文章编号: 1001-0920(2007)12-1417-04

一种基于信念状态压缩的实时 POMDP 算法

作 博[1,2], 吴 敏[1]
(1. 中南大学 信息科学与工程学院, 长沙 410083; 2. 深圳职业技术学院 计算机应用工程系, 广东 深圳 518055)

摘 要: 针对求解部分可观察马尔可夫决策过程(POMDP)信念状态空间是 NP 难问题,提出一种信念状态空间压缩(BSSC)算法. 将信念状态空间的高维压缩到低维,利用动态贝叶斯网络对状态转移函数、观察函数和报酬函数进行压缩,降低求解规模,达到实时决策的目的. 对比实验表明,所提出的算法可以快速求解最优策略和最优值函数.

关键词: 马尔可夫; 可观察马尔可夫决策过程; 决策算法; 决策树
中图分类号: TP393 **文献标识码**: A

Real-time POMDP algorithm based on belief states space compression

WU Bo[1,2], *WU Min*[1]
(1. School of Information Science and Engineering, Central South University, Changsha 410083, China; 2. Department of Computer Application Engineering, Shenzhen Polytechnic Institute, Shenzhen 518055, China. Correspondent: WU Bo, E-mail: wubo@oa.szpt.net)

Abstract: Solving belief state space for partially observable Markov decision processes (POMDP) is an NP-difficult problem. Therefore, a belief state space compression (BSSC) algorithm is proposed, which compacts belief state space from high dimension to low dimension. State transition function, observation function and reward function are compressed by using dynamic Bayesian network to reduce the solving dimension and realize real-time decision. The test data show that the BSSC algorithm can quickly solve optimal policy and optimal value function in the real-time environment.

Key words: Markov; POMDP; Decision algorithm; Decision tree

1 引 言

目前关于部分可观察马尔可夫决策过程(Partially Observable Markov Decision Processes, POMDP)的研究十分活跃,人们提出了一些解决 POMDP 问题的算法. 例如:策略迭代算法[1](Policy Iteration, PI),梯度法[2](Gradient Ascent, GA),干枝界限法(Branch and Bound, B &B)和随机局部查找法(Stochastic Local Search, SLS)等. PI 算法需求解线性方程组,其规模是一个指数函数; GA 算法难以求解整体最优; B &B 算法和 SLS 算法受限于计算时间而难以应用到大规模问题. Witness 算法[3]、Incremental Pruning 算法和 Point-based 算法[4]虽然可以解决有限时间问题,但对于无限时间问题却难以求出最优策略和最优值函数[5].

本文提出一种基于信念状态空间压缩的 POMDP 算法 Belief State Space Compression (BSSC). 该算法将信念状态空间进行压缩,从高维降到低维,以压缩求解规模,达到实时决策的目的.

2 POMDPs

在 POMDP 模型中,Agent 必须利用随机环境中部分观察到的信息进行决策. 在每个时间点上,Agent 都可能是众多可能状态中的某一状态,但由于观察到的信息不完整或不可能直接知道自己的当前状态,它必须利用现有的部分信息、历史动作序列和立即报酬值来采用一种策略. 一般情况下,POMDP 可用 1 个 6 元组$\langle S, A, T, R, \Omega, O\rangle$来描述[3]. 其中:

S 表示 Agent 的状态集,包含 Agent 所有可能处在的状态;

A 表示 Agent 的行为集;

收稿日期: 2006-09-11; **修回日期**: 2006-11-20.
基金项目: 国家十五 863 计划项目(2001AA4422200).
作者简介: 仵博(1979—),男,河南桐柏人,讲师,硕士,从事多智能体系统的研究; 吴敏(1963—),男,广东化州人,教授,从事先进控制理论与应用、工业过程智能集成控制技术等研究.

Ω表示 Agent 的可观察信息集;

B 表示 Agent 的信念状态空间,用 $b(s)$ 描述 Agent 处在 s 状态的概率;

$T(s,a,s')$ 表示 $T: S\times A\times S\to S$,Agent 的状态转移函数,当 Agent 在状态 s 下采用动作 a 时可能转移到状态 s' 的概率,用 $\Pr(s'\mid s,a)$ 表示;

$O(s',a,o)$ 表示 $O: S\times A\times O\to O$,Agent 的观察函数,它可以计算出采用动作 a 后在下一个状态 s' 时的可能观察值,用 $\Pr(o\mid s',a)$ 表示;

$R(s,a)$ 表示 $R: S\times A\to R$,Agent 的报酬函数,是在状态 s 情况下采用动作 a 时返回给系统的报酬值;

π表示 $B(S)\to A$,Agent 的行为策略,使 Agent 选择的动作能够获得环境报酬的累计值最大.

如果想知道状态 s' 下的信念状态 b',则可根据信念状态 b,行为 a 和观察 o 得到,具体过程根据贝叶斯计算如下:

$$b[s'] = \Pr(s'\mid o,a,b) = \frac{O(s',a,o)\sum_{s\in S}T(s,a,s')b(s)}{\sum_{s'\in S}O(s',a,o)\sum_{s\in S}T(s,a,s')b(s)}. \tag{1}$$

采用以前的观点解决 POMDP 问题时,由于必须知道历史动作才能决定当前的动作,这种解决方案是非马尔可夫链. 然而,当引入信念状态空间后,POMDP 问题就可以转化为基于信念状态空间的马尔可夫链问题来求解. 通过信念状态空间的引入,POMDP 问题可以看成 Belief MDP 问题[5]. 寻求一种最优策略将当前的信念状态映射到 Agent 的行动上,根据当前的信念状态和行为就可以决定下一个周期的信念状态和行为,具体描述如下[3]:

1) $\tau(b,a,b')$:$B\times A\times B\to B$:状态转移函数,定义如下:

$$\tau(b,a,b') = \Pr(b'\mid a,b) = \sum_{o\in O}\Pr(b'\mid a,b,o)\Pr(o\mid a,b); \tag{2}$$

2) $\rho(b,a)$:信念状态报酬函数,定义如下:

$$\rho(b,a) = \sum_{s\in S}b(s)R(s,a). \tag{3}$$

简单而言,POMDP 决策的策略是将信念状态映射到动作,同时满足使 Agent 选择的动作能够获得环境报酬的累计值最大,即

$$E\Big[\sum_{t}^{\infty}\gamma^t\rho_t\Big],\ \gamma\in(0,1]. \tag{4}$$

其中 γ为折扣因子,其目标是让期望值收敛.

由于 POMDP 决策过程中要同时考虑世界模型的不确定性和目标的长远性,需要在策略 π与报酬值之间构造值函数进行决策. t 时刻,在信念状态 b 和策略 π的情况下,根据 Bellman 原理,贪婪策略和贪婪值函数构造如下:

$$\pi_t^*(b) = \arg\max_a\Big[\sum_{s\in S}b(s)R(s,a) + \gamma\sum_{o\in O}\Pr(o\mid b,a)V_t^*(b')\Big], \tag{5}$$

$$V_t^*(b) = \max_a\Big[\sum_{s\in S}b(s)R(s,a) + \gamma\sum_{o\in O}\Pr(o\mid b,a)V_{t-1}^*(b')\Big]. \tag{6}$$

3 一种基于信念状态空间压缩的实时 POMDP 算法

3.1 信念状态空间压缩

传统求解 POMDP 的算法都受限于信念状态空间的规模,因而降低信念状态空间的规模是求解 POMDP 问题的必然要求. 在 POMDP 中,可使用一个有限变量集合来描述真实世界[6-8],有限变量可以是布尔值,也可以是二进制的 0 和 1. 令 $X=\{X_1,X_2,\cdots,X_n\}$,对每个 X_i 赋值可描述一种状态,例如 $s=\{X_1=x_1,X_2=x_2,\cdots,X_n=x_n\}$,则 $2^{|X|}$ 可描述所有的状态. 对于观察模型,可以使用 $Y=\{Y_1,Y_2,\cdots,Y_m\}$,则 $2^{|Y|}$ 可描述所有的观察. 对于动作模型,可使用 $Z=\{Z_1,Z_2,\cdots,Z_l\}$,则 $2^{|Z|}$ 可描述所有的动作. 根据状态的分解,信念状态 $b(s)$ 可以被描述为 $b=\Pr(X_1,X_2,\cdots,X_n)$,$X_1,X_2,\cdots,X_n$ 是独立分布的[8],所以 $b=\Pr(X_1)\Pr(X_2)\cdots\Pr(X_m)$.

$$b = \begin{bmatrix}\begin{bmatrix}0\\0\\0.5\\0\\0\\0.5\end{bmatrix}\begin{bmatrix}0.2\\0\\0\\0.8\\0\\0\end{bmatrix}\begin{bmatrix}1.0\\0\\0\\0\\0\\0\end{bmatrix}\begin{bmatrix}0.1\\0.2\\0\\0.4\\0\\0.3\end{bmatrix}\end{bmatrix}, \tag{7}$$

$$b_{\text{compress}} = \begin{bmatrix}\begin{bmatrix}0.5\\0.5\end{bmatrix}\begin{bmatrix}0.2\\0.8\end{bmatrix}[1.0]\begin{bmatrix}0.1\\0.2\\0.4\\0.3\end{bmatrix}\end{bmatrix}. \tag{8}$$

假如一个世界模型可以使用 4 个变量 X_1,X_2,X_3 和 X_4 描述,每个变量有 6 种不同的概率值,如式(7)所示,则 b 有 $6\times6\times6\times6=1\,296$ 个信念状态. 如果概率是 0,表示不可能存在的状态,则将这些 0 值去掉,信念状态压缩后变为 b_{compress},有 $2\times2\times1\times4=16$ 个信念状态,如式(8)所示. 这样,利用可分解表示法可以大大压缩信念状态空间.

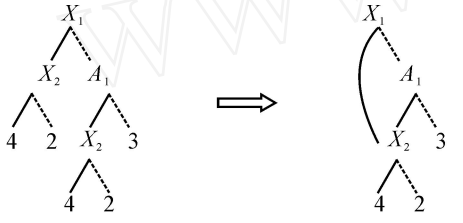
利用动态贝叶斯网络(DBN)[6] 可以对状态转移函数 $T(s,a,s')$,观察函数 $O(s',a,o)$ 和报酬函数

$R(s,a)$ 进行压缩,表 1 描述了整个压缩过程. 例如:当 $T=t$ 时,世界状态模型由 X_1 和 X_2 描述,动作用 A_1 描述,则当 $T=t+1$ 时,状态转移到 X_1' 和 X_2',得到观察 O_1',O_2 和报酬函数 R.

表 1　DBN 压缩过程

X_1	X_2	A_1	X_1'	X_2'	O_1'	O_2'	R
x_1	x_2	a_1	0.3	0.7	0.1	0.9	4
x_1	x_2	$\bar{a}_1$	0.6	0.4	0.2	0.8	4
x_1	$\bar{x}_2$	a_1	0.2	0.8	0.4	0.6	2
x_1	$\bar{x}_2$	$\bar{a}_1$	0.7	0.3	0.8	0.2	2
$\bar{x}_1$	x_2	a_1	0.8	0.2	0.7	0.3	4
$\bar{x}_1$	x_2	$\bar{a}_1$	0.6	0.4	0.9	0.1	3
$\bar{x}_1$	$\bar{x}_2$	a_1	0.3	0.7	0.6	0.4	2
$\bar{x}_1$	$\bar{x}_2$	$\bar{a}_1$	0.1	0.9	0.3	0.7	3

对于 DBN 计算的报酬函数值,可使用决策树(DT)[6] 进行再次压缩,如图 1(a) 所示. 在原 DBN 中 R 具有 8 个值,用 DT 描述则只需要 5 个叶子节点. 对于 DT 图,可以使用代数决策图(ADD)[6] 进行再次压缩,将 DT 图中两个 X_2 合并,变成图 1(b). 这样,叶子节点就变成 3 个,大大降低了求解规模.



(a) 压缩前　　(b) 压缩后

图 1　报酬函数压缩过程

3.2　BSSC 算法

由上节可知,通过将状态、观察和动作进行分解,转化为相应的变量表示可大大降低信念状态空间. 使用 DBN,DT 和 ADD 也可以将状态转移函数、观察函数和报酬函数进行压缩,降低求解的规模. 利用分解 POMDP 表示法,定义状态、观察和动作的表示函数 $\beta(b)$,$\delta(b)$ 和 $\alpha(b)$. $\beta(b)$ 是状态 S 的子集,$\delta(b)$ 是观察 Ω 的子集,$\alpha(b)$ 是观察 A 的子集,即 $\beta(b)\subseteq S$,$\delta(b)\subseteq\Omega$,$\alpha(b)\subseteq\Omega$.

$$\beta(b)=\{\{x_i\}_{i=1}^{n}\mid(\forall x_i)\,P_b(X_i=x_i)\neq 0\},\quad(9)$$

$$\delta(b)=\{\{y_i\}_{i=1}^{m}\mid(\forall y_i)\,P_b(Y_i=y_i)\neq 0\},\quad(10)$$

$$\alpha(b)=\{\{z_i\}_{i=1}^{l}\mid(\forall z_i)\,P_b(Z_i=z_i)\neq 0\}.\quad(11)$$

决策时只需考虑概率为非 0 的状态、观察和动作,根据新的集合,利用式(12) 可计算出新的 S',利用式(13) 计算出新的 O'. 同时,根据更新的状态和观察,可利用式(14) 计算出新的信念状态,利用式(15) 和(16) 计算出新的策略值和函数值.

$$\sigma(a,b,0)=\{s'\mid(\forall s\in\beta(b),\forall a\in\alpha(b))(T(s,a,s')\neq 0)\},\quad(12)$$

$$\tau(a,b,0)=\{o'\mid(\forall s\in\beta(b),\forall o\in\delta(b),\forall a\in\alpha(b))(O(s',a,o)\neq 0)\},\quad(13)$$

$$b'(s')=\frac{O(s',a,o)\sum_{s\in\beta(b)}T(s,a,s')b(s)}{\sum_{s'\in\sigma(a,b,o)}O(s',a,o)\sum_{s\in\beta(b)}T(s,a,s')b(s)},\quad(14)$$

$$\pi_t^*(b)=\arg\max_{a\in\alpha(b)}\Big[\sum_{s\in\beta(b)}b(s)R(s,a)+\gamma\sum_{o\in\delta(b)}\Pr(o\mid b,a)V_t^*(b')\Big],\quad(15)$$

$$V_t^*(b)=\max_{a\in\alpha(b)}\Big[\sum_{s\in\sigma(a,b,o)}b(s)R(s,a)+\gamma\sum_{o\in\delta(b)}\Pr(o\mid b,a)V_{t-1}^*(b')\Big].\quad(16)$$

算法 1 (BSSC 算法)

1) Initialize $V_1(s)=0$, $t=1$, $s\in S$, $a\in A$, $b\in B$;

2) Compress $S:\beta(b)$;

3) Compress $O:\delta(b)$;

4) Compress $A:\alpha(b)$;

5) while(true) {

6)　$V_t(b)=\arg\max_{a\in\alpha(b)}\Big[\sum_{s\in\beta(b)\wedge s\in\sigma(a,b,o)}b(s)R(s,a)+\gamma\sum_{o\in\delta(b)\wedge o\in\tau(a,b,o)}\Pr(o\mid b,a)V_{t-1}(b')\Big]$;

7)　$t=t+1$;

8)　if $\Big(\|V_t-V_{t-1}\|<\frac{\xi(1-\gamma)}{2\gamma}\Big)$ {

9)　　break;

10)　}

11) }

12) $V_t^*(b)=\max_{a\in\alpha(b)}\Big[\sum_{s\in\beta(b)\wedge s\in\sigma(a,b,o)}b(s)R(s,a,)+\gamma\sum_{o\in\delta(b)\wedge o\in\tau(a,b,o)}\Pr(o\mid b,a)V_{t-1}(b')\Big]$.

本文算法是先将求解规模进行压缩;然后根据动态规划的思想求解最优策略和最优值函数.

4　对比实验

算法 1(BSSC) 的测试对象为:Tiger-Grid,

Hallway, Tag 和 RockSample.Benchmark 平台[4] 中提供了几种经典算法:QMDP, PBVI 和 HSVI. 本文从运行时间和解决问题质量两方面与以上 3 种算法进行比较[6,8]. 实验环境是:Windows XP, CPU 2 GHz, 内存 512 M.

表 2 对比实验

问题	算法	报酬期望	计算时间	问题规模
Tiger-Grid	BSSC	3.21	561	S = 36
	QMDP	0.20	0.3	A = 5
	PBVI	2.35	3 501	O = 17
	HSVI	2.44	10 809	
Hallway	BSSC	0.57	112	S = 61
	QMDP	—	—	A = 5
	PBVI	0.51	288	O = 21
	HSVI	0.53	10 988	
RockSample[4,4]	BSSC	22.2	286	S = 257
	QMDP	—	—	A = 9
	PBVI	17.1	1 975	O = 2
	HSVI	18.0	578	
Tag	BSSC	- 5.91	521	S = 870
	QMDP	- 16.97	13.89	A = 5
	PBVI	- 9.21	180 880	O = 30
	HSVI	- 6.57	10 113	

从表 2 可知,本文算法在计算时间和效果上都具有较明显的优势,特别是对于较大规模问题,本文算法的计算效果更为明显.

图 2 描述的是当 $|S|=7$, $|O|\in[5,17]$ 时,计算 POMDP 问题所需要的时间. 从图 2 可以看出,本文算法和 Two-Pass 算法是比较可行的算法,而 Witness 算法需要的计算时间较多. 图 3 描述的是当 $|Z|=7$, $|S|\in[5,17]$ 时,计算 POMDP 问题所需要的时间,从图3可以看出, Two-Pass算法在某种程度上与本文算法相当, 但当状态很大时, Two-Pass 算法计算效率将会大大降低.

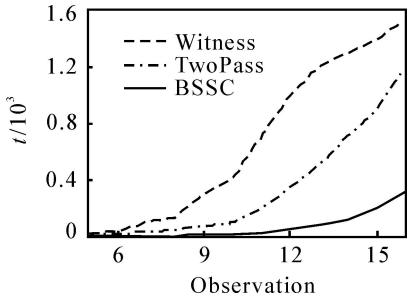


图 2 当 $|S|=7$ 时的计算时间

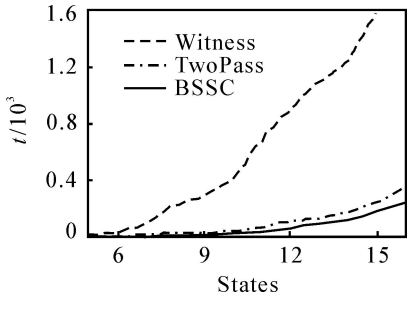


图 3 当 $|Z|=7$ 时的计算时间

5 结 语

本文提出了一种基于信念状态空间压缩的算法. 该算法利用可分解特性,将信念状态空间进行压缩,从高维降到低维,压缩求解规模. 首先利用动态贝叶斯网络对状态转移函数、观察函数和报酬函数进行压缩;然后使用决策树和代数决策图将报酬函数进行压缩,从而减少求解规模. 对比实验显示,本文算法在计算时间和求解效果上都具有较好的优势. 将本文算法分别与 Witness 算法和 Two-Pass 算法进行比较,实验数据表明,本文算法是一种解决 POMDP 问题可行的算法.

参考文献(References)

[1] S'ebastien Paquet, Ludovic Tobin, Brahim Chaibdraa. An online POMDP algorithm for complex multi-agent environment[C]. Proc of the 4th Int Joint Conf on Autonomous Agents and Multi Agent Systems (AAMAS-05). Netherlands: Utrecht University, 2005: 970-977.

[2] Nicolas Meuleau, Leonid Peshkin, Kee-Eung Kim, et al. Learning nite-state controllers for partially observable environments[C]. Proc of the 5th Conf on Uncertainty in Artificial Intelligence. Stockholm, 1999: 427-436.

[3] Leslie Pack Kaelbling, Michael L Littman, Anthony R Cassandra. Planning and acting in partially observable stochastic domains[J]. Artificial Intelligence, 1998, 101(l): 99-134.

[4] Pineau J, Gordon G, Thrun S. Point-based value iteration: An anytime algorithm for POMDPs[C]. Int Joint Conf on Artificial Intelligence (IJCAI). Mexico: Acapulco, 2003: 1025-1032.

[5] Darius Braziunas, Craig Boutilier. Stochastic local search for POMDP controllers[C]. Proc of the 19th National Conf on Artificial Intelligence. CA: San Jose, 2004: 690-696.

[6] Poupart P. Exploiting structure to efficiently solve large scale partially observable Markov decision processes [D]. Toronto: University of Toronto, 2005.

[7] Cassandra A R. Exact and approximate algorithms for partially observable Markov decision processes[D]. Rhode Island: Brown University, 1998.

[8] Eric A Hansen, Zhengzhu Feng. Dynamic programming for POMDPs using a factored state representation[C]. 5th Int Conf on Artificial Intelligence Planning and Scheduling. Colorado: Breckenridge, 2000: 130-139.