Partial Plan Development for Hierarchical POMDPs

Philipp Heeg

19. November 2013

Contents

1. Motivation and Introduction

Planning with Uncertainty and Partial Observability

- POMDPs can be used to model partially observable, uncertain domains, but solving them is PSPACE-complete
- in Hierarchical POMDPs, expert knowledge is introduced to optimize planning

Partial Plan Development

- usually, the whole plan is developed before execution is started
  $\Longrightarrow$ all eventualities have to be accounted for
- Idea: alternating between partial execution and planning, so the information gained in execution can be used to guide further planning

2.1 POMDPs and FSCs

POMDP: Partially Observable Markov Decision Process
A POMDP has 7 components: S, A, O, T, Z, R, H

- a finite set of states S
- a finite set of actions A
- a finite set of observations O
- a transition function T with $T(s, a, s') \in [0, 1]$
- an observation function Z with $Z(s, a, o) \in [0, 1]$
- a reward function R with $R(s, a) \in \mathbb{R}$
- a horizon $H \in \mathbb{N}$

Solution: A policy that maximizes the total expected reward

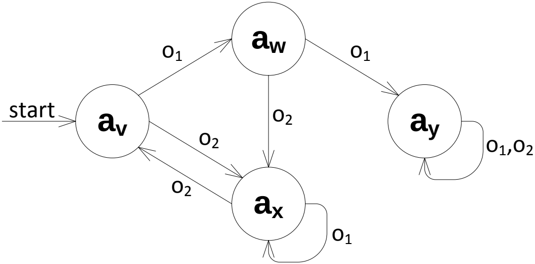
2.1 POMDPs and FSCs

FSC: Finite State Controller

Policy-representation that uses internal states instead of belief states

An FSC has 3 components: N, α, δ

- a set of controller-nodes N
- an action association function α with $\alpha(n) \in A$
- a transition function δ with $\delta(n, n') \in 2^O$



2.2 HPOMDPs and PAFSCs

HPOMDP: Hierarchical POMDP

Extension to POMDPs that allows for exploitation of expert knowledge:

- a new set of abstract actions A^a
- a new set of abstract observations O^a

2.2 HPOMDPs and PAFSCs

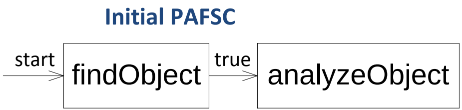
HPOMDP: Hierarchical POMDP

Extension to POMDPs that allows for exploitation of expert knowledge:

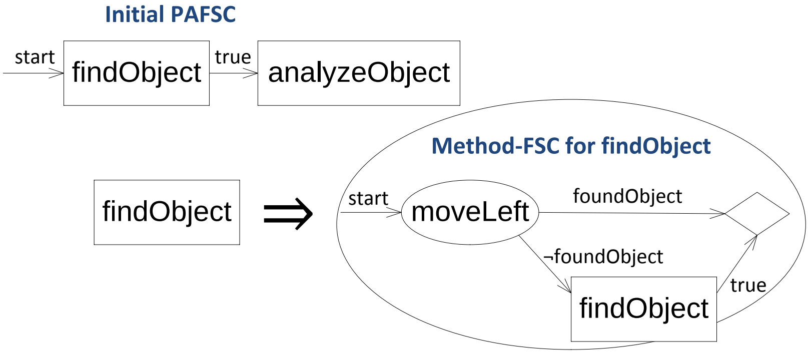
- a new set of abstract actions A^a
- a new set of abstract observations O^a

PAFSC: Partially abstract FSC

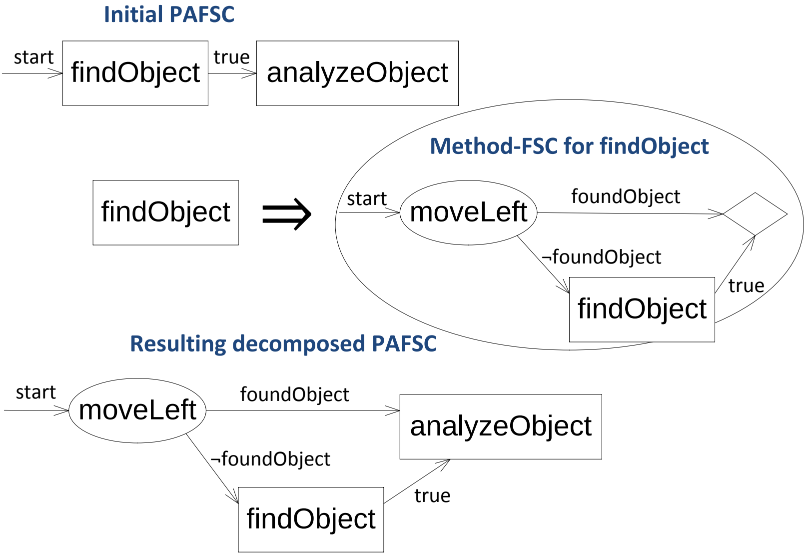
- controller-nodes can be associated with either primitive or abstract actions.
- abstract nodes can be decomposed using Method-FSCs (MFSCs)


Initial PAFSC
start
findObject
true
analyzeObject

2.2 HPOMDPs and PAFSCs



2.2 HPOMDPs and PAFSCs



2.3 The UCT-Algorithm

UCT: Upper Confidence Bound for Trees

- UCT is an instance of Monte Carlo Tree Search (MCTS)
- MCTS builds a partial search tree by interacting with a domain simulator

2.3 The UCT-Algorithm

UCT: Upper Confidence Bound for Trees

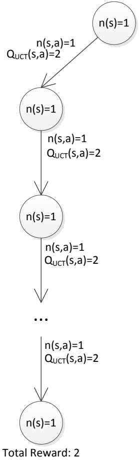
- UCT is an instance of Monte Carlo Tree Search (MCTS)
- MCTS builds a partial search tree by interacting with a domain simulator

A domain simulator consists of 4 components:

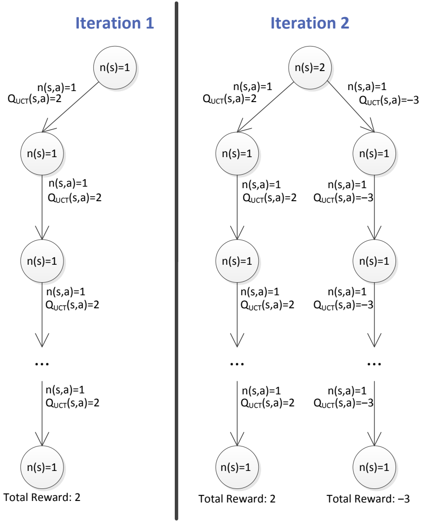
- a set of states S with initial state s_0
- a set of actions A
- a transitionsimulator T
- a rewardsimulator R

2.3 The UCT-Algorithm

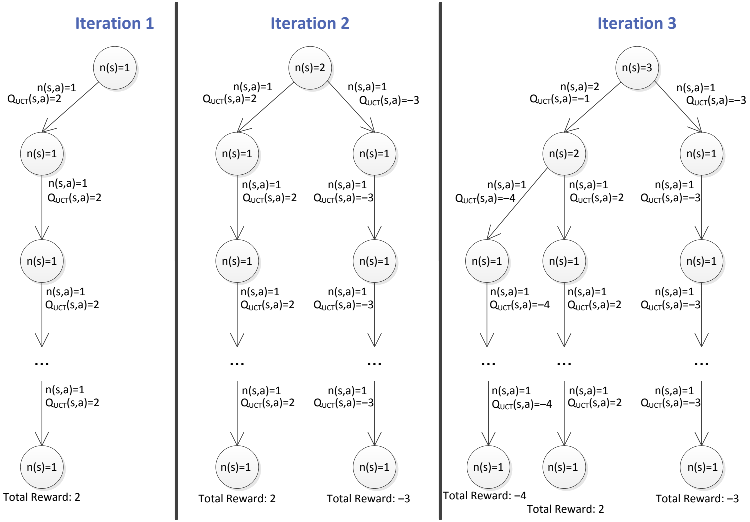
Iteration 1



2.3 The UCT-Algorithm



2.3 The UCT-Algorithm



Applying UCT to HPOMDP problems

- states are reachable PAFSCs
- actions are decompositions
- state transitions are deterministic
- rewards are generated by simulating an execution of the final primitive FSC

2.3 The UCT-Algorithm

Applying UCT to HPOMDP problems

- states are reachable PAFSCs
- actions are decompositions
- state transitions are deterministic
- rewards are generated by simulating an execution of the final primitive FSC
- since order of decompositions is irrelevant, generation of decompositions at each node in the search tree can be limited to 1 controller-node

3.1 Partial Plan Development for HPOMDPs

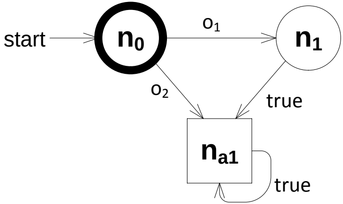
Partial executability of PAFSCs

- PAFSCs can be partially executed until the current controller-node is associated with an abstract action

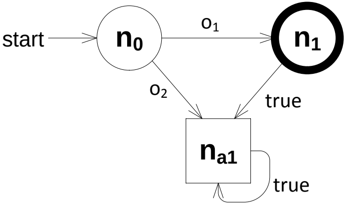
Partial Plan Development: Alternating between an execution phase and a planning phase. Total planning time is distributed over all planning phases.

- Execution phase: partially executing the current PAFSC until an abstract controller-node is reached
- Planning phase: refining the current PAFSC by applying a decomposition to the abstract controller-node that was reached in last execution phase

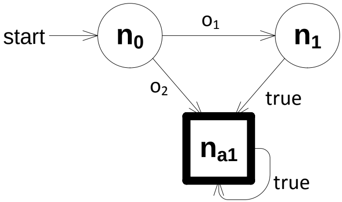
Beginning of execution phase 1



After one executed action

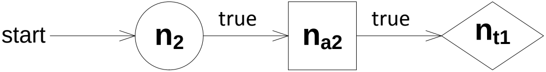


After two executed actions



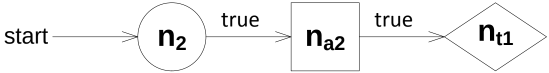
3.1 Partial Plan Development for HPOMDPs

MFSC that was selected in planning phase 1

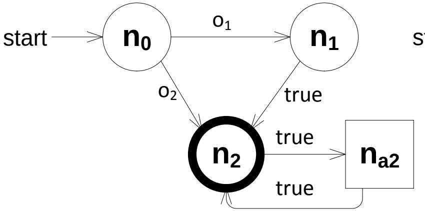


3.1 Partial Plan Development for HPOMDPs

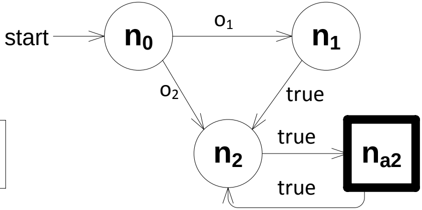
MFSC that was selected in planning phase 1



Beginning of execution phase 2



End of execution phase 2



3.1 Partial Plan Development for HPOMDPs

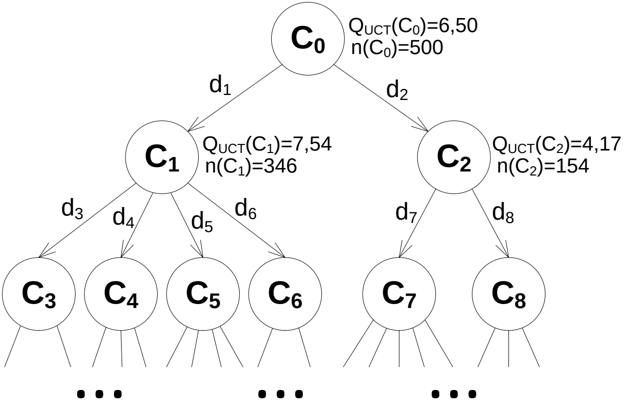
- Goal: higher total reward with same total planning time
- Disadvantage: less time to plan for earlier decompositions
- Advantage: planning specifically for the controller-node that was reached in the last execution phase

3.2 Partial Plan Development with UCT

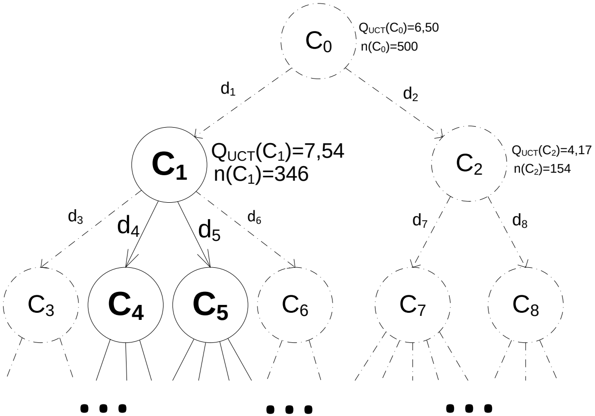
- Idea: reusing the same search tree over all planning phases
- in each planning phase, the latest PAFSC is used as root node
- only one decomposition applied at plan extraction
- as first decomposition in each planning phase, only decompositions for the abstract node that was reached in the latest execution phase are allowed

3.2 Partial Plan Development with UCT

First Planning Phase

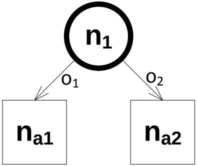


Second Planning Phase



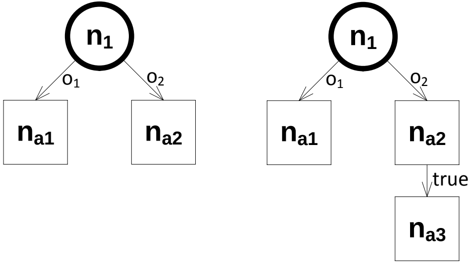
3.2 Partial Plan Development with UCT

- Problem: limiting decompositions to 1 abstract controller-node for each tree-node



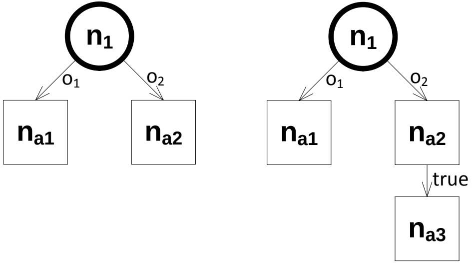
3.2 Partial Plan Development with UCT

- Problem: limiting decompositions to 1 abstract controller-node for each tree-node



3.2 Partial Plan Development with UCT

- Problem: limiting decompositions to 1 abstract controller-node for each tree-node



- Therefore: allow decompositions for all abstract controller-nodes for which there's a primitive path from the initial controller-node

4 Evaluation

- comparing partial planner to non-partial planner (Christian Späth Master Thesis), using the same total planning time
- for the partial planner, the total time Z is distributed over the planning phases by a geometric series:

$$t(n) = (1-q)q^{n-1}Z$$

- 3 different evaluation domains with several instances each

Results for the Reconnaissance domain, 100s planning time

	non-partial	$q = 0.1$	$q = 0.3$	$q = 0.5$	$q = 0.7$	$q = 0.9$	Ø
Instance 1	0.58	0.52	0.52	**0.64**	0.52	0.64	0.57
Instance 2	1.81	1.16	1.64	1.27	**1.66**	1.49	1.45
Instance 3	0.92	0.67	**1.23**	0.69	0.56	1.09	0.85
Instance 4	0.77	0.54	**0.95**	0.5	0.65	0.73	0.67
Instance 5	0.92	0.8	0.93	1.01	0.43	**1.17**	0.87
Instance 6	1.66	1.13	0.52	0.52	0.32	**1.76**	0.85
Instance 7	0.88	0.47	0.41	**0.81**	0.68	0.49	0.57
Instance 8	0.39	0.24	0.16	0.24	**0.4**	0.26	0.26
Instance 9	0.61	**0.7**	0.58	0.4	0.5	0.26	0.49
Instance 10	0.66	0.79	0.69	0.48	**0.83**	0.54	0.66
$Ø_w$	1.28	0.99	1.02	0.92	0.96	1.11	1

5 Summary

- Partial Plan Development for Hierarchical POMDPs using the UCT-Algorithm
- alternating between execution phases and planning phases
- same UCT-Tree is used for all planning phases, with changing root node
- branching factor had to be increased to allow for directed plan development
- therefore slightly worse performance in the Reconnaissance domain compared to non-partial planner

2.3 The UCT-Algorithm

- MCTS uses a Tree policy and a Rollout policy for selecting Actions during Simulations.
- UCT uses an adapted Upper Confidence Bound Algorithm as Tree policy. The selected Action a^* is determined by:

$$a^* = \arg\max_{a \in A} \left[Q_{UCT}(s,a) + c\sqrt{\frac{\ln n(s)}{n(s,a)}} \right]$$

- $Q_{UCT}(s,a) =$ average Simulation Reward when a was selected in s
- $n(s) =$ number of visits of s in previous Simulations
- $n(s,a) =$ number of times a was executed in s in previous Simulations