Practical Planning

Minh Do
ERA/PARC
Automated Planning Approaches

- Domain-Independent Planning:
  - Systematic search: progression, regression, plan-space  \(\leftarrow\) CS221
  - Graphplan:  \(\leftarrow\) CS221
  - Local search
  - Compilation approach

- Domain-Specific Planning:
  - HTN Planning  \(\leftarrow\) Lecture 16
  - Temporal Logic-based Planning
Constraint Satisfaction Problem: Map-Coloring [Lecture 14]

Variables $WA, NT, Q, NSW, V, SA, T$

Domains $D_i = \{\text{red, green, blue}\}$

Constraints: adjacent regions must have different colors
  e.g., $WA \neq NT$ (if the language allows this), or
  $(WA, NT) \in \{(\text{red, green}), (\text{red, blue}), (\text{green, red}), (\text{green, blue}), \ldots\}$
Constraint Satisfaction Problem: Map-Coloring [Lecture 14]

Solutions are assignments satisfying all constraints, e.g.,
\{WA = red, NT = green, Q = red, NSW = green, V = red, SA = blue, T = green\}
Varieties of CSPs [Lecture 14]

Discrete variables
- finite domains; size $d \Rightarrow O(d^n)$ complete assignments
  - e.g., Boolean CSPs, incl. Boolean satisfiability (NP-complete)
- infinite domains (integers, strings, etc.)
  - e.g., job scheduling, variables are start/end days for each job
  - need a constraint language, e.g., $StartJob_1 + 5 \leq StartJob_3$
  - linear constraints solvable, nonlinear undecidable

Continuous variables
- e.g., start/end times for Hubble Telescope observations
- linear constraints solvable in poly time by LP methods

Common Problem Structure Setup:
- What are the variables?
- What are the domain of each variable?
- What are the constraints between variables?
Outline

Compilation Approaches for Planning
- Satisfiability (SAT) (binary CSP – Lecture 14)
- Constraint Satisfaction Problem (CSP)
- Integer Linear Programming (ILP) (infinite domain, continuous variable – Lecture 14)
- Answer Set Programming (ASP) (Lecture 11)
  → Define Variables/Domain/Constraints

Planning Applications
Compiling Planning Problem to a Combinatorial Substrate

Planning Problem:

Initial State → Actions ↔ Goal State

preconditions effects

Compilation in CSP/SAT/ASP:
What are the variables?
What are the domain of each variable?
What are the constraints between variables?
set the plan length bound $k$
(is there a plan of length $k$?)

encode the $k$-step plan in a combinatorial substrate

off-the-shelf solver

Motivation:

$X$ times improvements in state-of-the-art solver
(SAT/CSP competition)

automatic

$X$ times improvements in planner performance

yes

de code the plan

no

Increase $k$
Planning as Satisfiability (SAT)

- Most popular compilation/translation based approach

- SAT-based planners regularly compete in IPCs (despite the domination of search-based planners)

- A paper on new SAT-based planning technique won AAAI-2010 Best Paper Award
Satisfiability Problem

- **Variables:** True/False
- **Constraints:** AND, OR, NOT
  - a ∧ b
  - a ∨ b
  - ~a

- **Problem Representation:**
  - ((a ∨ ((~b ∧ c) ∨ ~d) ∧ e) …….) …..

  - Conjunction Normal Form (CNF):
    - ~a ∧ (b ∨ c) ∧ (d ∨ ~e)
  - Disjunction Normal Form (DNF):
    - ~a ∨ (b ∧ c) ∨ (d ∧ ~e)

- **Problem:** find a complete T/F variable assignment satisfying all constraints
Example

Initial State

Goal State

Fluents:

OnTable(A), OnTable(B)
On(A,B), On(B,A)
Clear(A), Clear(B)
Holding(A), Holding(B)
HandEmpy

Actions:

Pickup(A), Pickup(B)
Stack(A,B), Stack(B,A)
Putdown(A), Putdown(B)
Naïve SAT Encoding

Use fluents describe what’s true at each “level”
Variables

- $l_i$ to denote the fluent of literal $l$ in level $i$
  $l_2 = On(A,B,2)$

- $a_i$ to denote if action $a$ is the $i^{th}$ step of the plan
  $a_1 = Pickup(A,1)$
Constraints

- **Formula describing the initial state:**
  \[ \land \{ l_i \mid i \in s_0 \} \land \{ \neg l_i \mid i \in L - s_0 \} \]

- **Formula describing the goal state:**
  \[ \land \{ l_i \mid i \in g^+ \} \land \{ \neg l_i \mid i \in g^- \} \]

- **Formulas describing the preconditions and effects of actions:**
  For every action \( a_i \) in \( A \), formulas describing what changes \( a \) would make if it were the \( i \)’th step of the plan:
  - \( a_i \Rightarrow \land \{ p_i \mid p \in \text{Precond}(a) \} \land \{ e_{i-1} \mid e \in \text{Effects}(a) \} \)

- **Formulas describing Complete exclusion:**
  - For all actions \( a \) and \( b \), formulas saying they cannot occur at the same time
    \( \neg a_i \lor \neg b_i \)
  - this guarantees there can be only one action at a time

- **Formulas providing a solution to the Frame Problem**
  Explanatory frame axioms: \( l_i \rightarrow l_{i-1} \lor (\lor a_{i-1}: l \text{ is } a \text{'s effect}) \)
SAT Encoding: Problems

(1) Huge number of variables and constraints

(2) Solve large number of SAT encodings (k = 1, 2, …, n)
Each level consists of:

- **Literals** = all those that *could* be true at that time step, depending upon the actions executed at preceding time steps.
- **Actions** = all those actions that *could* have their preconditions satisfied at that time step, depending on which of the literals actually hold.
Mutex Propagation: facts and actions that cannot happen together
• Reduce the number of “reachable” actions & facts at each level
• Better estimation of plan length
• Mutex constraints help solver during constraint propagation
Step 1: build the planning graph until all goals appear non-mutex
Step 2: backward relevant analysis to remove irrelevant actions/facts
Step 3: encode the remaining graph as SAT

(Note: There are other more recent techniques to improve SAT-based planners (e.g., long-distance mutex, transition-based encoding)
**GP-CSP: Planning Graph + (discrete) CSP**

**CSP Variable:** State variable $s$ with domain = actions in previous level supporting $s$

**Constraints:**
- **Activation:** $\text{he}_3 = \text{St-A-B}_2 \rightarrow (\text{h-A}_2 \neq \text{NULL}) \land (\text{cl-A}_2 \neq \text{NULL})$
- **Mutex:** $\text{NOT} (\text{he}_3 = \text{St-A-B}_2 \land \text{on-B-A}_3 = \text{St-B-A}_2)$
- **Goals:** $\text{on-A-B}_3 \neq \text{NULL}$
(discrete) CSP vs. SAT

- CSP encoding is generally (much) smaller
- CSP solver is more expressive → easier to extend the planner to more expressive planning problem
  → But people have found creative ways to use SAT for planning with continuous resources and preferences
- More suitable for newer “multi-value” planning representation

- SAT solvers advance much faster
  – SAT planner is generally faster (now)
Integer Linear Programming

- SAT T/F variables → ILP 0/1 variables
- SAT constraints → ILP constraints
  \[ a \lor b \lor c \rightarrow a + b + c \geq 1 \]
  \[ a \land b \land c \rightarrow a + b + c = 3 \]

- Advantages:
  – Some constraints are much more compactly represented in ILP:
    » Only one action in a given level (XOR): \( a_1 + a_2 + \ldots + a_n = 1 \)
  – Can represent continuous variable naturally (e.g., robot battery level)
  – Advance ILP encoding use bi-level graph with variables beyond 0/1 → even more compact representation
Answer Set Planning [Lecture 11]

- Planning problem \( \langle A, I, G \rangle \)
  - \( A \) – a set of action descriptions
  - \( I \) – initial state
  - \( G \) – goal state

→ Logic program \( P(A, I, G) \) – Three different sets of rules:
  - Representing \( A \) and \( I \)
  - Representing \( G \)
  - Generating action occurrences

- Each answer set of \( P(A, I, G) \) corresponds to a trajectory achieving \( G \) and vice versa.
\[ \{ (A, I, G) \} \]

- **Action theory:**
  
  \[ \text{drive causes at(airport) if at(home)} \]
  
  \[ \text{drive executable_if hasCar} \]

- **Initially:**
  
  \[ \text{at(home), hasCar} \]

- **Goal:**
  
  \[ \text{at(airport)} \]

**Logic program** \( P(A, I, G) \)

\[
\begin{align*}
\text{holds(at(airport), T+1)} & \leftarrow \text{holds(at(home), T), holds(hasCar, T), occ(drive, T)} \\
\text{holds(F, T+1)} & \leftarrow \text{holds(F, T), not holds(\neg F, T+1)}. \\
1 \{ \text{occ(A, T) : action(A)} \} 1 & \leftarrow \text{time(T)} \\
\text{holds(at(home), 0).} & \\
\text{holds(hasCar, 0).} & \\
& \leftarrow \text{not holds(at(airport), plan\_length)}. \quad \text{Goal}
\end{align*}
\]
Advantage of ASP Compilation

- Shown to be easy to extend to more complex planning problems:
  - Uncertainty: conformant, contingent planning (with sensing actions)
  - Planning with qualitative preferences
Outline

- Compilation approach for Planning
  - Satisfiability (SAT) *(binary CSP – Lecture 14)*
  - CSP
  - MILP *(infinite domain, continuous variable – Lecture 14)*
  - ASP

- Planning applications
ICAPS 2011 Stats

- 12/47 accepted papers are applications
- 13 papers accepted at Scheduling and Planning Applications Workshop (SPARK)
- 17/20 system demos are applications
### Dimensions of Planning

<table>
<thead>
<tr>
<th></th>
<th>Simple</th>
<th>Complex</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>State Scope</strong></td>
<td>Finite</td>
<td>Non-finite</td>
</tr>
<tr>
<td><strong>Action Determinism</strong></td>
<td>Deterministic</td>
<td>Nondeterministic</td>
</tr>
<tr>
<td><strong>Action Duration</strong></td>
<td>Instantaneous</td>
<td>Durative</td>
</tr>
<tr>
<td><strong>World Observability</strong></td>
<td>Full</td>
<td>Partial</td>
</tr>
<tr>
<td><strong>World Dynamics</strong></td>
<td>Static</td>
<td>Exogenous events</td>
</tr>
<tr>
<td><strong>Goal Attainment</strong></td>
<td>Full</td>
<td>Partial</td>
</tr>
<tr>
<td><strong>Time</strong></td>
<td>No time points</td>
<td>Rich model of time</td>
</tr>
</tbody>
</table>

**Classical Planning Problem**
Classical Planning: Application

- Simplest form of planning:
  - More complex planning domains can be “relaxed” and solved by classical planner

- Applications:
  - Games: iceblock, sokoban, freecell
  - Diagnosis as planning
  - Greenhouse logistic
  - Genome rearrangement
  - Analyzing computer network vulnerability
  - Military training
Genome Rearrangement

Genome edit operation = planning action

Problem: finding minimum edit distance = finding shortest plan
→ help build the most plausible “evolutionary tree”
Analyzing Computer Network Security

Plan = adversary course of action to exploit a given system vulnerability

• Find all plans (attack tree/graph)
• Find way to “fix/prevent” attack plans
(metric FF planner was used ← Lecture 16)
**SHOGUN: Military Training**

**Domain Characteristics:** temporal, non-deterministic action, partial observability

→ Clever “relaxation” scheme to map to use classical planner
- **Stochastic action effects:** ignore all possible effects except most likely one
- **Partial observability:** “optimistic sensing” assume no blue force found for all red sensing actions
Application: Temporal Planning/Scheduling

- Multi-modal logistic
- Mars rovers
- Satellite coordination
- Robotic task planning

Applications at ERA/PARC:
- Multi-engine printer
- LCD manufacturing plan
- Automated warehouse
Find paper routes that run arbitrary printer configurations at maximum productivity

220 pages/minute

180 pages/minute
PARC-IHI: LCD Manufacturing Plant

Maximum productivity
Real-time: avoid failures, maintenance, congestion
Deadlock Avoidance: Example 1
Multi-Modal Transportation

600 graph nodes, 179K edges, 300 trucks, 300 containers, 300 transportation routes, 42 train segments, 148 ship segments

Planning time requirement: < 2 hrs

[Borrajo et al., 2009]
Mapgen: Mixed-Initiative Planning & Scheduling for the Mars Exploration Rover Mission

Using EUROPA Planner developed at NASA Ames (also used to control: underwater autonomous vehicle at MBARI and robots)

ASPEN@JPL: spacecraft operation, mission design, Antenna utilization, coordinated multiple rover planning
Application: Planning with Uncertainty

- Power Supply Restoration (PSR)
- Workflow/Web-service composition
- Interactive Storytelling
- Robot Information Processing & Sensing
Supply restoration on faulty power distribution system:
(1) Localize the faulty line; (2) reconfigure the network
→ A natural contingency planning problem
Interactive Storytelling

Automatically generate story based on user’s preferences
Scheduling Applications

- Airport
- Satellite coordination
- Shipping port
- Timetabling
- Google calendar improvements
Conclusion