ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.19, No.6, June 2008, pp.1309−1316
DOI: 10.3724/SP.J.1001.2008.01309

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于点的 POMDP 算法的预处理方法*

卞爱华[1+], 王崇骏[2], 陈世福[2]

[1](南京大学 计算机软件新技术国家重点实验室,江苏 南京 210093)
[2](南京大学 计算机科学与技术系,江苏 南京 210093)

Preprocessing for Point-Based Algorithms of POMDP

BIAN Ai-Hua[1+], WANG Chong-Jun[2], CHEN Shi-Fu[2]

[1](National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093)
[2](Department of Computer Science and Technology, Nanjing University, Nanjing 210093)
+ Corresponding author: E-mail: bianaihua@iip.nju.edu.cn

Bian AH, Wang CJ, Chen SF. Preprocessing for point-based algorithms of POMDP. *Journal of Software*, 2008,19(6):1309−1316. http://www.jos.org.cn/1000-9825/19/1309.htm

Abstract: Point-Based algorithms are a class of approximation methods for partially observable Markov decision processes (POMDP). They do backup operators on a belief set only, so linear programming is avoided and fewer intermediate variables are needed, and the bottleneck turns from selecting vectors to generating vectors. But when generate vectors, there will be a great deal of repeated and meaningless computing. This paper will propose a preprocessing method for point-based algorithms (PPBA). This method preprocesses each sampled belief point, and before generating α-vectors it estimates which action and α-vectors to be selected first, in so doing repeated computing is eliminated. Base-vector is also defined in this paper, which cancels meaningless computing with sparseness of problem. Experiments on Perseus show that, PPBA accelerates the performance greatly.
Key words: POMDP; value iteration; point-based algorithm; preprocessing; base-vector

摘 要: 基于点的算法是部分可观察马尔可夫决策过程(partially observable Markov decision processes,简称 POMDP)的一类近似算法.它们只在一个信念点集上进行 Backup 操作,避免了线性规划并使用了更少的中间变量,从而将计算瓶颈由选择向量转向了生成向量.但这类算法在生成向量时含有大量重复和无意义计算,针对于此,提出了基于点的 POMDP 算法的预处理方法(preprocessing method for point-based algorithms,简称 PPBA).该方法对每个样本信念点作预处理,并且在生成 α-向量之前首先计算出该选取哪个动作和哪些 α-向量,从而消除了重复计算.PPBA 还提出了基向量的概念,利用问题的稀疏性避免了无意义计算.通过在 Perseus 上的实验,表明 PPBA 很大地提高了算法的执行速度.
关键词: POMDP;值迭代;基于点的算法;预处理;基向量
中图法分类号: TP301 **文献标识码**: A

* Supported by the National Natural Science Foundation of China under Grant No.60503021 (国家自然科学基金); the High-Tech Research Program of Jiangsu Province of China under Grant No.BG2006027 (江苏省高技术研究计划)
Received 2007-08-13; Accepted 2007-10-12

在传统的多 Agent 系统策略问题中,Agent 往往是在完全可观察的环境下行动,这导致了许多技术都不适合于实际的应用场景.部分可观察马尔可夫决策过程(partially observable Markov decision processes,简称 POMDP)为不确定环境下的序贯决策问题提供了一个丰富的框架[1,2].在 POMDP 中,系统的状态和决策动作的影响都是不确定的,仅仅可以获得对隐蔽状态的观察,它与状态满足一定的条件概率.

POMDP 自从被提出以来,在人工智能和控制研究领域受到广泛关注[11,12],并且许多精确算法随之被提出来[2−5],它们都是在整个信念空间上最优化值函数.然而,这些算法在运算中都将陷入众所周知的维度和历史问题[2],这导致它们成为典型的 NP-hard 问题(除低维度问题以外).

幸运的是,有这样一个基本事实:最优值函数在连续的信念空间上是一个分段线形凸函数[1],所以邻近点的函数值也近似.基于这个理论,可以用大量的离散信念点来近似整个连续的信念单形体.

文献[4]表明:如果每步中计算得到的新值函数都是上一步值函数的上界,值迭代仍会收敛于最优值函数.并且可以将标准的值迭代步骤与局部值迭代步骤相交叉,从而加速了整个算法.所以,可以用生成一系列上界来取代精确的值迭代.

根据以上两点,基于点的算法被提出来.在基于点的值迭代的算法(PBVI)[6]中,首先获取一个较小的信念点集 B,然后值更新和信念点集扩充交叉执行.在 Perseus 算法[7,8]中,首先让 Agent 在环境中随机探索以获取一个足够大的可达信念点集 B,然后在 B 的一个很小的子集上执行值函数更新,并且确保每步中计算得到的新值函数在 B 中所有点上都是上一步值函数的上界.

另外,最近的研究还利用到了另外一个事实:在大部分实际问题中,信念单形体都是稀疏的.也就是说,如果让 Agent 和环境直接交互,仅有有限的很小部分信念点可以到达.根据这个事实,基于点的 POMDP 解决方法仅仅在这些可达信念点上执行值更新,取代了在整个信念单形体上的规划.

本文提出了一种基于点的算法的预处理方法(preprocessing method for point-based algorithms,简称 PPBA),并且把它应用于 Perseus 算法.该算法不采用值迭代中先生成再选择向量的传统做法,而是借助于对取样信念点集 B 的预处理,先选择而后生成向量,这将节省大量重复计算的时间.更重要的是,在稀疏问题中,在初始信念下执行某个动作后大部分观察并不会被得到,通过预处理可以避免这些无意义计算.实验结果表明,为得到相同的最佳奖赏值,预处理方法仅用了传统算法 1/5 到 1/2 的时间.

本文第 1 节介绍了 POMDP 相关知识以及 Perseus 算法;第 2 节给出了 PPBA 算法的详细内容;第 3 节用实验将 PPBA 和 Perseus 算法进行比较并作分析;第 4 节为结束语.

1 POMDP 背景

1.1 POMDP

马尔可夫决策过程(Markov decision processes,简称 MDP)是一个经典的序贯决策问题(sequential decision problems)的模型,序贯决策问题需要在系统的整个生命周期内作决策.但因为系统总是被假定为完全可观察的,这使决策方法的使用很受限制.因此,一个更普遍的模型部分可观察马尔可夫决策过程(POMDP)被提了出来.

形式上,一个 POMDP 问题被定义为 6 个独立的量[1,9],记为 $\{S,A,Z,T,O,R\}$.S 是系统的状态集,状态不可被直接观察;A 是 Agent 执行的动作的集合,动作将改变系统状态; Z 是 Agent 对系统的观察集,由于被噪音干扰,它是状态的不完全反应;T 是状态转移函数:$S\times A\times S\to[0,1]$,其中,$T(s,a,s')=\Pr(s'|s,a)$表示 Agent 在状态 s 下执行动作 a 后得到状态 s'的概率;O 是观察函数:$A\times S\times Z\to[0,1]$,其中,$O(a,s',z)=\Pr(z|a,s')$表示 Agent 在执行动作 a 后到达状态 s'时得到观察 z 的概率;R 是奖赏函数:$S\times A\to\mathbf{R}$,其中 $R(s,a)$表示 Agent 在状态 s 下执行动作 a 所得到的即时奖赏值.另外,模型中还会有一个初始信念状态 b_0 和折扣因子 γ.其中 $b_0(s)=\Pr(s_0=s)$是在时间 $t=0$ 时在状态集 S 上的概率分布,$\gamma\in(0,1)$用来弱化未来得到的奖赏.

Agent 循环执行以下两个步骤和环境进行交互:(1) 在系统状态为 $s\in S$ 时,Agent 执行动作 $a\in A$,作为该动作的效果得到了奖赏 $R(s,a)$;(2) 系统按照转移分布 $T(s,a,s')$转向了新的状态 s',Agent 按照观察分布 $O(a,s',z)$获取一个观察 $z\in Z$,它为 Agent 提供了隐蔽状态 s'的信息.

POMDP 规划的目标是找一个选择动作的策略π,它最大化折扣奖赏的期望值:$V^{\pi}(b_0)=E\left[\sum_{t=0}^{\infty}\gamma^t r_t\right]$.其中,$r_t$是时间 t 时的奖赏值,E 表示数学上的期望值算子,折扣因子 γ 确保了该式的收敛性.

1.2 值迭代

在 POMDP 中,状态不能被直接观察,因此为了执行最优的动作,Agent 必须维护一个所经历动作和观察的完整序列,即历史.历史可以用信念状态来取代,信念状态 b 是状态集 S 上的一个概率分布,所有信念状态组成一个$|S|$维的单形体 Δ.

任意时间 t 时的信念状态 b'都可以由时间 t–1 时的信念状态 b 递归计算得出.当 Agent 执行动作 a 并得到观察 z 时,信念状态将根据 Bayes 法则由 b 更新至 b':

$$b'(s')=b_a^z(s')=\Pr(s'\mid b,a,z)=\frac{O(a,s',z)\sum_{s\in S}b(s)T(s,a,s')}{\Pr(z\mid a,b)} \tag{1}$$

其中分母是一个规范化常量,它是所有 $s'\in S$ 上的分子之和.

以信念状态代替状态,POMDP 问题就被转化为连续状态的 MDP 问题,策略π相应转化为由信念单形体到动作的映射:$\pi(b)\to a$.在最优策略π^*下,所有信念状态的折扣奖赏期望值组成最优值函数,记作 V^*.因为这里时间是被假定为无穷的,所以 V^*具有如下性质:

$$V^*(b)=\max_{a\in A}\left[R(b,a)+\gamma\sum_{z\in Z}\Pr(z\mid a,b)V^*(b_a^z)\right] \tag{2}$$

其中 $R(b,a)=\sum_{s\in S}b(s)R(s,a)$.可以看出,最优策略可以根据最优值函数贪婪选取,因此我们的目标转化为计算 V^*.不同于 MDP,V^*不可以被直接求得,而是由 $V_0(b)=\max_{a\in A}R(b,a)$ 开始,根据式(2)递归构造.值迭代法就是用这样的值函数序列$\{V_t\}$来逼近 V^*,每次迭代称为 Backup 操作 H.

有这样一个重要事实[1,2]:值函数 V_t 是分段线形凸函数.因此,V_t 可以用一个信念单形体上超平面集Γ_t 来表示,我们称这些超平面的系数为α-向量.值函数可以表示为:$V_t(b)=\max_{\alpha\in\Gamma_t}b\bullet\alpha$,其中(•)表示内积.用α-向量表示公式(2)中的 V_t并最终整理得:

$$V_{t+1}(b)=\max_{a\in A}\sum_{z\in Z}\max_{\alpha\in\Gamma_t}\sum_{s\in S}b(s)\left[\frac{R(s,a)}{|Z|}+\gamma\sum_{s'\in S}T(s,a,s')O(a,s',z)\alpha(s')\right] \tag{3}$$

根据式(3),可以用更新α-向量来实现值函数的更新,通过执行以下操作由Γ_t生成Γ_{t+1}:

$$\Gamma_{t+1}^{a,z}\leftarrow\alpha_a^z(s)=\frac{R(s,a)}{|Z|}+\gamma\sum_{s'\in S}T(s,a,s')O(a,s',z)\alpha(s'),\forall\alpha\in\Gamma_t \tag{4}$$

$$\Gamma_{t+1}^{a}=\oplus_{z\in Z}\Gamma_{t+1}^{a,z};\Gamma_{t+1}=\bigcup_{a\in A}\Gamma_{t+1}^{a}$$

其中,(⊕)表示叉和算子:$A\oplus B=\{x|x=a+b,a\in A,b\in B\}$.从式(4)可以看出,最坏的情况下,每步会产生$|\Gamma_{t+1}|=O(|A||\Gamma_t|^{|Z|})$个$\alpha$-向量,时间复杂度为 $O(|S|^2|A||\Gamma_t|^{|Z|})$.而且裁剪向量需要做大量的线性规划,这些一直以来都是将 POMDP 应用于实际问题的主要障碍.

1.3 基于点的算法

为了解决维度和历史这两个问题[2],人们提出了各种近似算法.基于点的算法在计算上有明显优势,它在预取样信念点集 B 上进行值更新操作而不是在整个信念空间上.首先,值函数被初始化为一个α-向量$\alpha_0(s)=R_{\min}/(1-\gamma)$,在信念点 b 上的 Backup 操作是式(4)加上式(5):

$$\begin{aligned}\Gamma_{t+1}^{b}&\leftarrow\alpha_a^b=\sum_{z\in Z}\arg\max_{\alpha\in\Gamma_{t+1}^{a,z}}b\bullet\alpha,\forall a\in A\\ \Gamma_{t+1}&\leftarrow\alpha^b=\arg\max_{\alpha\in\Gamma_{t+1}^{b}}b\bullet\alpha,\forall b\in B\end{aligned} \tag{5}$$

这里介绍一下本文要用到的一个基于点的算法 Perseus. Perseus 算法首先随机取样一个足够大的信念点集

B,每次值迭代都确保新的值函数在 B 上是旧的上界.它在函数值仍低于旧值的点中随机选取点进行 Backup 操作,当所有点的值都被改良时,结束一次值迭代.算法 1 给出了 Perseus 算法值更新的详细步骤.

算法 1. Perseus 算法中的值更新.

Step 1:设 $\Gamma_{t+1}=\varnothing, B'=B$;

Step 2:从 B'中随机的选取一个 b;

Step 3:在 b 上执行 Backup 操作并得到α;

Step 4:如果 $b\bullet\alpha<V_t(b)$,则将α添加至 Γ_{t+1},否则将 $\alpha'=\arg\max_{\alpha\in\Gamma_t} b\bullet\alpha$ 添加至 Γ_{t+1};

Step 5:更新 V_{t+1},并计算 $B'=\{b\in B|V_{t+1}(b)<V_t(b)\}$;

Step 6:如果 $B'=\varnothing$,停止,否则回至 Step 2.

2 PPBA:基于点的算法的预处理方法

2.1 思 想

基于点的算法是最近研究比较热、应用比较广的一类近似算法.从 PBVI 和 Perseus 算法中可以看出,它们直接在信念点上进行更新操作,从而避免了线形规划并且用更少的超平面来表示值函数.

可以理解:每次更新操作都要按式(4)生成$|\Gamma_t||A||Z|$个中间向量,再由式(5)针对每个点生成其更新向量,既耗时又耗空间,我们认为:基于点的优势还没有被充分利用.回顾式(3),它采用了先生成再选择向量的思想,因为生成向量的过程与信念点无关,这样的"预处理"避免了为每个点选择向量时的重复计算.

但这样的"预处理"并不是一劳永逸的,它与当前的α-向量相关,每次更新操作时都需要重新计算.而基于点的算法中更新操作的步数是很大的,因为它用多次值更新来代替一次精确值更新,这里再次出现了重复计算的问题.从而我们就需要对不随时间变化并且有现实意义的量进行预处理,预取样点集中的每个点一旦被选取就不再改变,本文主要就是对信念点进行预处理.

此外,阻碍 POMDP 技术实际应用的是高维度问题.而高维度往往伴随着高稀疏性,目前许多研究[14]就是针对稀疏性的,本文提出的 PPBA 算法也考虑了稀疏性,从而避免了无意义计算并降低了存储.

2.2 取样点的预处理

我们从最核心和最耗时间的 Backup 操作入手,将式(3)重新写作式(6):

$$V_{t+1}(b)=\max_{a\in A}\left[\sum_{s\in S}R(s,a)b(s)+\gamma\sum_{z\in Z}\max_{\alpha\in\Gamma_t}\sum_{s'\in S}\alpha(s')\sum_{s\in S}T(s,a,s')O(a,s',z)b(s)\right] \tag{6}$$

由上式可知值迭代过程中只有$\alpha\in\Gamma_t$是变化的,所以可以事先计算并存储其他部分.

第 1 部分可以存储在 $R(b,a)$中,它是信念状态的期望奖赏值.

我们为第 2 部分也就是最耗时的部分定义β-向量 $\beta_b^{a,z}$,其中 $\beta_b^{a,z}(s')=\sum_{s\in S}T(s,a,s')O(a,s',z)b(s)$.

因此,式(6)可以被重写为:

$$V_{t+1}(b)=\max_{a\in A}\left[R(b,a)+\gamma\sum_{z\in Z}\max_{\alpha\in\Gamma_t}\beta_b^{a,z}\bullet\alpha\right] \tag{7}$$

基于上面的分析,算法的大体思路是(详细参见算法 2):针对预处理好的 $R(b,a)$和 $\beta_b^{a,z}$,在每次对点 b 的更新操作中,先根据式(7)很快地计算出最优动作和每个观察下应选的 α-向量,然后再根据式(4)生成新的 α-向量.

$R(b,a)$和 $\beta_b^{a,z}$ 并不是一般的常量数据,它们有着自身的现实意义.将式(7)和式(2)对比,它正是最初的值函数模型:当前的直接奖赏值加上未来的折扣奖赏值.$R(b,a)$显然表示在当前信念状态 b 下执行动作 a 的直接奖赏值. $\beta_b^{a,z}$ 其实就是 $\Pr(z\mid a,b)\bullet b_a^z$,其中,$b_a^z$ 是在当前信念状态 b 下执行动作 a 并得到观察 z 后的新信念点.$\Pr(z|a,b)$

仅仅是 $\beta_b^{a,z}$ 的规范化因子,即 $\Pr(z\mid a,b)=\sum_{s'\in S}\beta_b^{a,z}(s')$,从而我们可以将 $\beta_b^{a,z}$ “等同”看为 b_a^z. $\beta_b^{a,z}$ 集成了取样信念点所有的转化信息,也是我们预处理的重点.

对 $\beta_b^{a,z}$ 的预处理是一劳永逸的,它避免了重复和无意义计算,对算法性能的评价见第 2.5 节.

2.3 基向量

选择α-向量的过程与传统算法类似,在最终生成α-向量时,我们同样利用稀疏性进行预处理,并引进基向量的概念.

因为 $\beta_b^{a,z}$ 在大部分观察 z 下为 0 向量,所以式(7)的第 2 个 max 算子往往不需要计算,它一定为 0.问题是,在这种情况下究竟该选择Γ_t 中的哪个α-向量呢?实际上,选择任何一个都是正确的,但为了能进行预处理,我们假定这种情况下都默认选择同一个α-向量,本文的实验中都是选择的第 1 个向量 $\bar{\alpha}$.

如果 $\beta_b^{a,z}$ 在每个观察下都为 0 向量,那么生成新α-向量α_b 的过程将变得更简单:对最优动作 $\hat{a}$,将式(4)中α 的换成 $\bar{\alpha}$,并在所有观察上 z 求和,即 $\alpha_b(s)=R(s,\hat{a})+\gamma\sum_{z\in Z}\sum_{s'\in S}T(s,\hat{a},s')O(\hat{a},s',z)\bar{\alpha}(s')$.

我们定义上式的求和算子部分为动作 $\hat{a}$ 下的基向量,同样可以为每个动作 a 定义一个基向量α_a,其中 $\alpha_a(s)=\sum_{z\in Z}\sum_{s'\in S}T(s,a,s')O(a,s',z)\bar{\alpha}(s')$.基向量与信念点无关,因此,可以在每次值迭代之前对其进行预处理.真正的新α-向量与α_b 的区别仅仅在于:对个别观察 $z,\bar{\alpha}$ 被替换成了实际选择的向量.所以在为某个信念点生成α-向量时,只需用实际选择向量来修正对应基向量就行了.设某个观察 z 下 $\beta_b^{a,z}$ 非零,且选择了最优向量α,修正的方法是在基向量 $\alpha_{\hat{a}}$ 上加上误差项 $TO(\alpha-\bar{\alpha})$ (算法 2 的 Step 4).

在 PPBA 算法中,生成向量成为了次要部分,与传统算法相比,资源消耗微不足道.但随着值更新的重点由生成向量转向选择向量,这部分消耗也是不容忽视的.另外,基向量的思想也为稀疏问题生成向量提供了一个可选择的方法.

2.4 算法描述

预处理方法的基本思路已在前面给出,完整的方法分为 3 部分:(1) 在整个算法之前计算并存储 $R(b,a)$和 $\beta_b^{a,z}$;(2) 在每次值迭代之前计算并存储基向量;(3) 在单个信念点上利用 $R(b,a)$和 $\beta_b^{a,z}$ 选择向量,利用基向量生成α-向量,算法 2 中详细描述了单个信念点上的向量更新.

算法 2. PPBA 中在点 b 上的 Backup 操作.

Step 1:如果 $\beta_b^{a,z}\neq 0$,计算 $R_a^z=\max_{\alpha\in\Gamma_t}\beta_b^{a,z}\bullet\alpha$ 并记录 $\alpha_a^z=\arg\max_{\alpha\in\Gamma_t}\beta_b^{a,z}\bullet\alpha$;

Step 2:计算 $R_a=R(b,a)+\gamma\sum_z R_a^z$;

Step 3:选择 $\hat{\alpha}=\arg\max_{a\in A}R_a$;

Step 4:设 $\alpha_b=\alpha_{\hat{a}}$;如果 $\beta_b^{\hat{a},z}\neq 0$, $\alpha_b(s)\leftarrow\alpha_b(s)+\sum_{s'\in S}T(s,\hat{a},s')O(\hat{a},s',z)\left[\alpha_{\hat{a}}^z(s')-\bar{\alpha}(s')\right]$;

Step 5: $\alpha_b(s)\leftarrow R(s,\hat{a})+\gamma\alpha_b(s)$.

前 3 步是选择最优动作和向量,后 2 步是生成新的α-向量.Step 1 检查所有的动作和观察,对非零的β-向量,计算最优未来奖赏值并记录相应的向量;Step 2 计算所有动作下的折扣奖赏值;Step 3 根据奖赏值选择最优动作;Step 4 初始化新α-向量为最优动作下的基向量,并用所有误差项对其进行修正;Step 5 根据式(4)最终生成新α-向量.

2.5 算法分析

PPBA 避免了大量的重复和无意义计算,但其额外的存储和处理将影响算法的效率.以下从空间复杂度和时间复杂度两方面比较新旧方法.

存储问题上,PPBA 算法和传统算法都需要保存大量的临时向量.PPBA 中 $R(b,a)$和基向量的大小微不足道,消耗存储空间的主要是 $\beta_b^{a,z}$,理论上要存储$|B||A||Z|$个长度为$|S|$的β-向量.由于稀疏性,在某一特定时间大部分观

察是得不到的,即 Pr($z|a,b$)通常为 0,从而绝大部分 $\beta_b^{a,z}$ 为 0 向量,设稀疏因子为λ_1.因为β-向量可以“等同”于信念点,它同样具有稀疏性,设稀疏因子为λ_2.这样,PPBA 算法需要的额外空间是$\lambda_1\lambda_2|B||S||A||Z|$.传统算法的每次更新操作前都要保存$|\Gamma_i||A||Z|$个长度为$|S|$的$\alpha$-向量,并且任一项都不具有稀疏性,因此共需额外空间$|\Gamma_i||S||A||Z|$.容易看出,新旧方法的空间复杂度差异取决于样本点与α-向量的数量关系,而任何算法都会将两者的比例控制在 100:1 以内.对于样本点数目受控制的算法,比如 PBVI,每个α-向量仅对应少量样本点,PPBA 算法的空间复杂度显然低于传统算法;即使是 Perseus 这样需要大量样本点的算法,由于稀疏性,两者的空间复杂度也相当.

处理效率上,PPBA 方法多出了预处理这一步,但它节省了每次值更新生成α-向量的时间.我们分两部分来比较新旧方法的时间复杂度:α-向量的选择和生成,我们把预处理所消耗时间都算到生成向量中.选择向量时新旧方法过程类似,分别参考式(5)和算法 2 的前 3 步.以单个信念点计算,传统算法共进行内积运算$|\Gamma_i||A||Z|+|A|$次,PPBA 方法因只选择非零的β-向量,仅有$\lambda_1|\Gamma_i||A||Z|$次内积运算,其中$\lambda_1$ 为 $\beta_b^{a,z}$ 的稀疏因子,预处理方法显然效率更高.生成向量部分我们计算每次完整值迭代所需乘法次数,设$|\Gamma|$表示平均α-向量数.参考式(4),传统算法中乘法次数的级别为$2|\Gamma||S|^2|A||Z|$.PPBA 算法中预处理基向量和生成向量(算法 2 后 2 步)耗时很少,最耗时的主要是预处理β-向量,平均到每次值迭代乘法次数的级别为$|B||S|^2|A||Z|/T$,其中 T 为总迭代次数.前面说过,基于点的算法以多次值迭代代替一次精确值迭代,迭代次数 T 较大(往往比$|B|/|\Gamma|$大很多).因此,$|B|/T$ 会比 $2|\Gamma|$小很多,尤其是对哪些样本点数目受控制的算法.总之,预处理所需的时间平均到每步中远小于所节省的时间,PPBA 算法在时间上有绝对优势.

但 PPBA 作为一个预处理方法有它固有的缺点:(1) 方法仅适用于基于点的算法.因为它依赖于对预取样点的预处理;(2) 对于有大量取样点的算法,预处理的消耗也是可观的.预处理的时间和空间消耗随取样点集的增大而增大,这从表 1 中可见一斑,Tag 问题比其他 3 个问题用了更多的预处理时间;(3) 仅仅加快了算法速度,对最优策略的改善甚微.PPBA 仅仅是个加速算法,从表 1 可以看出,不论是获得的策略的性能(reward),还是简易程度(向量个数)都与传统算法相当.

Table 1 Summary of results

表 1 结果总结

	Method	Reward	Vectors	Time(s)
Tiger-grid	Perseus	2.250 7	69	66
($\|S\|$=36, $\|A\|$=5, $\|Z\|$=17)	PPBA	2.224 5	39	28+3
Hallway	Perseus	0.559 2	50	30
($\|S\|$=60, $\|A\|$=5, $\|Z\|$=21)	PPBA	0.560 8	53	9+3
Hallway2	Perseus	0.355 9	80	79
($\|S\|$=92, $\|A\|$=5, $\|Z\|$=17)	PPBA	0.350 5	56	12+3
Tag	Perseus	−6.388 3	224	784
($\|S\|$=870, $\|A\|$=5, $\|Z\|$=30)	PPBA	−6.390 4	238	263+95

3 实 验

PPBA 算法理论上对大部分基于点的算法都是有效的,因为 Perseus 算法中的信念点集在值迭代过程中不会扩充,这使得实现预处理相对容易一些,所以作为验证,下面将 PPBA 方法应用于 Perseus 算法,并针对来自文献[6,10]的 4 个 POMDP 问题(Tiger-grid、Hallway、Hallway2 和 Tag)进行实验.

Tiger、Hallway 和 Hallway2 问题[10]都是迷宫问题,它们维度较高,稀疏性低,被普遍应用于检验可扩展的 POMDP 技术.Tag 问题[6]模拟了两个机器人之间的搜寻和标记游戏,它在量级上比其他 3 个问题更大,且具有高稀疏性,它以更苛刻条件来检验新的 POMDP 技术.表 1 中有这 4 个问题的参数.

我们在每个问题的新旧方法上都进行 10 次实验,每次实验都会重新设置不同的随机种子,最终对每个方法上的 10 次实验取平均值.为观察所生成策略随时间的变化情况,实验记录并测试每次迭代得到的策略.测试每个策略时重新设置随机种子,我们用随机探索的方法来近似计算它的折扣奖赏期望值.分别从随机获取的 100 个初始状态开始探索并获得折扣奖赏,每个初始状态重复执行 10 次,最后对这 1 000 次探索的折扣奖赏取平均

值,就是该策略的近似期望奖赏.

我们设折扣因子为γ=0.95,每次探索在到达目标或超过预设的最大步数时停止.迷宫问题中,我们都取样有 1 000 个信念点的点集B,并设 Tiger-grid 的最大探索步数为 500,达到目标后重设状态并继续探索,而其它问题则开始新一轮的探索; Hallway 和 Hallway2 的最大探索步数为 251;Tag 问题中,我们取样有 10 000 个信念点的点集 B,并设最大探索步数为 100.这里所有的参数都遵循文献[7].

我们用以下 2 种度量值[7]来评价算法:(1) 策论的改变数目:相比较 V_t,在 V_{t+1} 中选择了不同的最优动作的点 $b\in B$ 的个数,它可以用来指示算法收敛性,如果策略不再有较大变化,我们就可以认为它已经收敛了.(2) 折扣奖赏的期望值,可以用它来检验所生成策略的优劣.图 1、图 2 是这两个度量值随时间的变化,其中虚线表示 Perseus 算法的执行情况而实线表示 PPBA 算法.因为值迭代的时间大致成指数增长,这里的横坐标轴也采用指数坐标形式,所以两条线之间很小的间隔往往表示较大的差距.

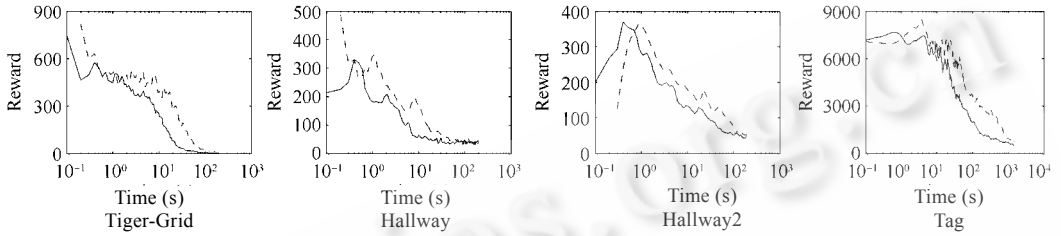

Fig.1 Change of the Policy

图 1 策略的变化

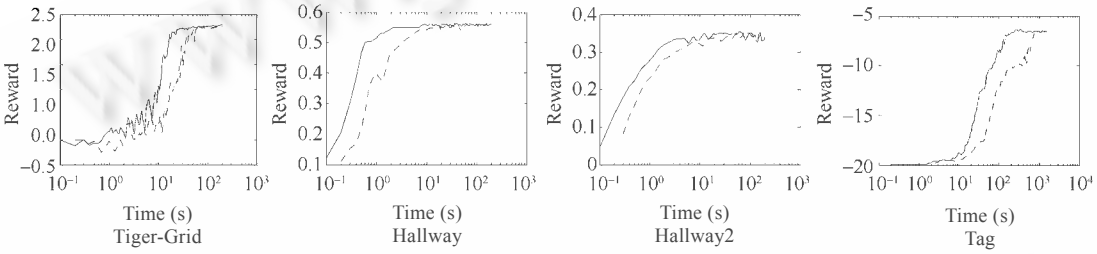

Fig.2 Expected discounted reward

图 2 期望折扣奖赏

从两组图中可以看出,PPBA 比 Perseus 更早地得到了收敛的策略和更快地获得了最佳奖赏.当奖赏值开始在最佳奖赏值附件波动时,我们认为策略收敛.表 1 中总结了策略收敛时的重要数据,前两项是最优策略的折扣奖赏值和 α-向量个数,最后一项是策略收敛的时间,其中加号后面的是预处理消耗的时间.

Perseus 算法要求预取样大量的信念点,这对预处理方法是个挑战.但从实验结果可以看出,不管是较小的低稀疏的迷宫问题,还是较大的高稀疏的 Tag 问题,PPBA 算法都大大地提高了执行速度.为达到最佳奖赏,PPBA 算法只用了 Perseus 算法 1/5 到 1/2 的时间(包含预处理时间).但同时,该方法对提高折扣奖赏帮助甚微,最优策略的复杂度(向量个数)也没有得到降低,这些固有的缺点是由预处理方法的本质所决定的.

4 结束语

本文提出了一个基于点的 POMDP 算法的预处理方法.传统的基于点的算法往往是首先生成所有可能的α-向量,然后再决定执行那个动作和选择哪些α-向量.有两个因素制约了算法的执行速度:(1) 绝大部分α-向量最终都不会被选择;(2) 在每个信念点上的计算将被重复.而在 PPBA 算法中,首先对所有选择的信念点做预处理,这样我们就可以先决定执行那个动作和选择哪些α-向量,然后再生成这些少数的α-向量.PPBA 算法还利用到

了问题的稀疏性这个特点,避免了大量的无意义计算,所以它对稀疏问题更有效.实验结果表明,本方法很大程度地提高了执行速度.

在未来的工作中,我们还要在其他基于点的算法中继续检验我们的方法,并考虑在基于策略的算法中运用基于点的思想.

致谢 感谢 Spaan MTJ 的基于 Matlab 语言的 POMDP 软件,在本文的实验中,我们复用了其中的很多代码.

References:

[1] Sondik EJ. The Optimal control of partially observable Markov processes over the infinite horizon: Discounted costs. Operations Research, 1978,26(2):282−304.

[2] Kaelbling LP, Littman ML, Cassandra AR. Planning and acting in partially observable stochastic domains. Artificial Intelligence, 1998,101:99−134.

[3] Cheng HT. Algorithms for partially observable Markov decision processes [Ph.D. Thesis]. Vancouver Columbia: University of British Columbia, 1988.

[4] Zhang NL, Zhang W. Speeding up the convergence of value iteration in partially observable Markov decision processes. Journal of Artificial Intelligence Research, 2001,14:29−51.

[5] Cassandra A, Littman M, Zhang N. Incremental pruning: A simple, fast, exact method for partially observable Markov decision processes. In: Dan G, Prakash PS, ed. Proc. of the 13th Annual Conf. on Uncertainty in Artificial Intelligence. San Francisco: Morgan Kaufmann Publishers, 1997. 54−61.

[6] Pineau J, Gordon G, Thrun S. Point-Based value iteration: An anytime algorithm for POMDP. In: Proc. of the 18th Int'l Joint Conf. on Artificial Intelligence (IJCAI). 2003. 1025−1030.

[7] Vlassis N, Spaan MTJ. A fast point-based algorithm for POMDP. In: Proc. of Annual Machine Learning Conf. of Belgium and the Netherlands, 2004. 170−176.

[8] Spaan MTJ, Vlassis N. Perseus: Randomized point-base value iteration for POMDP. Journal of Artificial Intelligence Research, 2005,24:195−220.

[9] Pineau J, Gordon G, Thrun S. Point-Based approximations for fast POMDP solving. Technical Report, SOCS-TR-2005.4, School of Computer Science, McGill University, 2005. 1−45

[10] Littman ML, Cassandra AR, Kaelbling LP. Learning policies for partially observable environments: Scaling up. Armand P, Stuart R, ed. Proc. of the 12th Int'l Conf. on Machine Learning. San Francisco: Morgan Kaufmann publishers Inc., 1995. 362−370.

[11] Zhang B, Cai QS, Guo BN. POMDP model and its solution for spoken dialogue system. Journal of Computer Research and Development, 2002,39(2):217−224 (in Chinese with English abstract).

[12] Zhou JE, Liu GQ, Zhang CY, Cai QS. User modeling based on inner-belief state POMDP. Mini-Micro Systems, 2004,25(11): 1979−1983 (in Chinese with English abstract).

[13] Chen M, Chen XP. Sampling based approximate algorithm for POMDP. Computer Simulation, 2006,23(5):64−67 (in Chinese with English abstract).

附中文参考文献:

[11] 张波,蔡庆生.口语对话系统的 POMDP 模型及求解.计算机研究与发展,2002,39(2):217−224.

[12] 周继恩,刘贵全.基于内部信念状态 POMDP 模型在用户兴趣获取中的应用.小型微型计算机系统,2004,25(11):1979−1983.

[13] 陈茂,陈小平.基于采样的 POMDP 近似算法.计算机仿真,2006,23(5):64−67.

卞爱华(1983－),男,江苏大丰人,硕士生,主要研究领域为多 Agent 系统,POMDP.

陈世福(1937－),男,教授,博士生导师,主要研究领域为人工智能,知识工程,智能系统应用.

王崇骏(1975－),男,博士,副教授,主要研究领域为多 Agent 系统,智能化信息处理,模式识别.