CAP6671 Intelligent Systems

Lecture 17:

Evaluating Role Allocation

Instructor: Dr. Gita Sukthankar
Email: gitars@eecs.ucf.edu
Schedule: T & Th 9:00-10:15am
Location: HEC 302
Office Hours (in HEC 232):
T & Th 10:30am-12

Research contributions?

Problem

- Large number of potential team policies (doubly exponential in team members and time steps)
- Most coordination algorithms don't make any type of guarantee about the optimality of the assignment
- Need mechanism to evaluate the utility of different role allocation strategies
- MTDP formulation focuses on the key parts of the decision space

Weaknesses

- Can be difficult to create and evaluate an MTDP for different types of problems
- Might get better results by simulating the effects of different policies (since it takes a day or so to run RMTDP anyways)

Team-oriented Planning

- Deliberative planning framework for developing team plans
- Typically decouples the planning/scheduling framework from the role assignment problem
- Role allocation is often executed as a run-time decision or a pre-condition for planning operators

STEAM

- Separate reasoning about teamwork for reasoning about taskwork
- Implemented as a set of domain-independent rules in SOAR
- Based on Joint Persistent Goal teamwork formalism (Cohen and Levesque)

POMDPs

- *Partially observable Markov Decision Process (POMDP)*:
  - a stochastic system $\Sigma = (S, A, P)$ as before
  - A finite set O of *observations*
    - $P_a(o|s)$ = probability of observation o in state s after executing action a
  - Require that for each a and s, $\sum_{o \text{ in } O} P_a(o|s) = 1$

- O models partial observability
  - The controller can't observe s directly; it can only observe o
  - The same observation o can occur in more than one state

- Why do the observations depend on the action a? Why do we have $P_a(o|s)$ rather than $P(o|s)$?
  - This is a way to model *sensing actions*, which do not change the state but return information make some observation available (e.g., from a sensor)

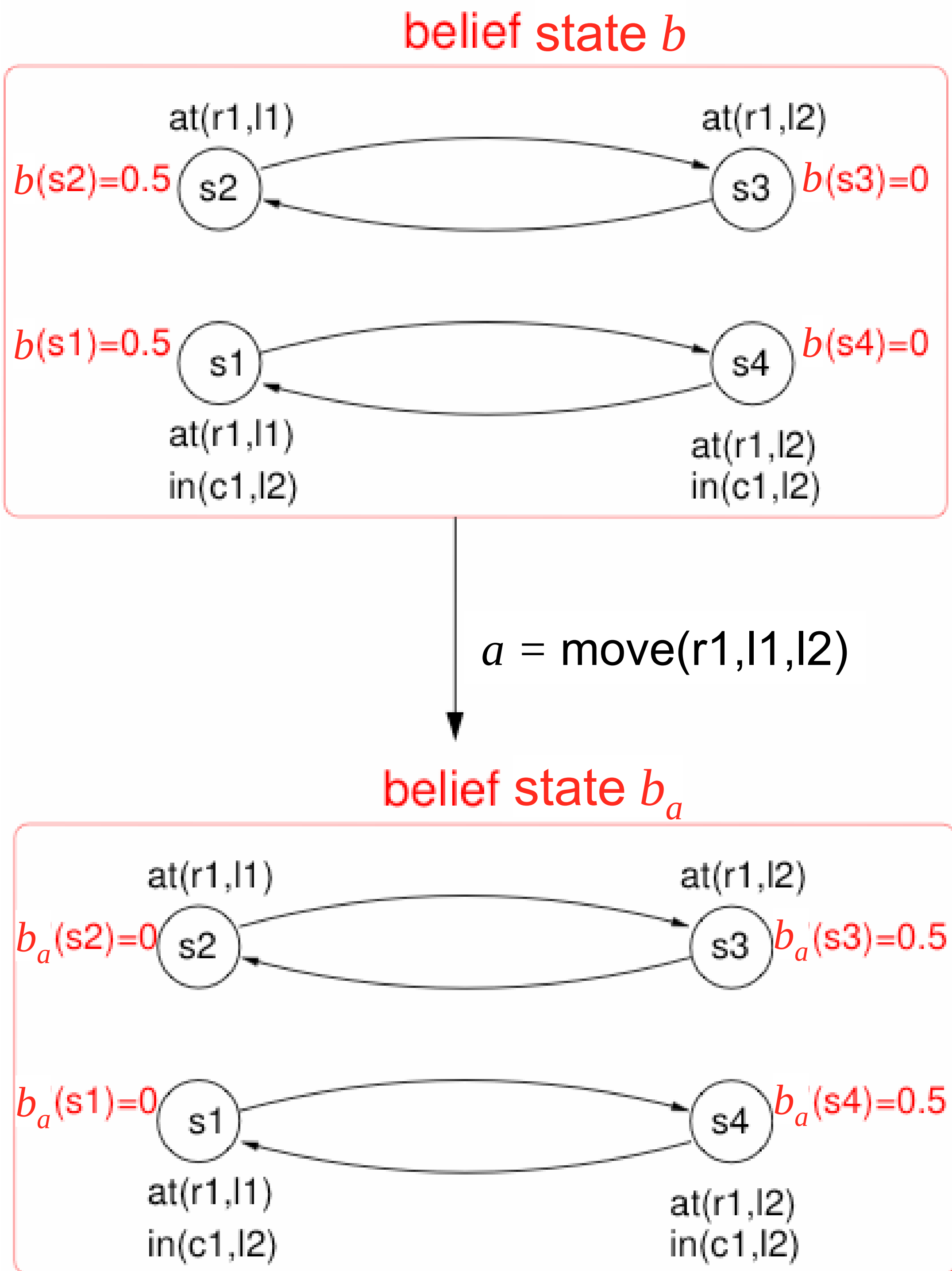
Belief States

- At each point we will have a probability distribution $b(s)$ over the states in S
  - $b(s)$ is called a *belief state* (our belief about what state we're in)
- Basic properties:
  - $0 \leq b(s) \leq 1$ for every s in S
  - $\sum_{s \text{ in } S} b(s) = 1$
- Definitions:
  - b_a = the belief state after doing action a in belief state b
    - Thus $b_a(s) = P(\text{in } s \text{ after doing } a \text{ in } b) = \sum_{s' \text{ in } S} P_a(s|s')\, b(s')$
  - $b_a(o) = P(\text{observe } o \text{ after doing } a \text{ in } b)$ **Marginalize over states**
    $= \sum_{s \text{ in } S} P_a(o|s)\, b(s)$
  - $b_a^o(s) = P(\text{in } s \text{ after doing } a \text{ in } b \text{ and observing } o)$

Belief states are n-dimensional vectors representing the probability of being in every state..

Example

- Robot r1 can move between l1 and l2
  - move(r1,l1,l2)
  - move(r1,l2,l1)
- There may be a container c in location l2
  - in(c1,l2)
- O = {full, empty}
  - full: c1 is present
  - empty: c1 is absent
  - abbreviate full as f, and empty as e



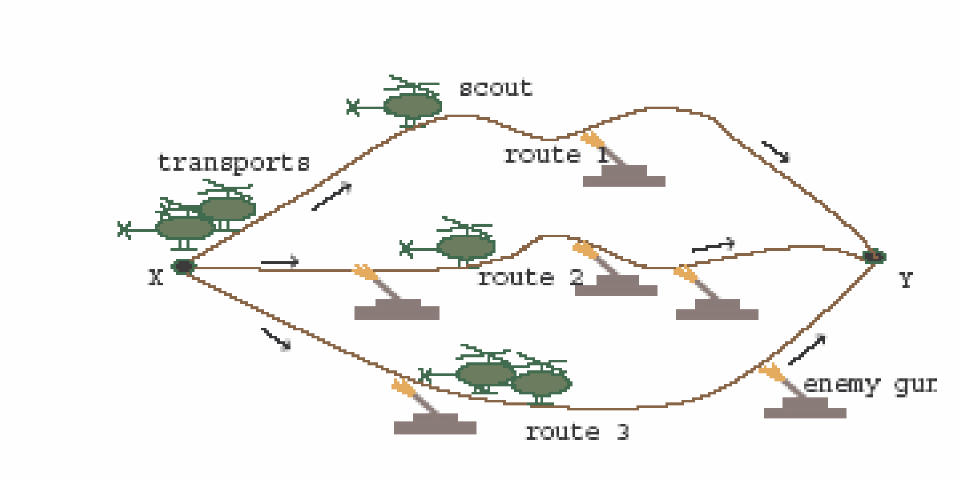
Algorithm

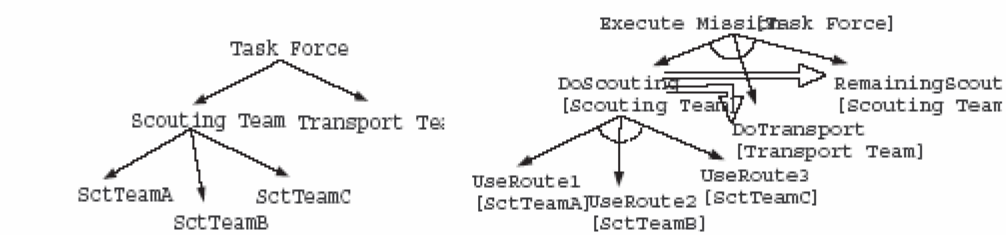
1. Select a group of candidate role allocation strategies
2. Create an abstract Markov representation of the decision-space
3. Create separate RMTDP for different sections of the problem
4. Search through the space of potential policies
5. Prune space of valid role assignments using various heuristics (MAXEXP, NOFAIL)

MTDP

- MTDP=Markov Team Decision Problem (equivalent to a distributed POMDP framework)
- COM-MTDP=Communication Markov Team Decision Problem (cited in prior work)
  - Create to evaluate the effects of different communication policies
  - Proofs in this paper extend on work in previous paper
- RMTDP=Role-based Markov Team Decision Program
  - Used to evaluate the effects of different role allocation policies
- Note that the authors use the MDP formalism to conceptualize the problem, not as the actual solver (which is a team-oriented planner)

Team-Oriented Plan: Transport





- Scenario used by MRE and TacAirSOAR
- 3 subplans
  - DoScouting
  - DoTransport
  - ExecuteMission
- Can allocate different numbers of helicopters to each section of the plan

Team-Oriented Plan: Rescue

- Full city version of the RoboCupRescue (not the Virtual Robot Competition that we talked about in class)
- Agent must allocate human rescue workers in a city that has recently experienced an earthquake
- Subplans are: **ExtinguishFire** and **RescueCivilians**

MTDP Formalism

- Includes states, actions, agents, observations, rewards
- State description does not have to be complete but should model variables in precondition and termination conditions
- Search through spaces of joint policy to find optimal reward
- Agents alternate between role-taking actions (even timesteps) and role-execution actions (odd timesteps)
- Role-taking can involve penalties as the agent ceases participation in a task

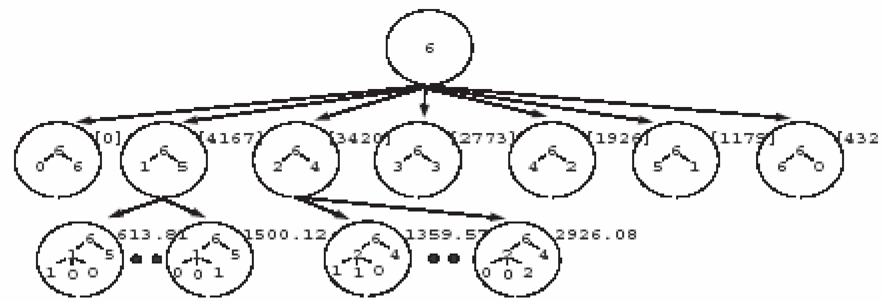
Trigger Actions

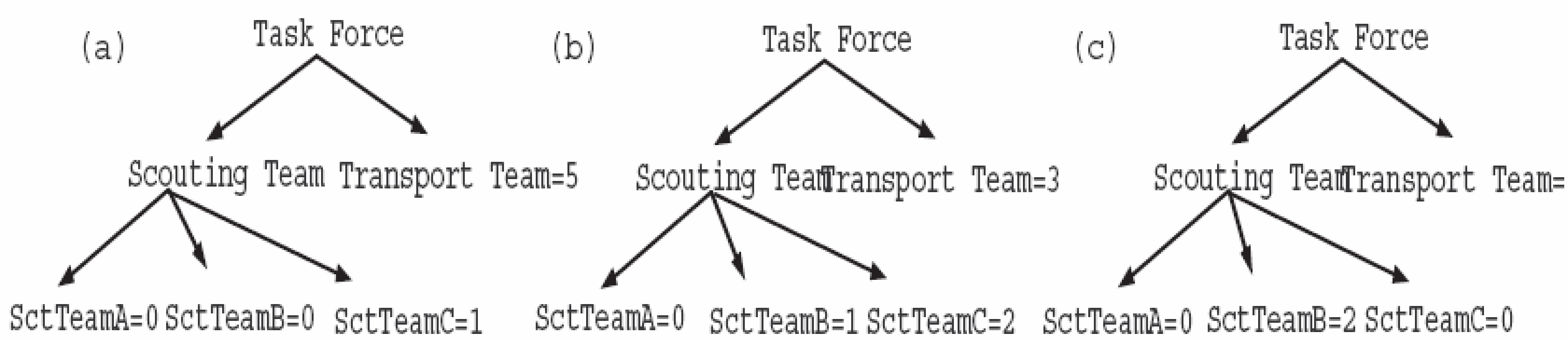
- Triggers=observations that prompt role-taking actions (role re-allocation)
- Task failures are often a trigger
- The allocation algorithms evaluated here use utility based heuristics to decide whether to reallocate the agent
  - Agent criticality (STEAM)
  - Utility tradeoff (FCP_helo)

Policy Generation

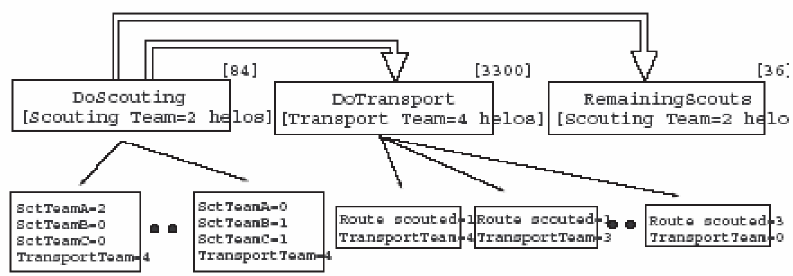
```
epochs $\leftarrow \{0,\ldots,T\}$; policySpace $\leftarrow$ null
for each agent decision epoch $t <= T$
 for each joint obs. history $\omega^0 \ldots \omega^{t-1}$
  for each policy $\pi$ in policySpace
   $\pi' \leftarrow \pi$; $\pi'' \leftarrow \pi$
   for each joint observation $\omega^t$
    for each trigger in trigger list
     if trigger.triggered($\omega^0 \ldots \omega^t$)=true
      $i \leftarrow$ trigger.respondingAgent
      $\pi'_i[\omega_i^0 \ldots \omega_i^t] \leftarrow$ respond
      $\pi''_i[\omega_i^0 \ldots \omega_i^t] \leftarrow$ dontRespond
      policySpace $\leftarrow$ policySpace.add($\pi'$)
      policySpace $\leftarrow$ policySpace.add($\pi''$)
     else
      $\pi'_i[\omega_i^0 \ldots \omega_i^t] \leftarrow \pi_{original}[\omega_i^0 \ldots \omega_i^t]$
      policySpace $\leftarrow$ policySpace.add($\pi'$)
   policySpace $\leftarrow$ policySpace.remove($\pi$)
return best(policySpace)
```

Possible Allocations



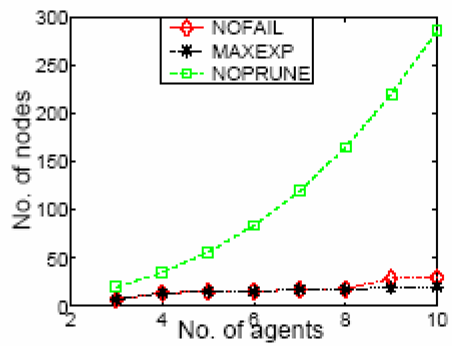


Pruning Strategy



- Generate over-estimate for value of parent node
- Prune if over-estimate is less than other nodes (MAXEXP)
- Evaluated 3 pruning conditions
  - NOPRUNE
  - MAXEXP
  - NOFAIL (assume nodes never fail)

Results



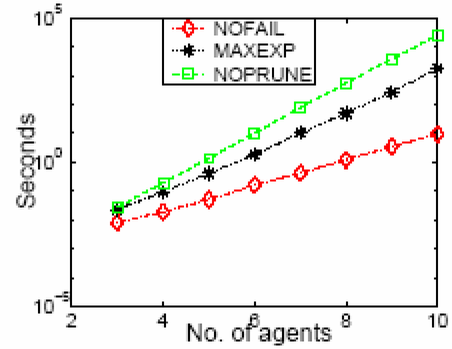


Table 1: Performance of role allocations in RoboCupRescue

	Civilians saved	Bldg. damage	RoboCupRescue
Best Alloc.	6	0.66	1.48
Alt. Alloc. 1	4	0.78	3.52
Alt. Alloc. 2	2	0.88	5.61

Best allocation algorithm predicted by RMTDP also performs the best in the domain

Conclusion

- Formalize the role-allocation problem as a multi-agent POMDP
- Simplify the domain down to crucial variables and triggers
- Use pruning strategies to reduce number of nodes consider
- Abstract method for evaluating the utility of different role allocation strategies