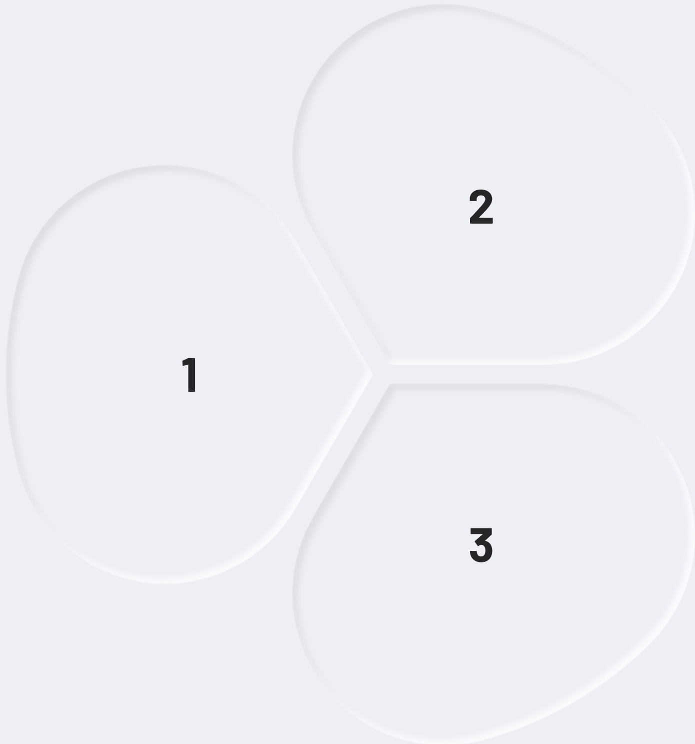
Key Actions:

- **start_move, start_elevator_move**: Initiate continuous movement.
- **load_med, deliver_med**: Manage medicine transport.
- **enter_elevator, exit_elevator**: Handle elevator usage.

Continuous Dynamics and Safety in PDDL+

Continuous Movement:

- **Processes**: move_progress (robots) and elevator_move_progress (elevator) increment progress based on speed and time.
- Robot movement: 5 time units (move_progress += **robot_speed * time**).
- Elevator movement: 10 time units (elevator_progress += **elevator_speed *time**).
- **Events**: move_complete, elevator_move_complete trigger when progress reaches 1.0, updating locations and availability.



Safety Features:

- **Overload Events**: robot_overload_warning and elevator_overload_warning cap robot_load and elevator_load at 4.
- **Availability Checks**: robot_available and elevator_available prevent conflicts during movement or elevator use.

Workflow:

1. Robots load up to **4** medicines in storage.
2. Robots use elevator to reach assigned floors.
3. Robots deliver medicines to patient rooms.
4. Robots return to robot room via elevator.

ROBOT MOVEMENT - ACTIONS, PROCESSES, EVENTS

Robot Movement - action, process, event

```
(:action start_move
  :parameters (?r - robot ?from - location ?to - location)
  :precondition (and
    (robot_at ?r ?from)
    (connected ?from ?to)
    (not (exists (?e - elevator) (robot_in_elevator ?r ?e)))
    (robot_available ?r)) ;robot is not busy
  :effect (and
    (not (robot_available ?r))    ;robot is busy(at work)
    (robot_moving ?r ?from ?to)
    (assign (move_progress ?r) 0.0)
    (increase (total-cost) 0.5)))

(:process move_progress
  :parameters (?r - robot ?from - location ?to - location)
  :precondition (and
    (robot_moving ?r ?from ?to))
  :effect (and
    (increase (move_progress ?r) (* #t (robot_speed ?r)))))

(:event move_complete
  :parameters (?r - robot ?from - location ?to - location)
  :precondition (and
    (robot_moving ?r ?from ?to)
    (>= (move_progress ?r) 1.0))
  :effect (and
    (not (robot_moving ?r ?from ?to))
    (not (robot_at ?r ?from))
    (robot_at ?r ?to)
    (robot_available ?r)
    (assign (move_progress ?r) 0.0)
    (increase (total-cost) 0.5)))
```

Single-Robot Medicine Delivery with PDDL+ and Battery Management

Hospital Logistics Overview

Objective: Deliver 8 medicines to patients across 3 floors and return the robot to its base, minimizing cost while managing battery levels.

Hospital Structure

Three floors: Ground (floor0), Floor 1, and Floor 2. Each floor has four designated patient rooms.

Key Locations

Storage (floor0) holds 8 medicines. The Robot Room (floor0) serves as the robot's base and charging station.

Robot & Elevator

A single robot (robot1) services all floors. One elevator is available, used exclusively by the robot.

Core Goal

Efficiently deliver all 8 medicines to their specific patient rooms, then return robot to base with sufficient battery.

Battery-Aware PDDL+ Modeling

1. **PDDL+ Framework:**

- Extends PDDL 2.1 with **processes** and **events** for continuous dynamics.
- **Key Predicates**: robot_at, medicine_at, robot_moving, elevator_moving, robot_available, charging, low_battery, battery_critical..

2. **Numeric Fluents:**

- robot_load (0-4 medicines), total-cost (minimized).
- **move_progress** (speed 0.5, 2 time units/move), **elevator_progress** (speed 0.2, 5 time units/move).
- **battery_level** (0-100), **battery_drain_rate** (5.0), **battery_charge_rate** (5.0).
- low_battery_threshold (40), critical_battery_threshold (20).

3. **Battery Management:**

- **Actions**: start_charging (in robot_room, cost 0.1), stop_charging (at battery ≥ 90).
- **Process**: battery_charging increases battery_level (5.0/time unit).
- **Events**: battery_low_warning (≤ 40), battery_critical_warning (≤ 20), battery_recovered (> 40).
- **Constraints**: Actions require minimum battery_level (e.g., 1.5 for load, 1.0 for move) and not battery_critical.

4. **Workflow:**

1. Robot loads medicines in storage (up to 4).
2. Uses elevator to reach floor1 or floor2.
3. Delivers medicines to patient rooms.
4. Returns to robot_room, charging if battery ≤ 40.

Why PDDL+ with Battery Excels

Benefits:

- **Realism**: Continuous movement (2-unit robot moves, 5-unit elevator moves) and battery dynamics reflect real-world constraints.
- **Reliability**: Proactive charging prevents critical battery states, ensuring task completion.

Planner: ENHSP:

Extended Numeric Heuristic Search Planner: Handles PDDL+ processes and events.

- Handles PDDL+ processes (**move_progress**, **battery_charging**) and events (**battery_low_warning**).
- Generates plans with ~37.1 cost, modeling 2-unit robot moves and 5-unit elevator moves.
- **Limitation**: Limited planner compatibility due to PDDL+ complexity.

Command: java -jar enhsp.jar -o pddl-added/domain.pddl -f pddl-added/problem.pddl -planner sat-hmax

Thank you !