Compiling Numeric Planning to Classical Planning

Bachelor Thesis

Eda Erkek
Artificial Intelligence – Department of Mathematics and Computer Science
Prof. Dr. Malte Helmert

July 29, 2026

Motivation

- Classical planning:
  - Well studied, fast solvers
  - Can only represent facts
- Numeric planning:
  - Can represent real quantities (fuel, cost)
  - Solving techniques far less mature

Goal with this thesis → getting the benefits of both

Background

$$\Pi_n = \langle V_p, V_n, \beta, A, I, G \rangle$$

- Finite domain Variables: Classic & Numerical
- Bounds of numeric variables
- Actions with preconditions & effects
- Initial & Goal state

Background

$\Pi_n = \langle V_p, V_n, \beta, A, I, G \rangle$

Example: 2d Drone

- $V_n = \{battery, x, y\}$
- $\beta(battery) = (0, 15)$
- $\beta(x) = (0, 3), \beta(y) = (0, 3)$
- $I = (battery = 15, x = 0, y = 0)$
- $G_n = (x = 3, y = 3)$

Actions:

- increase_x, decrease_x, (additive)
- increase_y, decrease_y, (additive)
- recharge (assignment)

Background

$$\Pi_p = \langle V'_p, A', I', G' \rangle$$

- Finite domain Variables: only classic
- Actions with classical preconditions & effects
- Initial & Goal state

Translation

$$\Pi_n = \langle V_p, V_n, \beta, A, I, G \rangle \quad \Rightarrow \quad \Pi_p = \langle V_p', A', I', G' \rangle$$

- Numeric variables → Classical finite domain variables
- Grounding: for each numeric state, separate action
- Initial state: each numeric value → classic variable
- Goal: numeric → only exact value comparison

Benchmark Domains

IPC2023 Dataset Repository:

- **Drone**: Drone in 3d grid
  - visiting points, assignment effect for recharging
  - **Actions**: moving on axes, recharge
- **Expedition**: Sled shuttling supplies between waypoints
  - multi-variable preconditions, large bound (up to 1000)
  - **Actions**: moving between waypoints, retrieve/store supplies

Soundness Verification

2 different checks:

- Unreachable goal test (wrong bounds, unsatisfiable preconditions)
- Manual replay of solution

Drone Example

action: increase_x **current state** :n0=25, n1=0 **cost**:1

action: visit x1y0z0 **current state**: n0=24, n1=1, n2=0, n3=0 **cost**:1

action: increase_y **current state**: n0=23, n2=0 **cost**:1

action: visit x1y1z0 **current state**: n0=22, n1=1, n2=1, n3=0 **cost**:1

...

action: decrease_y **current state**: n0=4_n2=1 **cost**:1

action: visit x0y0z0 **current state**: n0=3, n1=0, n2=0, n3=0 **cost**:1

action: recharge **current state**: n1=0, n2=0, n3=0 **cost**:1

action: increase_x **current state**: n0=25, n1=0 **cost**:1

..

action: decrease_y **current state**: n0=2, n2=2 **cost**:1

action: decrease_y **current state**: n0=1, n2=1 **cost**:1

plan ends here

Goal: visiting points, end at 0,0,0

Variables:
n0 = battery
n1 = x value
n2 = y value
n3 = z value

Results

- Solved problems in both benchmark domains

  → equality-goal, additive/assignment-effect
- Bigger problems blow up fast drone6:

  → 609 operators, but ~13GB / 354M states

Limitations, Future Work

- **Bounds**: manual (drone's could be automated, but domain-specific)
- **Goals**: exact-value only, not comparisons
- → new classical variable goal_n as extra effect to grounded action
- **Cost**: multi-variable / preconditions actions → grounded actions blow-up → memory limit

- **Motivation** → Numeric planning problems can be solved correctly using mature classical tools