An approach to generate MDPs using HTN representations

Felipe Meneguzzi – CMU
Yuqing Tang – CUNY
Katia Sycara – CMU
Simon Parsons – CUNY

Carnegie Mellon
THE ROBOTICS INSTITUTE

Overview

- Background
  - HTNs
  - MDPs
- Conversion
  - Generating MDP states
  - Generating transition functions
  - Generating reward function
- Experiments
- Future Work

BACKGROUND

Planning

- Planning algorithms more or less divided into:
  - Deterministic
  - Probabilistic
- Formalisms differ significantly
  - Domain representation
  - Concept of solution
    - Plan
    - Policy

Hierarchical Task Networks (HTN)

- Offshoot of classical planning
- Domain representation more intuitive to human planners
  - Actions (state modification operators)
  - Tasks (goals and subgoals)
  - Methods (recipes for refining tasks)
- Problem comprises
  - Initial State
  - Task

Domain Example

- Going to London from New Jersey
  - Various options of vehicle to use
  - Various routes depending on the chosen vehicle
- Two operator/action templates:
  - Move To $a^{mt(F,T,V)}$
  - Get Vehicle $a^{gv(V)}$

Operators

```
(:operator (!moveTo ?l1 ?l2 ?v)
             ((has ?v) (at ?l1))
             ((at ?l1))
             ((at ?l2)))
(:operator (!getVehicle ?v)
             ()
             ()
             ((has ?v)))
```

Tasks

- Go To – top level task

$$t^{GT(L)}$$

- Obtain Vehicle – task to obtain long range transportation

$$t^{OV(V)}$$

- Move To – task to move through multiple intermediary points to reach a destination

$$t^{MT(L)}$$

Methods

```
(:method (goTo ?l)
    ()
    ((!getVehicle car)
     (obtainVehicle)
     (!moveTo ?l))
  )
```

Methods

```
(:method (obtainVehicle)
  ((at home))
    ((!moveTo home airport car)
     (!getVehicle plane))
  ((at home))
    ((!moveTo home harbor car)
     (!getVehicle ship))
  )
```

Methods

```
(:method (moveTo ?l)
  ((has plane))
    ((!moveTo airport nyc plane)
     (!moveTo nyc london plane))
  ((has ship))
    ((!moveTo harbor soton ship)
     (!moveTo soton london ship))
  ((has ship))
    ((!moveTo harbor lpool ship)
     (!moveTo lpool london ship))  )
```

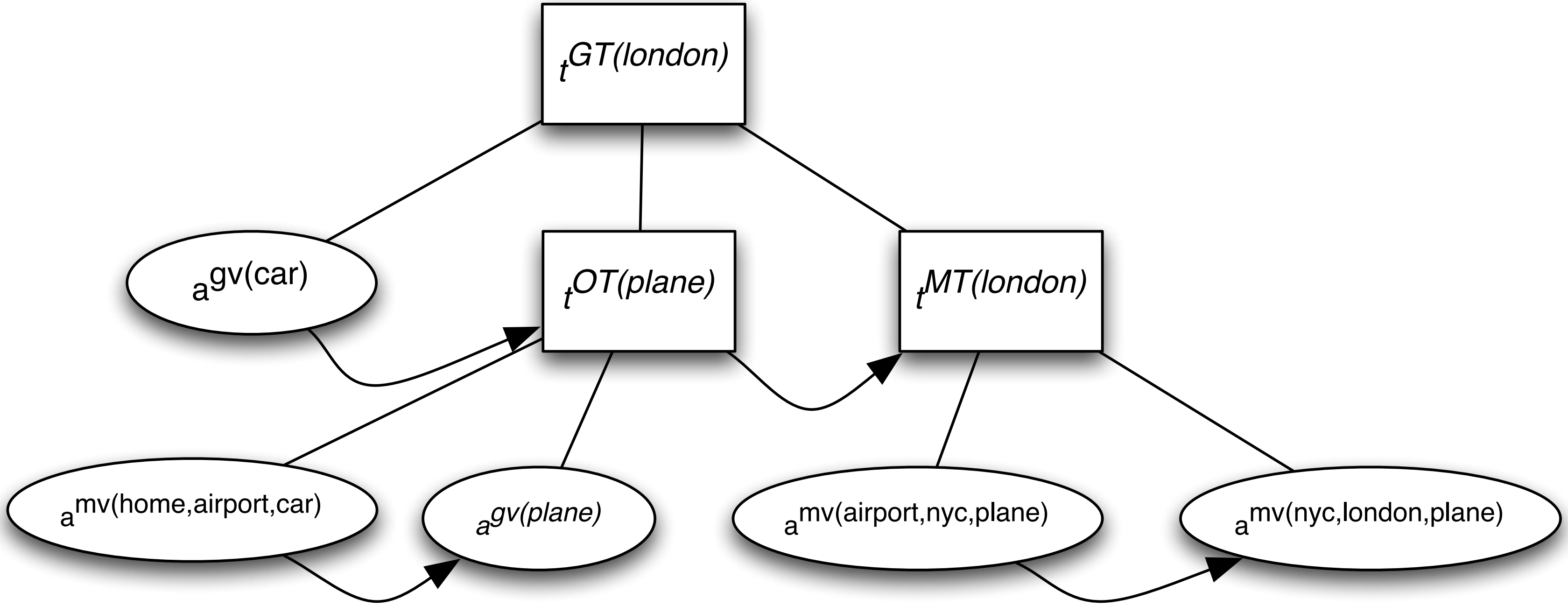
HTN Problem

- How to execute task goTo(london)

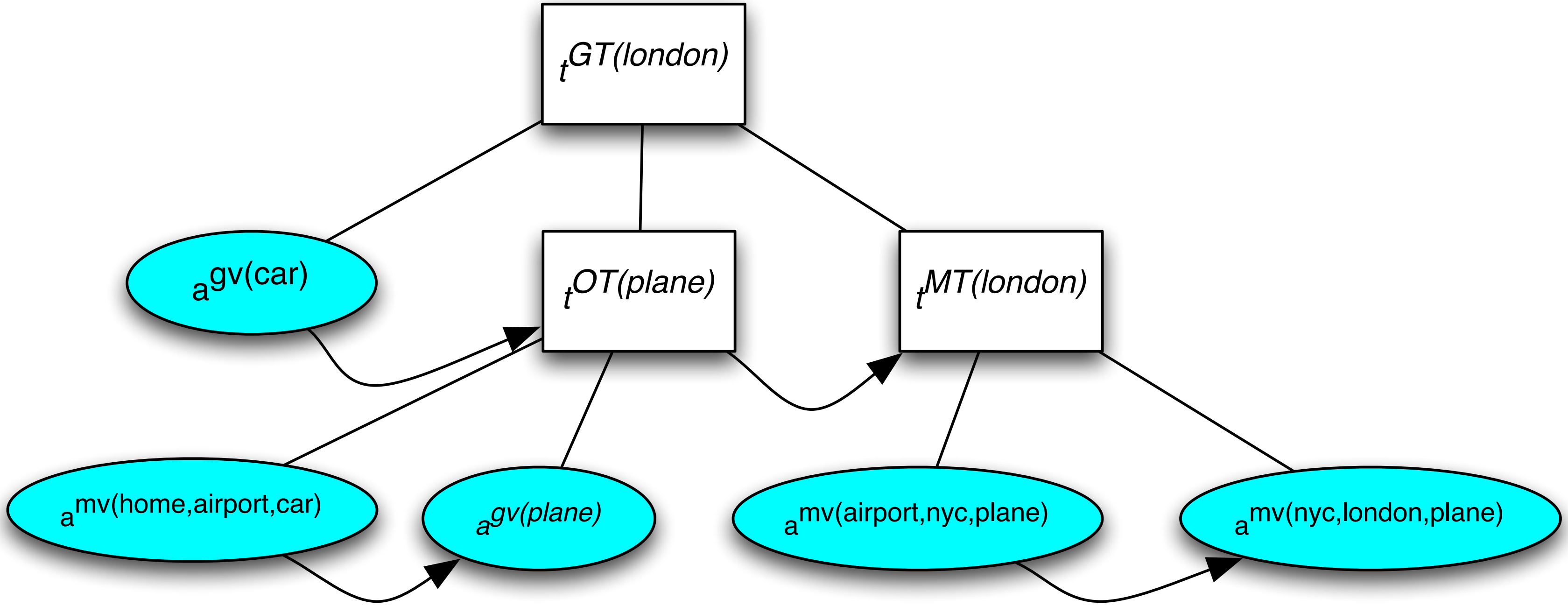
$$t^{GT(london)}$$

  - Decompose task through methods in the domain until actions reached
  - Ordered actions are the solution

Decomposed Problem



HTN Solution



Markov Decision Processes (MDP)

- Mathematical model for decision-making in a partially controllable environment
- Domain is represented as a tuple

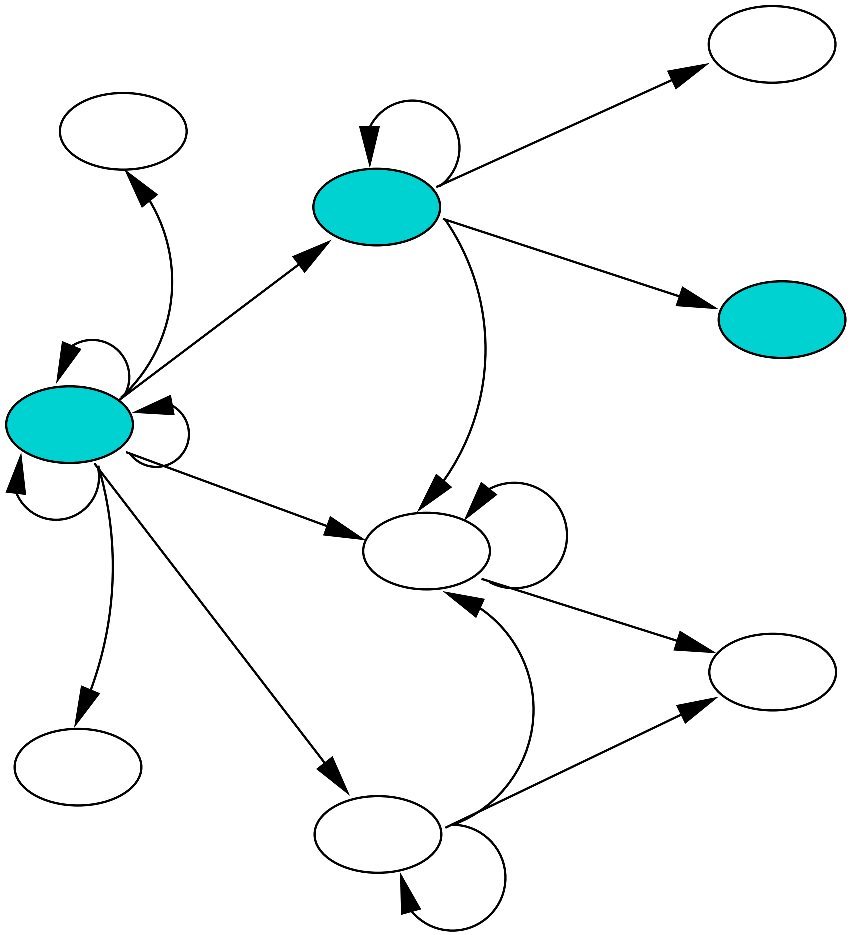
$$\Sigma = (S, A, \Pr, u)$$

where:

- S is the entire state space
- A is the set of available actions
- Pr is a state transition function
- u is a utility function

MDP Domain

- Represented as a hypergraph
- Connections are not necessarily structured
- **All** reachable states are represented
- State transition function specifies how actions relate states



Computing an MDP Policy

- An MDP policy is computed using the notion of expected value of a state:

$$V^*(s) = \max_{a \in A(s)} \left[u(a,s) + \gamma \sum_{s' \in S} \Pr_a(s' \mid s) V^*(s') \right]$$

- Expected value comes from a reward function
- An optimal policy is a policy that maximizes the expected value of every state

$$\pi^*(s) = \arg\max_{a \in A(s)} \left[u(a,s) + \gamma \sum_{s' \in S} \Pr_a(s' \mid s) V^*(s') \right]$$

MDP Solution

- Solution for an MDP is a policy
- Policy associates an *optimal* action to every state
- Instead of a sequential plan, policy provides contingencies for every state

state0 → actionB

state1 → actionD

state2 → actionA

Using HTNs to Represent MDPs

CONVERSION

States

Hierarchical Task Network

- Not enumerated exhaustively
- State consists of properties of the environment

$$at(airport) \wedge has(plane)$$

- Each action modifies properties of the environment
- Set of properties induces a *very large* state space

Markov Decision Process

- MDP domain explicitly enumerates all *relevant states*
- Formally speaking, MDP states are monolithic entities
- Implicitly represent the same properties expressed in HTN state
- Large state-spaces make the algorithm flounder

State Space Size

Hierarchical Task Network

- Set of actions induces a smaller state space (still quite large)
- Set of methods induces a smaller still state space
- HTN planning consults this latter state space

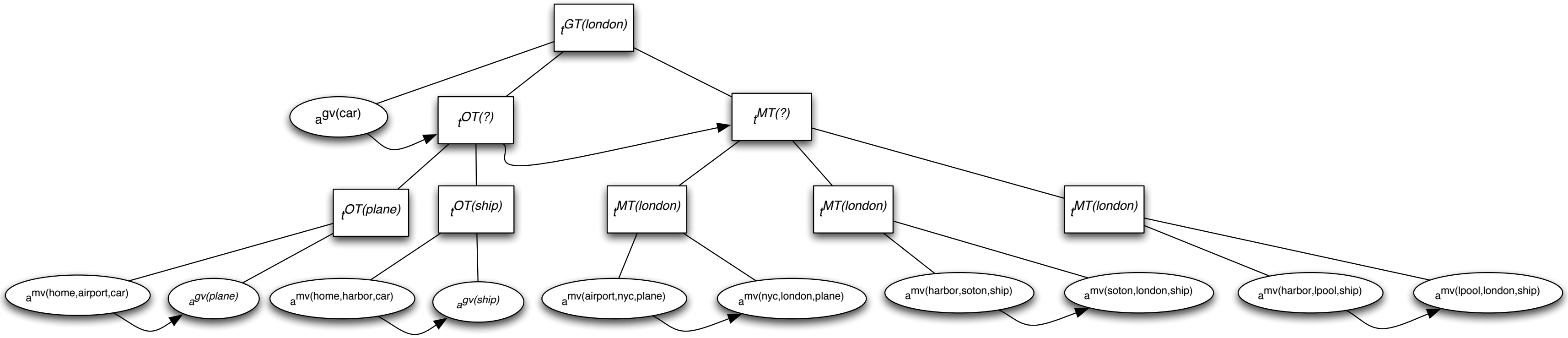
Markov Decision Process

- MDP solver must consult the *entire* state space
- State-space reduction techniques include:
  - Factorization
  - ϵ-homogeneous aggregation

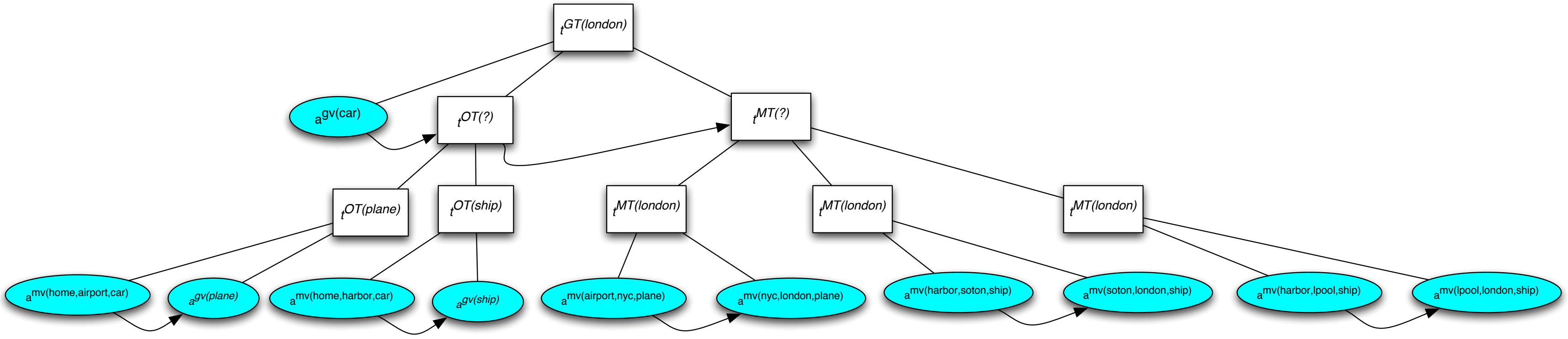
Generating States

- MDP State-space is derived from the reachable states induced by the primitive actions in a fully expanded HTN
- State-space comes from the reachable primitive actions induced my HTN methods

Fully Expanded HTN



Reachable States



Generating States

- Primitive tasks in the fully expanded graph induce a state:
  - Disjunction of all possible states given previous decompositions
  - In most domains tasks will induce overlapping states
- Algorithm must keep track of overlapping states (for the transition functions)

Generating Actions

- This page intentionally left blank

Generating Transition Functions

- Formally, an HTN is a graph defined by tasks (vertices) and ordering constraints (edges)
- Each primitive task *ti* corresponds to both an action name and a possible state
  - If there is a constraint between tj and ti, then there is a non-zero probability of a transition
- Probabilities are uniformly distributed over a planner's choice

Generating a Reward Function

- States in the last tasks in the HTN are the implicit objective
- Thus, for each possible path through the HTN's primitive actions, there is a *gradient* of utility
- Reward for each state then is proportional to the index of the task in a given path

Improving Performance

- Algorithm analysis and profiling has shown that most time is spent computing possible states
  - Results of action execution on disjunctive formulas
  - Support for precondition from a disjunction of possible states
- BDDs provide polynomial time operations

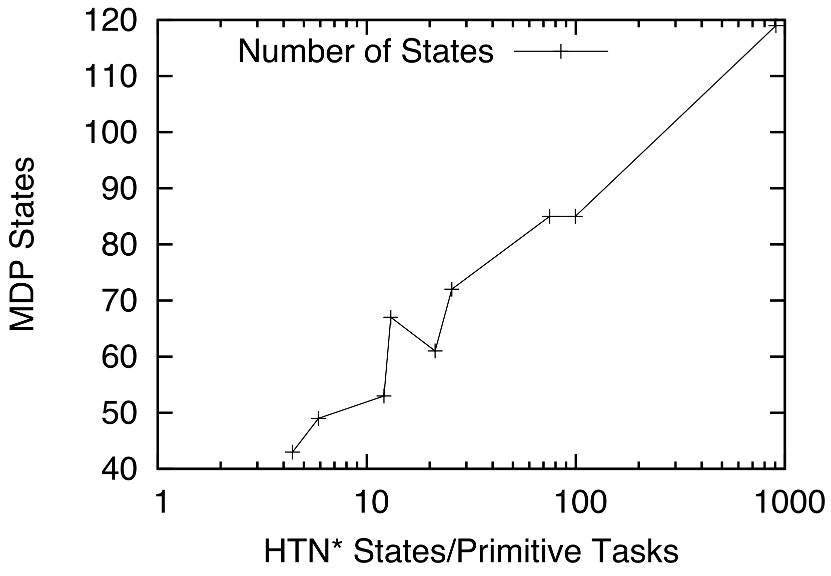
Experiments and Results

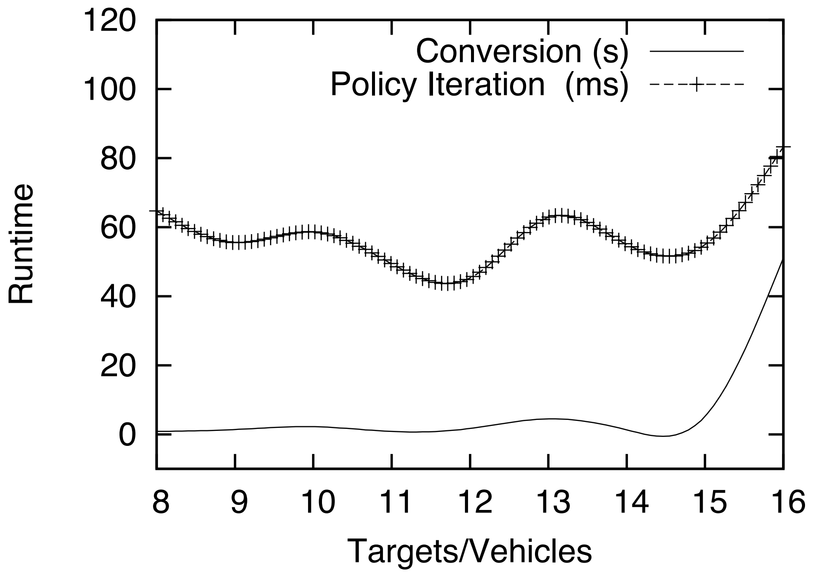
EXPERIMENTS

Experiment Domain

- Military planning scenario
  - Allocation of vehicles to routes and targets
  - 7 to 11 different vehicles attacking 1 to 3 targets over multiple possible route combinations
  - Herbrand base with 15 to 25 thousand predicates

Experiments





Conclusions and Future Work

FUTURE WORK

Future Work

- Using a BDD-based state and operator representation
  - Challenges: enumeration of reachable state-space
  - Encoding planning operators using the least amount of BDD variables
  - Proper variable ordering
- Applications
  - Domain modeling in probabilistic planning
  - Probabilistic planning in agent programming languages