Uncertainty in Planning

Manuela M. Veloso

Carnegie Mellon University
School of Computer Science

15-887 – Planning, Execution, and Learning

Fall 2016

Sources of Uncertainty

- Initial State
  - Unknown predicates
- Action Models
  - Non-deterministic effects
- Sensor Noise
  - Partially hidden state

Handling Uncertainty

- **Conformant Plans**
  - Guaranteed to succeed despite uncertainty
  - Sequence
- **Conditional Plans**
  - With, or without, failure
  - Branching
- **Universal Plans / Policies**
  - Non-deterministic
  - Probabilistic
  - State/action mapping

Conformant Planning

- Create a (Non-Branching) Plan that Achieves Goals Despite Uncertainty
  - Uncertainty in initial state
  - Uncertainty in effects
- No Sensing Actions
- Basic Idea:
  - ***Remove uncertainty by forcing world into known states***

Conditional Effects

- Action Sprinkle-grass
- Pre:
  - turned-on water
- Effects:
  - wet grass
  - If object on grass
    - Then wet object
- Result of applying action
- Use of conditional effects as goals

Planning and Acting in Nondeterministic Domains

- Problems with domains:
  - partially observable (which state)
  - nondeterministic (multiple effects of actions)
  - unknown or poorly-known environments
- Sensorless planning – conformant planning
- Contingency planning
- Planning and Replanning

Planning and Acting

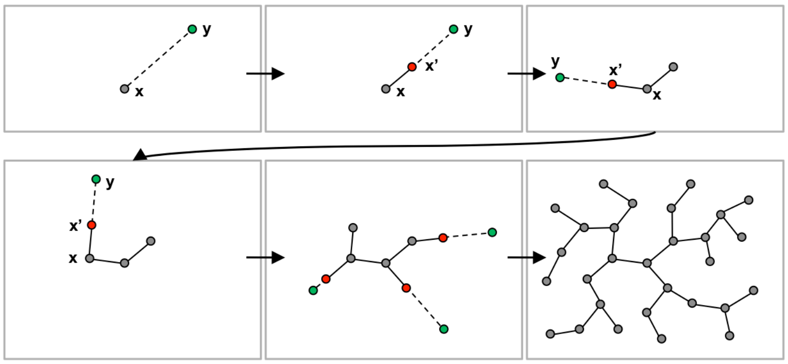
- Explicitly represent the variables with unknown values and allow preconditions with such variables
- Add to the domain a PERCEPT action schema for all variables whose value is unknown
- Many examples

Contingency Planning

- When applying an action, need “to sense” to verify conditions – use percept action
- Determine state in which action will be applied
  - State-space planners
  - Plan-state planners
  - Other planners

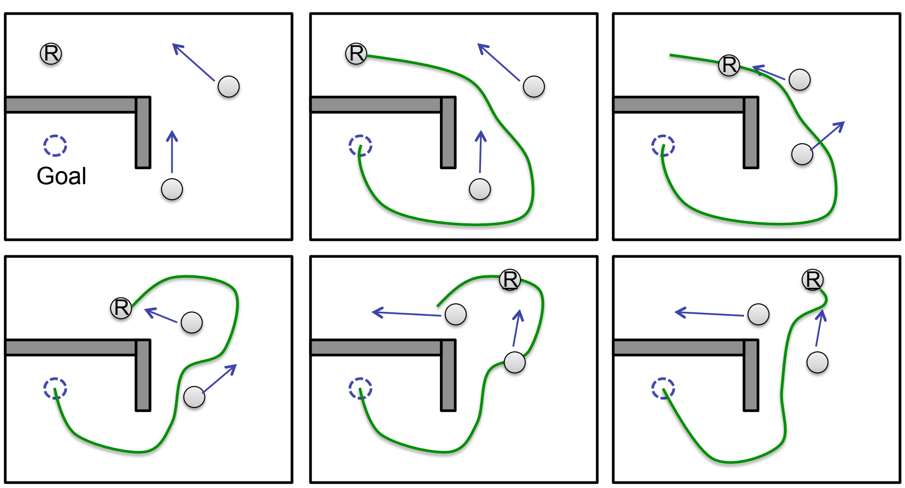
Continuous Spaces: Rapidly-Exploring Random Trees (RRT)

- Create a random sample **y** from some subspace **Y** of **X**
- Find the nearest neighbor **x** in tree **T** using some distance metric
- Expand from **x** toward **y**, creating a new child node y'



[S. LaValle, 1998] [J. Kuffner, S. LaValle, 2000]

Planning Problems in Adversarial Poorly Modeled Domains



Ⓡ = Controlled Robot ◯ = Other Poorly Predictable Moving Bodies

Planning in a Dynamic World

- At each (re)planning iteration, the planner searches for a complete solution, trying to find a precise trajectory around moving objects , **all the way to the goal state.**
- In domains with high uncertainty, predictions for poorly modeled objects are likely incorrect.
- Generated plans fail early, and most of the plan will **never be successfully executed.**
- A waste of computational planning resources!

Key Question

- How can we make planning and replanning in unknown or poorly modeled environments more efficient?

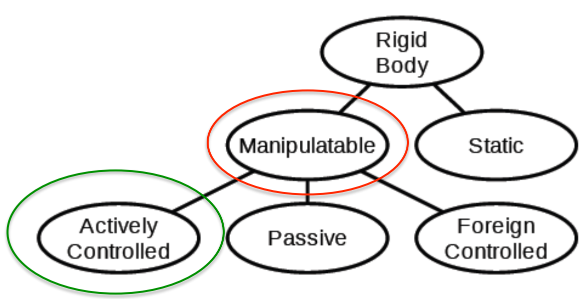
Approach

- ***Variable Level-of-Detail (VLOD)*** planning
- The planner adjusts level of knowledge:
  - prevents collisions in the *near future*
  - ignorss collisions in the *far future*

- **Future**: defined by time horizon threshold: $\mathbf{t_{LOD}}$
  - If **x.t >** $\mathbf{t_{LOD}}$ then ignore **details** during planning
  - Otherwise, plan with **full detail**

Variable Level-of-Detail Planning

- What is a **detail**?
  - Locally solvable multi-body interactions that do not affect the global topology of the plan if temporarily ignored
  - In our model: Interactions between the **actively controlled** and other **manipulatable** bodies

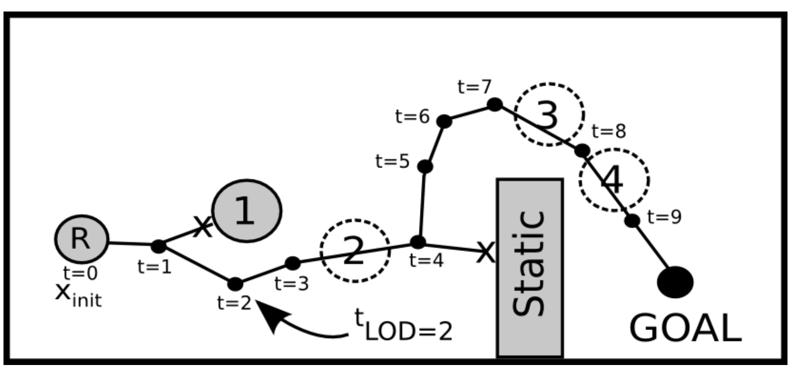


VLOD Physics-Based Planning

T.AddVertex(x_{init}); //start with a tree that only contains our initial state
repeat until we give up
y := SampleRandomState(Y); //generate a sample the sampling space Y
x := NearestNeighbor(T,y); //find nearest neighbor to y in T
a := Controller(x,y); //generate an action from x toward y
SetupCollisionMatrix(x); //sets up the currently applicable level of detail
[x', L] := SimulatePhysics(x, a); //apply and simulate the control action
if isValidState(x', L) **then** //check if any constraints have been violated
T.AddVertex(x'); //add x' as a child of x
T.AddEdge(x, x', a); //saving the control action a as edge
if x' is in X_{goal} **then** return x'; //have we reached the goal?
end if

VLOD Example

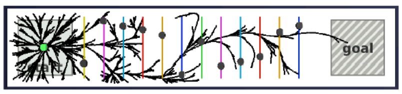
- t_{LOD}=2



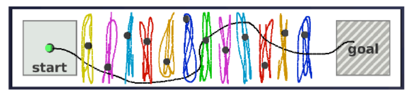
R: Controlled Robot Body
1-4: Foreign-Controlled or other Manipulatible Bodies

Experimental Domains

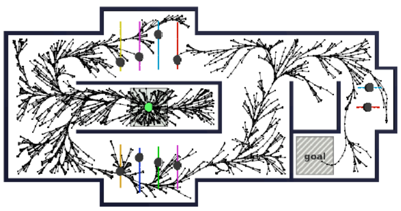
- Navigate through a field of rapidly moving foreign-controlled bodies
- Foreign-controlled bodies have simulated uncertainty
- Controllable replanning interval t_{replan}
- Controllable VLOD time horizon t_{LOD}



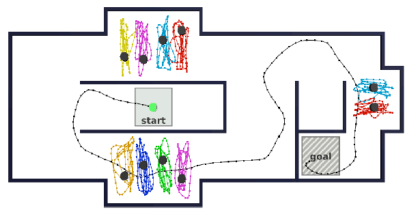
(a) A search tree in the Hallway domain



(b) A solution trajectory in the Hallway domain, under high uncertainty



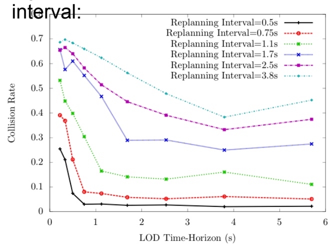
(c) A search tree in the Maze domain



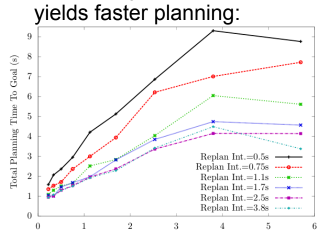
(d) A solution trajectory in the Maze domain, under high uncertainty

Results: VLOD Planning Performance

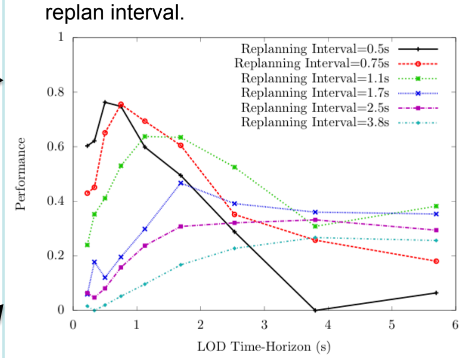
Collisions only significantly increase if t_{LOD} is close or below the replan interval:



Smaller t_{LOD} generally yields faster planning:



VLOD Planning achieves best overall performance if t_{LOD} is slightly above the replan interval.



$$\text{performance} = (1 - \text{NormCollisions})\,(1 - \text{NormTime})$$

VLOD Planning can significantly reduce cumulative planning time without increasing collision rates.

Hallway domain, uncertainty: 0.75, each data point: 120 trials (6480 simulated trials total)

Summary

- Uncertainty
  - Sensorless planning
  - Contingency planning
  - Planning and replanning
- Later
  - Probabilistic representations
  - Planning under probabilistic uncertainty