Planning for Partially Observable, Nondeterministic Domains

Dagstuhl Tutorial 05241

Piergiorgio Bertoli

bertoli@itc.it

ITC-IRST, Via Sommarive 18, 38055 Povo, Trento, Italy
Outline of talk

- Problem statement and motivation
- The framework
- And-or search
- Symbolic representation
- Related work
The problem

Given a *domain* which evolves according to performed *actions*, producing *observations*...

Given a *goal* (i.e. a behavioral requirement)...

![Diagram](image-url)
The problem

- Given a domain which evolves according to performed actions, producing observations...
- Given a goal (i.e. a behavioral requirement)...
- Synthesize a plan that controls the domain to achieve the goal.
Planning problem categories

- Deterministic
- Full obs.
- Simple goals
- No obs.
- Partial obs.
- Extended goals
- Init. uncertainty
- Gen. nondeterminism

Goals

Observability

Nondeterminism
Planning problem categories

- Goals
  - extended goals
  - simple goals
  - full obs.
  - deterministic
  - no obs.
  - init. uncertainty
  - gen. nondeterminism

- Observability
  - partial obs.

Our problem
Qualitative nondet. vs. probabilities

- **Probabilistic model:**
  - State uncertainty as probability distribution
  - Nondeterminism as probabilistic transition and output

- **Qualitative model:**
  - State uncertainty as set of possible states
  - Nondeterminism as nondeterministic transition/output functions
Qualitative nondeterminism vs. probabilities

- Probabilistic model: $\rightarrow$ (PO)MDPs [Givan, Hansen, Geffner, ...]
  - State uncertainty as probability distribution
  - Nondeterminism as probabilistic transition and output

- Qualitative model: $\rightarrow$ this talk
  - State uncertainty as set of possible states
  - Nondeterminism as nondeterministic transition/output functions
The general framework

- The *domain* evolves according to performed *actions*.

- The *plan* provides the *actions* according to the *observations* on the domain.
The model of the domain

A domain D is a nondeterministic Moore machine:

- D’s input is an action.
- D’s output is an observation.
- D’s state represents the domain state.
- D’s output and next state are non-deterministic.
A plan $P$ is a deterministic Mealy machine:

- P’s input is an *observation*.
- P’s output is an *action*.
- P’s state represents the *plan state*.
- P’s output and next state are deterministic.
The execution of a plan on a domain is the “synchronous execution” of the two machines.

The state of the execution (configuration) justaposes the state of the domain and that of the plan.
A planning domain is defined as a tuple $D = \langle S, S_0, A, T, P, L, O, X \rangle$:

- $S$ is a finite set of states.
- $S_0$ is the initial set of states.
- $A$ is a finite set of actions.
- $T \subseteq S \times A \times S$ is the transition relation.
- $P$ is a finite set of atomic propositions.
- $L : P \mapsto 2^S$ assigns to each $p \in P$ the set of states where $p$ holds.
- $O$ is a finite set of possible observations.
- $X \subseteq S \times O$ is the observation relation.
Example: a simple robot domain

- Domain state: robot position
- Domain observations: configuration of walls in current room
- Actions: moving (and sliding...)
A plan $\Pi$ for a planning domain $\mathcal{D}$ is a tuple $\Pi = \langle \Sigma, \sigma_0, \Psi, \Upsilon \rangle$ where:

- $\Sigma$ is a finite set of plan states.
- $\sigma_0$ is the initial plan state.
- $\Psi : \Sigma \times O \rightarrow A$ associates to a pair $\langle \sigma, o \rangle$ an action $\alpha$ to execute.
- $\Upsilon : \Sigma \times O \rightarrow \Sigma$ associates to a pair $\langle \sigma, o \rangle$ a new plan state $\sigma'$.

...but for reachability goals, enough to consider conditional tree-like plans such as:

$GoEast; \text{if WallN then GoSouth; GoWest else GoWest}$
Execution traces

An execution is represented as a trace of configurations.

Uncertainty makes several executions possible:

Strong planning: “every final state of execution trace must be in goal”.
Search space

- At each execution step a set of states is possible (*belief*)
- Search space: and-or *cyclic* graph whose nodes are beliefs
Induced lattice

- Entailment relation induces a lattice
- Can be used to propagate success/failure
Search mechanism

State of search: *acyclic* prefix of search space (DAG)

Expansion step: apply every action to a fringe node, and consider every possible observation

Lazyness: consider sensors incrementally (add intermediate nodes)

Heuristics for fringe node selection
Symbolic Representation

- States represented as in symbolic model checking: $\xi(s)$.
- A vector $\alpha$ of Boolean variables, each corresponding to an action in $A$:
  $$\alpha = \{GoWest, GoEast, GoNorth, GoSouth\}$$
- A transition $t = \langle s_s, a, s_d \rangle$ encoded as
  $$\xi(t) = \xi(\langle s_s, a, s_d \rangle) = \xi(s_s) \land \xi(a) \land \xi'(s_d)$$
- Observations represented by vector $\sigma$ of Boolean observation variables
- Observation function encoded by associating a pair of beliefs to each Boolean observation variable
- BDD used to represent encoded transitions and functions.
Practical considerations

Approach scales effectively for a wide variety of problems:

Leverages on symbolic representation to manage uncertainty:

- Large beliefs may correspond to small BDDs
- Image primitives evolve sets of traces at once effectively, e.g.
  \[ FImg(S, a) = \{ s' \mid s \in S \land \langle s, a, s' \rangle \in T \} \leadsto \exists x. (S(x) \land T(x, a, x')) \]
- Effective entailment-checking primitives

Variable-based representation of sensing allows for lazyness

- Very often reduces branching
- Efficient detection of “redundant” sensing

Informed heuristics can be efficiently computed, e.g. Distance-based heuristics (by preimages), Cardinality-based heuristics, ...
A different search style
A different search style

...not a good idea?
A different search style

...yes, if domain is “modular”: 
Related approaches

- QBF-based encoding (Rintanen)
- Regression-based symbolic planning (Rintanen)
- Based on Planning Graphs (Weld, Smith; Bryce, Kambhampati)
- Based on Implicit Representation of Beliefs (Hoffmann, Brafman)
- Based on Knowledge Abstractions (Bacchus)
Extensions

Extended goals →
- enrich belief structure
- consider general (looping) plans
- hard to do (esp. symbolically)

On-line reactive planning →
- consider means to constrain search (e.g. assumptions, constraints)
- add monitoring/replanning component to architecture