Controller Compilation and Compression for Resource Constrained Applications

Marek Grześ, Pascal Poupart and Jesse Hoey

UNIVERSITY OF WATERLOO

International Conference on Algorithmic Decision Theory

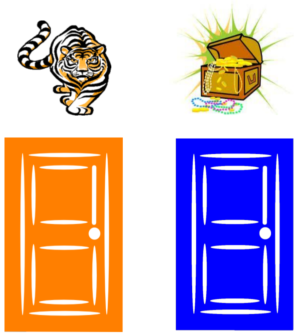
Bruxelles, November 13–15, 2013

POMDP

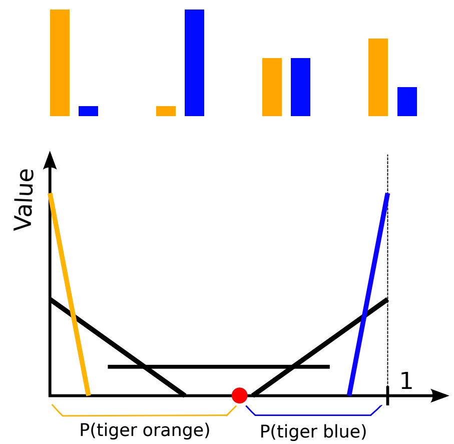
Partially Observable Markov Decision Process

- a discrete time, dynamical system with controls (actions)
- a policy of action optimises a utility function
- the state of the system is partially observable through noisy sensors

POMDP—Example



POMDP—Example



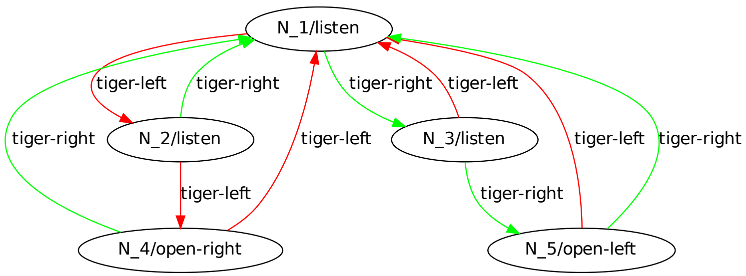
- 2 states (tiger left or right)
- 2 noisy observations (tiger left or tiger right)
- 3 actions (listen, open left, and open right)

Policy Execution

b ← an initial belief
while (true)
  action ← determine an action for b using alpha vectors
  execute the action
  read observations from sensors
  update b using observations and the action
end

Policy Execution in the Cloud

Finite State Controllers



NB: No need to do a belief update nor to consult alpha-vectors.

Battery Efficiency Experiment

- A POMDP, lacasa4.batt, with 2880 states, 72 observations, and 6 actions (engineered using WEB-SNAP[1])
- Installed on a smart phone to assist a cognitively disabled person; allows the patient enjoy her walk, helps find way home when lost, or calls a care giver when necessary
- When the patient is lost and the phone runs out of battery, the application is not available
- Relying on cloud computing requires a reliable network connection and server infrastructure

[1] https://bitbucket.org/mgrzes/web-snap

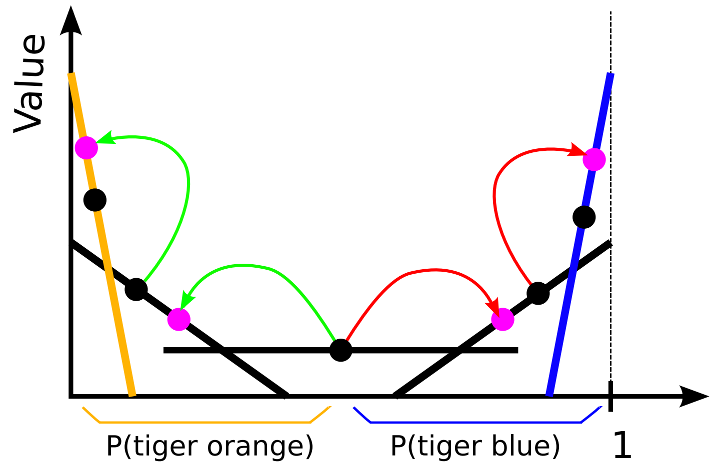
Battery Efficiency on the Nexus 4 Phone

Experiment	1% battery depletion time (minutes)	standard error
OS only	6.74	0.13
OS with WIFI	6.69	0.08
Observation generator	6.40	0.26
Constant policy	5.82	0.21
FSC	5.71	0.18
Client/Server (cloud)	5.52	0.22
Flat policy	4.10	0.11
Symbolic Perseus	3.91	0.15

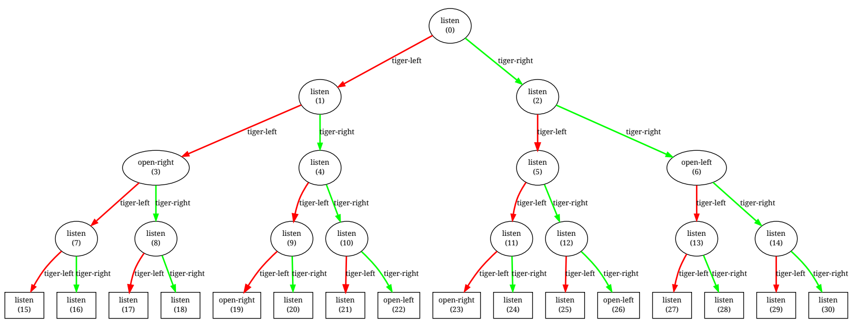
Policies for lacasa4.batt evaluated on the Nexus 4 phone. Every policy was queried once per second (time interval 1 second).

Thanks to Xiao Yang for help with running the battery consumption experiment on the phone.

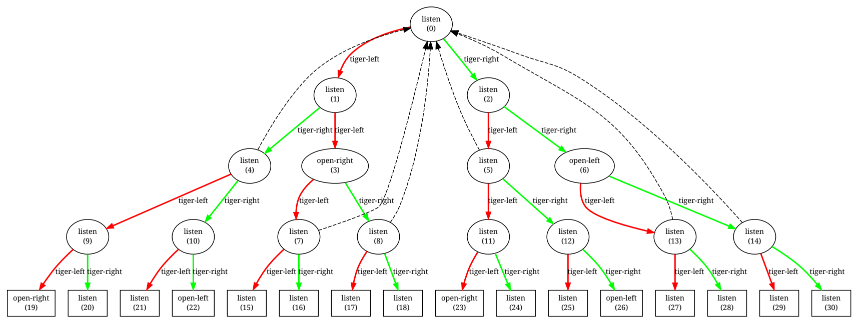
Controller Compilation Using Alpha Vectors



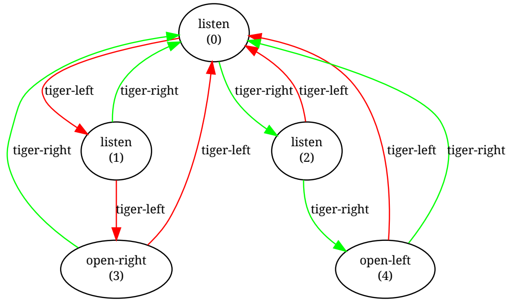
Controller Compilation Using Policy Trees: (1) Policy Tree



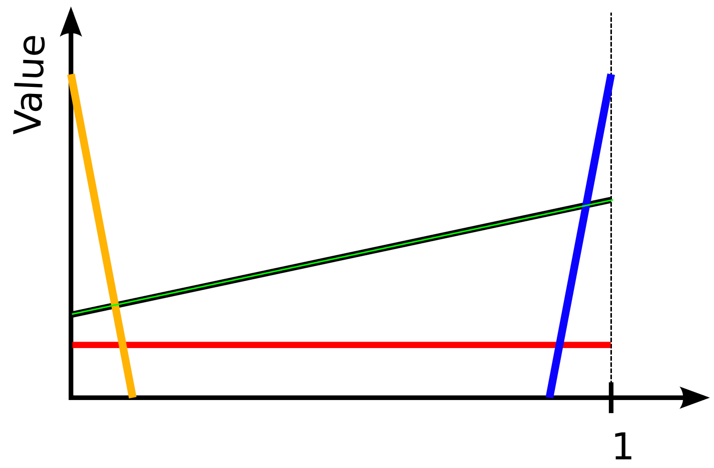
Controller Compilation Using Policy Trees: (2) Identical Conditional Plans



Controller Compilation Using Policy Trees: (3) Removing Repeated Nodes



Controller Compilation: More Pruning of Redundant Nodes



Controller Compilation Results (1)

POMDP	GapMin	method	depth	tree size	nodes	value	time	c
chainOfChains3	GM-lb=157	**alpha2fsc**	10	10	10(10)	**157**(157)	0.26	0
$\|S\|$=10, $\|A\|$=4	GM-ub=157	**GM-LB**	11	11	10(10)	**157**(157)	0.42	0
$\|O\|$=1, $\gamma = 0.95$	time=0.86s	**GM-UB**	11	11	10(10)	**157**(157)	0.26	0
	$\|lb\|$=10	**B&B**			10	**157**	1.69	
	$\|ub\|$=1	EM			10	0.17 ± 0.06	6.9	
		QCLP			10	0 ± 0	0.16	
		BPI			10	25.7 ± 0.77	4.25	
cheese-taxi	GM-lb=2.481	**alpha2fsc**			17(22)	**2.476**(2.476)	0.29	1
$\|S\|$=34, $\|A\|$=7	GM-ub=2.481	**GM-LB**	15	167	17(24)	**2.476**(2.476)	0.56	1
$\|O\|$=10, $\gamma = 0.95$	time=1.88s	**GM-UB**	15	167	17(24)	**2.476**(2.476)	0.55	1
	$\|lb\|$=22	B&B			10	-19.9*	24h	
	$\|ub\|$=13	EM			17	-12.16 ± 2.08	337.9	
		QCLP			17	-18.22 ± 1.77	227.4	
		BPI			16	-18.1 ± 0.39	7.18	
lacasa4.batt	GM-lb=291.1	alpha2fsc			10(10)	285.5(285.5)	302	0
$\|S\|$=2880, $\|A\|$=6	GM-ub=292.6	GM-LB	3	745	19(22)	287.3(287.1)	3652	1
$\|O\|$=72, $\gamma = 0.95$	time=8454s	**GM-UB**	4	23209	87(94)	**290.8**(290.8)	3681	1
	$\|lb\|$=10	B&B			10	285.0*	24h	
	$\|ub\|$=23	EM			3	290.2 ± 0.0	19920	
		BPI			6	290.6 ± 0.2	4124	
machine	GM-lb=62.38	alpha2fsc			5(39)	54.61(54.09)	5.53	1
$\|S\|$=256, $\|A\|$=4	GM-ub=66.32	GM-LB	9	376	26(41)	62.92(62.84)	18.5	1
$\|O\|$=16, $\gamma = 0.99$	time=3784s	**GM-UB**	12	2864	11(159)	**63.02**(60.29)	86.8	2
	$\|lb\|$=39	B&B			6	62.6	52100	
	$\|ub\|$=243	EM			11	62.93 ± 0.03	1757	
		QCLP			11	62.45 ± 0.22	4636	
		BPI			10	35.7 ± 0.52	2.14	

Controller Compilation Results (2)

POMDP	SARSOP	method	depth	tree size	nodes	value	time	c
baseball	time 122.7s	**policy2fsc**	7	175985	10(47)	**0.641**(0.641)	78.22	1
$\|S\|$=7681, $\|A\|$=6	$\|\alpha\|$ =1415	B&B			5	0.636*	24h	
$\|O\|$=9, $\gamma = 0.999$	UB=0.642	EM			2	0.636 ± 0.0	48656	
	LB=0.641	BPI			9	0.636 ± 0.0	445	
elevators_inst_pomdp_1	time 11,228s	**policy2fsc**	11	419	20(24)	**-44.41**(-44.41)	1357	1
$\|S\|$=8192, $\|A\|$=5	$\|\alpha\|$ =78035	B&B			10	-149.0*	24h	
$\|O\|$=32, $\gamma = 0.99$	UB=-44.31							
	LB=-44.32							
tagAvoid	time 10,073s	**policy2fsc**	28	7678	91(712)	**-6.04**(-6.04)	582.2	1
$\|S\|$=870, $\|A\|$=5	$\|\alpha\|$ =20326	B&B			10	-19.9*	24h	
$\|O\|$=30, $\gamma = 0.95$	UB=-3.42	EM			9	-6.81 ± 0.12	19295	
	LB=-6.09	QCLP			2	-19.99 ± 0.0	12.9	
		BPI			88	-12.42 ± 0.13	1808	
underwaterNav	time 10,222s	policy2fsc	51	1242	52(146)	745.3(745.3)	5308	1
$\|S\|$=2653, $\|A\|$=6	$\|\alpha\|$ =26331	B&B			10	747.0*	24h	
$\|O\|$=103, $\gamma = 0.95$	UB=753.8	**EM**			5	**749.9** ± 0.02	31611	
	LB=742.7	BPI			49	748.6 ± 0.24	14758	
rockSample-7_8	time 10,629s	**policy2fsc**	31	2237	204(224)	**21.58**(21.58)	1291	1
$\|S\|$=12545, $\|A\|$=13	$\|\alpha\|$ =12561	B&B			10	11.9*	24h	
$\|O\|$=2, $\gamma = 0.95$	UB=24.22	BPI			5	7.35 ± 0.0	78.8	
	LB=21.50							

Table: Compilation and compression of SARSOP policies.

Conclusion

1. Execution of POMDP policies on mobile devices can be battery consuming
2. Finite-state controllers are computationally cheap to execute
3. Two methods to compile POMDP policies into finite-state controllers were shown
4. Compilation is more robust against local optima than direct optimization with local search
5. Compilation is feasible on large problems where exhaustive search with branch-and-bound can be challenging