Artificial Intelligence Foundation – JC3001

Lecture 33: Hierarchical Planning - III

Prof. Aladdin Ayesh (aladdin.ayesh@abdn.ac.uk)

Dr. Binod Bhattarai (binod.bhattarai@abdn.ac.uk)

Dr. Gideon Ogunniye, (g.ogunniye@abdn.ac.uk)

October 2025

Course Progression

- Part 1: Introduction
  1. Introduction to AI ✓
  2. Agents ✓
- Part 2: Problem-solving
  1. Search 1: Uninformed Search ✓
  2. Search 2: Heuristic Search ✓
  3. Search 3: Local Search ✓
  4. Search 4: Adversarial Search ✓
- Part 3: Reasoning and Uncertainty
  1. Reasoning 1: Constraint Satisfaction ✓
  2. Reasoning 2: Logic and Inference ✓
  3. Probabilistic Reasoning 1: BNs ✓
  4. Probabilistic Reasoning 2: HMMs ✓
- Part 4: Planning
  1. Planning 1: Intro and Formalism ✓
  2. Planning 2: Algorithms & Heuristics ✓
  3. **Planning 3: Hierarchical Planning**
  4. Planning 4: Stochastic Planning
- Part 5: Learning
  1. Learning 1: Intro to ML
  2. Learning 2: Regression
  3. Learning 3: Neural Networks
  4. Learning 4: Reinforcement Learning
- Part 6: Conclusion
  1. Ethical Issues in AI
  2. Conclusions and Discussion

Objectives

- Control Knowledge in Planning
- Hierarchical Planning

Outline

- ▶ Hierarchical Planning
- ▶ Hierarchical Goal Networks
- ▶ Conclusion

- HTN planning is even more general
- Can have constraints associated with tasks and methods
  Conditions that must be true before, during, or afterwards

Domain-Configurable Planners Compared to Classical Planners

- Disadvantage: writing a knowledge base can be more complicated than just writing classical operators
- Advantage: can encode “recipes” as collections of methods and operators
  - Express problems and knowledge that cannot be expressed in classical planning
  - Specify standard ways of solving problems
    - Otherwise, the planning system would have to derive these again and again from “first principles,” every time it solves a problem
    - Can speed up planning by many orders of magnitude (e.g., polynomial time versus exponential time)

Domain Knowledge

- Domain knowledge requires extensive domain engineering
- Domain knowledge is not always available
- It is possible to generate “unsolvable” methods
- What can you do when you do not have this domain knowledge?

[1] MAGNAGUAGNO, M. C.; MENEGUZZI, F. Method Composition through Operator Pattern Identification. KEPS@ICAPS 2017.

Domain Knowledge

- Domain knowledge requires extensive domain engineering
- Domain knowledge is not always available
- It is possible to generate "unsolvable" methods
- What can you do when you do not have this domain knowledge?
  - Use the naive conversion above
    (this is bad, performance is worse than classical planning)

[1] MAGNAGUAGNO, M. C.; MENEGUZZI, F. Method Composition through Operator Pattern Identification. KEPS@ICAPS 2017.

Domain Knowledge

- Domain knowledge requires extensive domain engineering
- Domain knowledge is not always available
- It is possible to generate "unsolvable" methods
- What can you do when you do not have this domain knowledge?
  - Use the naive conversion above
    (this is bad, performance is worse than classical planning)
  - Automatically create domain knowledge using regularities in the operator descriptions
    (object of much research from our own group, much more efficient[1])

[1] MAGNAGUAGNO, M. C.; MENEGUZZI, F. Method Composition through Operator Pattern Identification. KEPS@ICAPS 2017.

Comparative Summary

Classical Planning

- Initial state
- Goal state
- Use actions/operators
- Optimality is search/heuristic dependent
- Anarchical/flat description
  - Easier to make/maintain
  - Harder to solve

Hierarchical Planning

- Initial state
- Task list
- Use operators and methods
- Optimality is description dependent
- Hierarchical description
  - Harder to make/maintain
  - Easier to solve

Outline

2 Hierarchical Goal Networks

- Hierarchical Planning
- Hierarchical Goal Networks
- Conclusion

- States, operators, **and goals**
  The same as in classical planning
- Methods $m = (head(m), pre(m), sub(m))$
  - $head(m)$: like the head of a planning operator
  - $pre(m)$: like the precondition of a planning operator
  - $sub(m)$: $sub(m) = \langle g_1, \ldots, g_k \rangle$, where each g_i is a goal formula (a set of literals)
  - $post(m)$: implicit postcondition g_k if $sub(m)$ is non-empty, $pre(m)$ otherwise

```
Method for using truck ?t to move crate ?o
        from location ?l1 to location ?l2 in city ?c:

  Head: (move-within-city ?o ?t ?l1 ?l2 ?c)
   Pre: ((obj-at ?o ?l1) (in-city ?l1 ?c)
         (in-city ?l2 ?c) (truck ?t ?c) (truck-at ?t ?l3))
   Sub: ((truck-at ?t ?l1) (in-truck ?o ?t)
         (truck-at ?t ?l2) (obj-at ?o ?l2)))

Method for using airplane ?plane to move crate ?o
        from airport ?a1 to airport ?a2:

  Head: (move-between-airports ?o ?plane ?a1 ?a2)
   Pre: ((obj-at ?o ?a1) (airport ?a1)
         (airport ?a2) (airplane ?plane))
   Sub: ((airplane-at ?plane ?a1) (in-airplane ?o ?plane)
         (airplane-at ?plane ?a2) (obj-at ?o ?a2)))

Method for moving ?o from location ?l1 in city ?c1
        to location ?l2 in city ?c2, via airports ?a1 and ?a2:

  Head: (move-between-cities ?o ?l1 ?c1 ?l2 ?c2 ?a1 ?a2)
   Pre: ((obj-at ?o ?l1) (in-city ?l1 ?c1) (in-city ?l2 ?c2)
         (different ?c1 ?c2) (airport ?a1) (airport ?a2)
         (in-city ?a1 ?c1) (in-city ?a2 ?c2))
   Sub: ((obj-at ?o ?a1) (obj-at ?o ?a2) (obj-at ?o ?l2)))
```

Algorithm for HGN Planning

- Much like the TFD algorithm
  - Process goals in network sequentially
  - Choose decompositions applicable to current state
  - Execute actions when they come up for decomposition
- Key difference, objective is to achieve goals

procedure GDP($D, s, G\pi$)
    if G is *empty* **then return** π
    $g \leftarrow$ the first goal formula in G
    if $s \models g$ **then**
        remove g from G and **return** GDP(D, s, G, π)
    $g \leftarrow \{$ actions and methods instances relevant for g and applicable to $s\}$
    if $U = \emptyset$ **then return** failure
    nondeterministically choose $s \in U$
    if u is an *action* **then**
        append u to π and set $s \leftarrow \gamma(s, u)$
    else
        insert $sub(u)$ at the front of G
    return GDP(D, s, G, π)

Heuristics for HGN Planning

- GDP can be modified to use heuristics in the decomposition
  - Instead of nondeterministic search, sort decomposers
- Example heuristic
  - Modified RPG used in FF
  - $l_{PG}(p)$ is the level where p is reachable
  - Estimates de distance between: the first level in which G is true; and the first level in which s is true

$$h_{s,G}(u) = \begin{cases} 1 + \max\limits_{p \in G} l_{PG}(p) - \max\limits_{p \in \gamma(s,u)} l_{PG}(p), & \text{if } u \text{ is an action,} \\ \max\limits_{p \in G \cup sub(u)} l_{PG}(p) - \max\limits_{p \in s} l_{PG}(p), & \text{if } u \text{ is a method.} \end{cases}$$

procedure GDP(D, s, G, π)
- **if** G is *empty* **then return** π
- $g \leftarrow$ the first goal formula in G
- **if** $s \models g$ **then**
  - remove g from G and **return** GDP(D, s, G, π)
- $g \leftarrow \{$ actions and methods instances relevant for g and applicable to $s\}$
- **if** $U = \emptyset$ **then return** failure
- **sort** U with $h(u), \forall u \in U$
- **for all** $u \in U$ **do**
  - **if** u is an *action* **then**
    - append u to π
    - remove g from G
    - $s \leftarrow \gamma(s, u)$
  - **else**
    - push $sub(u)$ into G
  - $\pi \leftarrow$ GDP(D, s, G, π)
  - **if** $\pi \neq$ failure **then return** π
- **return** failure

HGN Planning Current Work

- GoDel - mixed forward search and decomposition
- HOGL - heuristic mixed forward search and decomposition (AAAI-17)

- Hierarchical Planning
- Hierarchical Goal Networks
- Conclusion

- **SHOP2:** implementation of PFD-like algorithm + generalizations
  - Won one of the top four awards in the AIPS-2002 Planning Competition
  - Freeware, open source
  - Implementation available at: `http://www.cs.umd.edu/projects/shop`

- **SHOP3:** Latest incarnation of the SHOP software
  - Currently being maintained by Robert Goldman (SIFT)
  - Updated to latest versions of LISP
  - Freeware, open source
  - Implementation available at: `https://github.com/shop-planner/shop3`

- **PyHop:** implementation of TFD-like algorithm in Python
  - Domain is encoded as a number of Python functions representing methods and operators
  - Freeware, open source
  - Implementation available at: `https://bitbucket.org/dananau/pyhop`

PyHop's Syntax


```
def travel_by_foot(state,a,x,y):
    if state.dist[x][y] <= 4:
        return [('walk',a,x,y)]
    return False

def travel_by_taxi(state,a,x,y):
    if state.cash[a] >= 1.5 + 0.5 * state.
     dist[x][y]:
        return [('call_taxi',a,x), ('
     ride_taxi',a,x,y),
                ('pay_driver',a,x,y)]
    return False


declare_methods('travel', travel_by_foot,
     travel_by_taxi)
```

Travel by foot from x to y
Task: travel from x to y
Precond: agent is at x, distance to y is ≤ 4 km
Subtasks: walk from x to y
Travel by taxi from x to y
Task: travel from x to y
Precond: agent is at x, agent has money $\geq 1.5 + \frac{1}{2}$ distance(x,y)
Subtasks: call taxi to x, ride taxi from x to y, pay driver

- Panda: implementation of both HTN and plan-space planners
  - Domain encoded in HDDL (a PDDL-based HTN language)
  - Easier to define partially ordered methods
  - Implementation available at:
    `https://www.uni-ulm.de/en/in/ki/research/software/panda/`

Science activity planning (https://github.com/nasa/europa/wiki/What-Is-Europa)

- Airborne observatory SOFIA
- Remote Agent Experiment (RAX)
- Mars Exploration Rovers (MER) - 15 years of continuous operation
- Phoenix Mars Mission
- Mars Science Laboratory (MSL)

Power Systems Control of the International Space Station through the Solar Array Constraint Engine (SACE)

- HTN Planner in Ruby, partially inspired by PyHop
- Developed in my research group:
  `https://github.com/pucrs-automated-planning/HyperTensioN`
- Various parsers to convert JSHOP and HDDL into the planner
- **Winner of the 2020 International Planning Competition (IPC)**
  `http://gki.informatik.uni-freiburg.de/competition/results-fixed.pdf`

Hierarchical Planning Summary

- Heuristics and Control Strategies
- Domain Knowledge
- Hierarchical Task Network Planning
  - Comparison with classical planning
- Hierarchical Goal Networks

Any Questions.