Current Trends in Automated Planning

Dana S. Nau
Homework

- Planning in games
What is a plan?
[a representation] of future behavior ... usually a set of actions, with temporal and other constraints on them, for execution by some agent or agents. - Austin Tate

[MIT Encyclopedia of the Cognitive Sciences, 1999]
Generating Plans of Action

- Computer programs to aid human planners
  - Project management (consumer software)
  - Plan storage and retrieval
    » e.g., *variant process planning* in manufacturing
  - Automatic schedule generation
    » various OR and AI techniques
- For some problems, we would like generate plans (or pieces of plans) automatically
  - Much more difficult
  - Automated-planning research is starting to pay off
What are planners useful for?
Space Exploration

- Autonomous planning, scheduling, control
  - NASA: JPL and Ames
- Remote Agent Experiment (RAX)
  - Deep Space 1
- Mars Exploration Rover (MER)
Manufacturing

- Sheet-metal bending machines - Amada Corporation
  - Software to plan the sequence of bends
    [Gupta and Bourne, *J. Manufacturing Sci. and Engr.*, 1999]
Games

- *Bridge Baron* - Great Game Products
  - 1997 world champion of computer bridge
    [Smith, Nau, and Throop, *AI Magazine*, 1998]
  - 2004: 2nd place

West—♦A2
(North—♠Q)
(North—♠3)

East—♠J
South—♠5
South—♠Q

**Games**

- *Bridge Baron* - Great Game Products
  - 1997 world champion of computer bridge
    [Smith, Nau, and Throop, *AI Magazine*, 1998]
  - 2004: 2nd place
Outline

- Conceptual model for planning
- Example domain
- Types of planners
  - Domain-dependent
  - Domain-independent
  - Configurable
- Classical planning assumptions
Conceptual Model

1. Environment

State transition system

\[ \Sigma = (S, A, E, \gamma) \]
State Transition System

Σ = (S, A, E, γ)

- S = {states}
- A = {actions}
- E = {exogenous events}
- State-transition function
  γ: S × (A ∪ E) → 2^S

- S = {s_0, ..., s_5}
- A = {move1, move2, put, take, load, unload}
- E = {}
- γ: see the arrows

The Dock Worker Robots (DWR) domain
Complete observability: 
\[ h(s) = s \]

Observation function 
\[ h: S \rightarrow O \]

Conceptual Model

2. Controller

State transition system 
\[ \Sigma = (S,A,E,\gamma) \]
Conceptual Model

3. Planner’s Input

State transition system

\[ \Sigma = (S,A,E,\gamma) \]

Depends on whether planning is online or offline

Planning problem

Initial state

Objectives

Execution status

Planner

Description of \( \Sigma \)

Controller

Planner’s Input

Observation function

\( h: S \rightarrow O \)

Given observation \( o \) in \( O \), produces action \( a \) in \( A \)

System \( \Sigma \)

Events

Observations

Plans

Actions
Planning Problem

Description of $\Sigma$
Initial state or set of states
Initial state $= s_0$
Objective
Goal state, set of goal states, set of tasks, “trajectory” of states, objective function, …
Goal state $= s_5$

The Dock Worker Robots (DWR) domain
Conceptual Model

4. Planner’s Output

- **Planning problem**
  - Depends on whether planning is online or offline

- **Observation function**
  - $h(s) = s$

- **State transition system**
  - $\Sigma = (S, A, E, \gamma)$

- **Description of $\Sigma$**
  - Given observation $o$ in $O$, produces action $a$ in $A$

- **Instructions to the controller**
  - Depends on whether planning is online or offline

- **Controller**
  - Plans
  - Actions

- **System $\Sigma$**
  - Observations
  - Execution status

- **Planner**
  - Initial state
  - Objectives

Dana Nau: Lecture slides for *Automated Planning*
Licensed under the Creative Commons Attribution-NonCommercial-ShareAlike License: http://creativecommons.org/licenses/by-nc-sa/2.0/
Plans

**Classical plan**: a sequence of actions

\[ \langle \text{take}, \text{move1}, \text{load}, \text{move2} \rangle \]

**Policy**: partial function from \( S \) into \( A \)

\[ \{(s_0, \text{take}), (s_1, \text{move1}), (s_3, \text{load}), (s_4, \text{move2})\} \]

The Dock Worker Robots (DWR) domain
A running example: Dock Worker Robots

- **Locations**: l₁, l₂, …
- **Containers**: c₁, c₂, …
  - can be stacked in piles, loaded onto robots, or held by cranes
- **Piles**: p₁, p₂, …
  - fixed areas where containers are stacked
  - pallet at the bottom of each pile
- **Robot carts**: r₁, r₂, …
  - can move to adjacent locations
  - carry at most one container
- **Cranes**: k₁, k₂, …
  - each belongs to a single location
  - move containers between piles and robots
  - if there is a pile at a location, there must also be a crane there
A running example: Dock Worker Robots

- Fixed relations: same in all states
  - adjacent($l,l'$)
  - attached($p,l$)
  - belong($k,l$)

- Dynamic relations: differ from one state to another
  - occupied($l$)
  - at($r,l$)
  - loaded($r,c$)
  - unloaded($r$)
  - holding($k,c$)
  - empty($k$)
  - in($c,p$)
  - on($c,c'$)
  - top($c,p$)
  - top(pallet,$p$)

- Actions:
  - take($c,k,p$)
  - put($c,k,p$)
  - load($r,c,k$)
  - unload($r$)
  - move($r,l,l'$)
Planning Versus Scheduling

- What is the difference between these two types of problems?
Planning Versus Scheduling

• Scheduling
  - Decide when and how to perform a given set of actions
    - Time constraints
    - Resource constraints
    - Objective functions
  - Typically NP-complete

• Planning
  - Decide what actions to use to achieve some set of objectives
  - Can be much worse than NP-complete; worst case is undecidable
Three Main Types of Planners

1. Domain-specific
2. Domain-independent
3. Configurable
Types of Planners: 1. Domain-Specific

- Made or tuned for a specific domain
- Won’t work well (if at all) in any other domain
- Most successful real-world planning systems work this way
Types of Planners
2. Domain-Independent

- In principle, a domain-independent planner works in any planning domain
- Uses no domain-specific knowledge except the definitions of the basic actions
- Representation written in a STRIPS or PDDL-type formalism
- In practice,
  - Not feasible to develop domain-independent planners that work in every possible domain
- Make simplifying assumptions to restrict the set of domains
  - Classical planning
  - Historical focus of most automated-planning research
Restrictive Assumptions

- **A0**: Finite system
  - finitely many states, actions, and events
- **A1**: Fully observable
  - the controller always knows what state $\Sigma$ is in
- **A2**: Deterministic
  - each action or event has only one possible outcome
- **A3**: Static
  - No exogenous events: no changes except those performed by the controller
Restrictive Assumptions

A4: Attainment goals
- a set of goal states $S_g$

A5: Sequential plans
- a plan is a linearly ordered sequence of actions $(a_1, a_2, \ldots, a_n)$

A6: Implicit time
- no time durations
- linear sequence of instantaneous states

A7: Off-line planning
- planner doesn’t know the execution status
Classical Planning

- Classical planning requires all eight restrictive assumptions
  - Offline generation of action sequences for a deterministic, static, finite system, with complete knowledge, attainment goals, and implicit time
- Reduces to the following problem:
  - Given \((\Sigma, s_0, S_g)\)
  - Find a sequence of actions \((a_1, a_2, \ldots, a_n)\) that produces a sequence of state transitions \((s_1, s_2, \ldots, s_n)\) such that \(s_n\) is in \(S_g\).
- This is just path-searching in a graph
  - Nodes = states
  - Edges = actions
- Is this trivial?
Classical Planning

- Generalize the earlier example:
  - Five locations, three robot carts, 100 containers, three piles
    » Then there are $10^{277}$ states
  - Number of particles in the universe is only about $10^{87}$
    » The example is more than $10^{190}$ times as large!

- Automated-planning research has been heavily dominated by classical planning
  - Dozens (hundreds?) of different algorithms
Plan-Space Planning (UCPOP)

- Decompose sets of goals into the individual goals
- Plan for them separately
  - Bookkeeping info to detect and resolve interactions

For classical planning, not used very much any more
RAX and MER use temporal-planning extensions of it
Planning Graphs

- Relaxed problem
  [Blum & Furst, 1995]
- Apply all applicable actions at once
- Next “level” contains all the effects of all of those actions
For \( n = 1, 2, \ldots \)
- Make planning graph of \( n \) levels (*polynomial time*)
- State-space search *within the planning graph*

Graphplan’s many children
- IPP, CGP, DGP, LGP, PGP, SGP, TGP, …
**Heuristic Search**

- Can we do an A*-style heuristic search?
- For many years, nobody could come up with a good $h$ function
  - But planning graphs make it feasible
    - Can extract $h$ from the planning graph

- Problem: A* quickly runs out of memory
  - So do a greedy search

- Greedy search can get trapped in local minima
  - Greedy search plus local search at local minima

- HSP [Bonet & Geffner]
- FastForward [Hoffmann]
Translation to Other Domains

- Translate the planning problem or the planning graph into another kind of problem for which there are efficient solvers
  - Find a solution to that problem
  - Translate the solution back into a plan

- Satisfiability solvers, especially those that use local search
  - Satplan and Blackbox [Kautz & Selman]

- Integer programming solvers such as Cplex
  - [Vossen et al.]
Types of Planners

- Domain-independent planners are quite slow compared with domain-specific planners
  - Blocks world in linear time [Slaney and Thiébaux, *A.I.*, 2001]
  - Can get analogous results in many other domains
- But we don’t want to write a whole new planner for every domain!
- Configurable planners
  - Domain-independent planning engine
  - Input includes info about how to solve problems in the domain
    » Hierarchical Task Network (HTN) planning
    » Planning with control formulas
HTN Planning

- Problem reduction
  - Tasks (activities) rather than goals
  - Methods to decompose tasks into subtasks
  - Enforce constraints, backtrack if necessary
- Real-world applications
- Noah, Nonlin, O-Plan, SIPE, SIPE-2, SHOP, SHOP2
Planning with Control Formulas

At each state $s_i$ we have a control formula $f_i$ in temporal logic:

$$ontable(x) \land \neg \exists y:GOAL(on(x,y)) \Rightarrow \Box(\neg holding(x))$$

“never pick up $x$ from table unless $x$ needs to be on another block”

For each successor of $s$, derive a control formula using logical progression.

Prune any successor state in which the progressed formula is false.

- TLPlan [Bacchus & Kabanza]
- TALplanner [Kvarnstrom & Doherty]
Which type of planner is better?
Comparisons

- Domain-specific planner
  - Write an entire computer program - lots of work
  - Lots of domain-specific performance improvements

- Domain-independent planner
  - Just give it the basic actions - not much effort
  - Not very efficient
Comparisons

- A domain-specific planner only works in one domain

- **In principle**, configurable and domain-independent planners should both be able to work in any domain

- **In practice**, configurable planners work in a larger variety of domains
  - Partly due to efficiency
  - Partly due to expressive power
**Example**

- The planning competitions
  - All of them included domain-independent planners
- In addition, AIPS 2000 and *IPC* 2002 included configurable planners
- The configurable planners
  - Solved the most problems
  - Solved them the fastest
  - Usually found better solutions
  - Worked in many non-classical planning domains that were beyond the scope of the domain-independent planners
But Wait …

- The 2004 International Planning Competition contained *no* configurable planners.
  - Why not?
But Wait …

- The 2004 International Planning Competition contained no configurable planners.
  - Why not?
- Hard to enter them in the competition
  - Must write all the domain knowledge yourself
  - Too much trouble except to make a point
  - The authors of TLPlan, TALplanner, and SHOP2 felt they had already made their point
- Why not provide the domain knowledge?
  - Drew McDermott proposed this at ICAPS-05
  - There was a surprising amount of resistance
  - Cultural bias against the idea
Cultural Bias

- Most automated-planning researchers feel that using domain knowledge is “cheating”
- Researchers in other fields have trouble comprehending this
  - Operations research, control theory, engineering, …
  - Why would anyone not want to use the knowledge they have about a problem they’re trying to solve?
- In the past, the bias has been very useful
  - Without it, automated planning wouldn’t have grown into a separate field from its potential application areas
- But it’s not useful any more
  - The field has matured
  - The bias is too restrictive
What are classical planners bad at?
Example

- Typical characteristics of application domains
  - Dynamic world
  - Multiple agents
  - Imperfect/uncertain info
  - External info sources
    - users, sensors, databases
  - Durations, time constraints, asynchronous actions
  - Numeric computations
    - geometry, probability, etc.
- Classical planning excludes all of these
Relax the Assumptions

- Relax A2 (deterministic $\Sigma$):
  - Actions have more than one possible outcome
  - Seek policy or contingency plan
  - With probabilities:
    » Discrete Markov Decision Processes (MDPs)
  - Without probabilities:
    » Nondeterministic transition systems

$$\Sigma = (S, A, E, \gamma)$$
$$S = \{\text{states}\}$$
$$A = \{\text{actions}\}$$
$$E = \{\text{events}\}$$
$$\gamma: S \times (A \cup E) \rightarrow 2^S$$
Relax the Assumptions

- Relax A1 and A2:
  - Finite POMDPs
    - Plan over belief states
    - Exponential time & space
- Relax A0 and A2:
  - Continuous or hybrid MDPs
    - Control theory
- Relax A0, A1, and A2:
  - Continuous or hybrid POMDPs
    - Robotics

\[ \Sigma = (S, A, E, \gamma) \]
\[ S = \{ \text{states} \} \]
\[ A = \{ \text{actions} \} \]
\[ E = \{ \text{events} \} \]
\[ \gamma: S \times (A \cup E) \rightarrow 2^S \]
Relax the Assumptions

- Relax A3 (static $\Sigma$):
  - Other agents or dynamic environment
    - Finite perfect-info zero-sum games (introductory AI courses)
  - Randomly behaving environment
    - Decision analysis (business, operations research)
    - Can sometimes map this into MDPs or POMDPs
  - Case studies: Chapters 19 (space), 22 (emergency evacuation)

- Relax A1 and A3
  - Imperfect-information games
  - Case study: Chapter 23 (bridge)

\[ \Sigma = (S, A, E, \gamma) \]
\[ S = \{ \text{states} \} \]
\[ A = \{ \text{actions} \} \]
\[ E = \{ \text{events} \} \]
\[ \gamma: S \times (A \cup E) \rightarrow 2^S \]
Relax the Assumptions

- Relax A5 (sequential plans) and A6 (implicit time):
  - Temporal planning
- Relax A0, A5, A5
  - Planning and resource scheduling

\[
\Sigma = (S, A, E, \gamma)
\]

\[
S = \{\text{states}\}
\]

\[
A = \{\text{actions}\}
\]

\[
E = \{\text{events}\}
\]

\[
\gamma: S \times (A \cup E) \rightarrow 2^S
\]