ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.19, No.11, November 2008, pp.2869−2878
DOI: 10.3724/SP.J.1001.2008.02869

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

实时动态规划的最优行动判据及算法改进*

范长杰, 陈小平+

(中国科学技术大学 计算机科学与技术系,安徽 合肥 230027)

Optimal Action Criterion and Algorithm Improvement of Real-Time Dynamic Programming

FAN Chang-Jie, CHEN Xiao-Ping+

(Department of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China)

+ Corresponding author: E-mail: xpchen@ustc.edu.cn

Fan CJ, Chen XP. Optimal action criterion and algorithm improvement of real-time dynamic programming. *Journal of Software*, 2008,19(11):2869−2878. http://www.jos.org.cn/1000-9825/19/2869.htm

Abstract: This paper is primarily to improve the efficiency of real-time dynamic programming (RTDP) algorithm for solving Markov decision problems. Several typical convergence criteria are compared and analyzed. A criterion called optimal action criterion and a corresponding branch strategy are proposed on the basis of the upper and lower bound theory. This criterion guarantees that the agent can act earlier in a real-time decision process while an optimal policy with sufficient precision still remains. It can be proved that under certain conditions one can obtain an optimal policy with arbitrary precision by using such an incremental method. With these new techniques, a bounded incremental real-time dynamic programming (BI-RTDP) algorithm is designed. In the experiments of two typical real-time simulation systems, BI-RTDP outperforms the other state-of-the-art RTDP algorithms tested.

Key words: MDP (Markov decision process); RTDP (real-time dynamic programming); convergence criterion; incremental solving; heuristic search

摘 要: 主要以提高求解马尔可夫决策问题的实时动态规划(real-time dynamic programming,简称 RTDP)算法的效率为目的.对几类典型的实时动态规划算法所使用的收敛判据进行了对比分析,并利用值函数上界、下界给出了称为最优行动判据的收敛判据,以及一个更适合实时算法的分支选择策略.最优行动判据可以更早地标定当前状态满足精度要求的最优行动供立即执行,而新的分支选择策略可以加快这一判据的满足.据此设计了一种有界增量实时动态规划(bounded incremental RTDP,简称 BI-RTDP)算法.在两种典型仿真实时环境的实验中,BI-RTDP 均显示出优于现有相关算法的实时性能.

关键词: 马尔可夫决策过程;实时动态规划;收敛判据;增量求解;启发式搜索

中图法分类号: TP301 **文献标识码**: A

马尔可夫决策模型适用于行动结果具有不确定性的多步规划问题[1],它有着广泛的现实应用背景.在人工智能领域已发展出大量针对马尔可夫决策问题的求解算法.马尔可夫模型将客观世界的动态特性用状态转移

* Supported by the National Natural Science Foundation of China under Grant No.60745002 (国家自然科学基金); the National Basic Research Program of China under No.2003CB317002 (国家重点基础研究发展计划(973))

Received 2007-10-10; Accepted 2008-02-04

来描述,相关算法可以按是否求解全部状态空间进行划分.早期求解算法有值迭代和策略迭代,这些方法采用动态规划,以一种后向的方式同时求解出所有状态的最优策略.随后,一些利用状态可达性的前向搜索算法,如AO*,LAO*[2]相继被提出来,其特点是只求解从给定初始状态开始的最优策略,通常可以避免大量不必要的计算,获得更高的效率.与 AO*算法相比,LAO*能够处理状态转移存在环的系统.同样利用状态可达性并结合动态规划的算法有:Heuristic Search/DP(HDP)[3],Envelope Propagation(EP)[4]以及 Focused Dynamic Programming (FP)[5].

从另一个角度来说,相关算法还可以按照离线或在线来划分.对于很多现实世界应用中的大规模问题,无论是否利用状态可达性,解都不可能以离线的方式一次性求出,这种情况更适合使用在线算法,也称为实时算法.实时算法的决策计算与执行交替进行,且解的质量通常随给定计算时间的增加而提升.最早的基于动态规划的实时算法是 RTDP[6].RTDP 通过不断循环 Trail 来改进策略,每次 Trail 确定一个从初始状态到目标状态的路径,然后进行反向的值迭代,然而 RTDP 不处理停止问题(stopping problem)[7].停止问题是指如何判断当前解的质量是否已满足要求进而停止计算并提交策略供执行.在值迭代类算法中,停止问题对应于收敛判据.Labeled RTDP(LRTDP)[8]通过标记各经历状态是否已被求解,给出了一种处理停止问题的方式,同时避免已经求解过的状态处的计算进而加快收敛.最新的实时动态规划(real-time dynamic programming,简称 RTDP)算法,如BRTDP[9]及 FRTDP[10]使用了另外一种技术,求解过程记录并不断更新相关状态期望值函数的上界、下界.这些信息用来指导分支选择,显著地提高了算法性能.另一方面,上、下界提供了最优值函数的一个区间估计,当给定初始状态的值函数上、下界间隔足够小时,便可认为已经获得满足精度的最优策略.

事实上,实时系统的多步决策具有下面的特性,决策与执行交替进行,整个在线计算被分割在各步执行之间,而当前第 1 步的行动是要被马上执行的,且执行后仍有时间计算后续策略.实时系统本身可以分为软实时与硬实时两类[11].软实时系统对于每步决策没有严格的时间限制,可以在任何时刻选择决策或者执行,只要求总的规划时间尽可能短.而硬实时系统每步决策有严格的时间限制,通常要求在每个给定时间片内必须提交一个行动供执行.本文结合上、下界给出了一个最优行动判据,在实时环境中,相关算法利用该判据可以更早地提交一个行动供执行,而保证满足精度要求的最优策略仍然存在.同时,我们证明,当满足额外几个适当的条件时,可以在求解过程中利用最优行动判据来标定状态是否已被求解.这不同于 LRTDP 的标定方式,后面的章节将详细加以区别.另一方面,实时动态规划本质上是一个异步值迭代过程,值迭代的顺序决定了算法的收敛速度.RTDP 类算法使用了类似前向搜索的分支选择来确定值迭代顺序.本文给出一种利用最优行动判据对分支选择加以改进的方法,其对应值迭代顺序更倾向于集中计算来确定当前步的最优行动.结合上述几点改进,我们设计了一种有界增量实时动态规划算法 BI-RTDP(bounded incremental RTDP).在 RaceTrack 标准问题上模拟两种实时系统所进行的实验都显示出,较之现有相关算法,BI-RTDP 具有更好的实时性能.

第 1 节介绍一些 MDP 相关基本概念及 RTDP 算法的基本框架.第 2 节首先分析现有几类处理停止问题的RTDP 算法的收敛判据,然后给出新的最优行动判据.第 3 节分析最新实时动态规划算法与相关启发式搜索类算法之间分支控制方面的联系,进而给出一个更好的配合最优行动判据的分支选择策略.第 4 节为 BI-RTDP 算法增量实时求解的实现.第 5 节为实验及结论.

1 背景及相关工作

1.1 马尔可夫模型

马尔可夫决策过程(Markov decision process,简称 MDP)[12]的模型为一个四元组$\langle S,A,T,R\rangle$.其中,S 为状态集合;A 为行动集合;$T: S\times A\times S\to[0,1]$ 为转移函数,对于每个状态及选择的行动,由 $T^a(s,s')$指定了转移到下一个状态的概率分布 $P(s'\mid s,a)$;$R: S\times A\to\mathbb{R}$ 为立即收益的回报函数(reward function),它指定了在某个状态要选择哪个行动.定义值函数(value function)$V_\pi: S\to\mathbb{R}$ 为采用策略π时在状态 s 的期望回报:

$$V_\pi(s_0)=E\left[\sum_{t=0}^{\infty}\gamma^t R(s_t,\pi(s_t))\right] \tag{1}$$

其中,s_t为时刻 t 所处状态.类似地,定义行动值函数 $Q_\pi : S\times A\to\mathbb{R}$ 为在状态 s 采用行动 a,其他状态采用策略π的期望回报.最优策略记为π^*,对应值函数为 V^*,称为最优值函数.

通常,当一个策略π满足对状态 s,有 $V^*(s)-V_\pi(s)\le\varepsilon$ 时,我们称π为状态 s 处的ε最优策略,当π对问题所有状态均满足上述条件时,称其为问题的ε最优策略.

1.2 策略迭代与值迭代

策略迭代与值迭代是求解 MDP 问题的两种最基本的方法.在策略迭代中,策略显式表示,可以计算得到对应 V_π,然后使用以下的公式来改进策略:

$$Q_\pi(s,a)=R(s,a)+\gamma\sum_{s'\in S}T^a(s,s')V_\pi(s') \tag{2}$$

$$\pi'(s)=\arg\max_{a\in A}Q_\pi(s,a) \tag{3}$$

其中,γ为折扣因子(discount factor), $\gamma\le 1$.重复这个过程,直到策略固定为止.

在值迭代过程中,策略没有显式表示,整个过程按照 Bellman 公式对值函数进行迭代计算:

$$V(s)=\max\left\{R(s,a)+\gamma\sum_{s'\in S}T^a(s,s')V(s')\right\} \tag{4}$$

每次迭代,所有状态值函数更新前后的最大差值称为 Bellman 误差 r,$r=\max_{s\in S}\left|V(s)-V'(s)\right|$.

定义一致性条件(monotonic condition)为:对所有 s,有 $V(s)\le\max_a\left[R(s,a)+\sum_{s'\in S}T^a(s,s')V(s')\right]$[8].

策略可以由值函数 V 得到,定义一步前瞻的贪婪策略 π_G 为:对所有 s,有

$$\pi_G(s)=\arg\max_{a\in A}\left\{R(s,a)+\gamma\sum_{s'\in S}T^a(s,s')V(s')\right\} \tag{5}$$

如果下界满足一致性,则 π_G 是对当前值函数对应的隐式策略π_V 的改进,有 $V_{\pi_G}\ge V_{\pi_V}$. 即对$\forall s$,当 $V^*(s)-V(s)\le\varepsilon$ 时,π_G 亦为ε最优策略.

1.3 结合启发式搜索的RTDP

对于行动结果具有不确定性的问题的启发式搜索是在一个与或树(图)上进行的.或节点对应状态,此处进行行动选择;与节点对应行动,行动的结果可能并不唯一,需要考虑所有分支的后继状态.RTDP 是一个 Trials-based 方法,每次实验也可以看成在与或图上做的搜索,分支选择分为行动选择与后继状态选择.行动选择根据值函数上界提供的启发式信息采用一步前瞻的贪婪策略,后继状态选择模拟行动结果的不确定性随机选取.一次 Trial 从初始状态出发,到终止状态结束,然后进行反向值函数的更新,此过程不断循环.流程见算法 1.

算法 1. RTDP.

Function trialRecurse(s):
  if *isTerminal*(*s*) **then return end if**
  a←*chooseGreedyAction*(*s*)
  s′←*chooseSuccessorStochastically*(*s*,*a*)
  trialRecurse(*s′*)
  update(*s*)
Function *RTDP*(s_0):
  loop *trialRecurse*(s_0)

一些较新的 RTDP 类算法同时还记录下界的信息,结合上界可以提供对最优值函数的一个区间估计.定义 $V_U(s)$和 $V_L(s)$分别为状态 s 处最优值函数的上界和下界,下界为最优策略值函数的保守估计,上界为乐观估计.对于 $\forall s\in S$, 有: $V_L(s)\le V^*(s)\le V_U(s)$.在值迭代过程中,上界和下界的更新都按照 Bellman 公式独立进行.

2 收敛判据

2.1 现有RTDP类算法的收敛判据

收敛判据用以解决解的质量的评估问题.在同步值迭代算法中,每次迭代会将所有状态全部更新一遍,可以结合下面两点来判定解是否收敛:(1) 每次值迭代的 Bellman 误差;(2) 所有状态处策略执行到问题结束的期望步数,两者的乘积决定了当前值函数与最优值函数差值的上限[2].MDP 求解问题通常假设任何适当的策略的期望执行步数是有限的.在有限步 MDP 问题中,这一点显然满足.而无限步 MDP 问题的求解会设置小于 1 的折扣因子γ,事实上,这是人为地附加了一种条件,表示过程每步都可能以$(1-\gamma)$的概率结束,这同样满足策略不会无限步执行.

实时动态规划类算法的求解也是一个值迭代过程,每次由前向搜索确定一个状态序列,然后进行反向的异步值迭代[6].由于此过程中每次值迭代并不更新所有状态,因此,同步值迭代中使用的收敛判定方法在这里是不可行的.RTDP 是最早的实时动态规划算法,它没有处理此异步值迭代过程的收敛判据问题,也就是说,它不处理停止条件,因而,算法不适合离线运行.事实上,RTDP 只适合在硬实时环境中执行,依靠问题本身每步决策的时间限制来停止决策.

LRTDP 给出一种适用于异步值迭代的收敛判定方法.在反向值迭代的过程中,它会进行检测并标记哪些状态已经被求解.状态被标记的条件是其后继状态均满足 Bellman 误差小于某给定值.当问题给定初始状态被标记为已求解时,则认为已经获得满足精度要求的解,算法可以停止.LRTDP 能够离线运行,同时也可以在软实时及硬实时环境下执行.

BRTDP 和 FRTDP 使用了一种新的技术——结合上界、下界的区间估计,给出了设初始状态为 s 时解的一个非常简洁的收敛判据:

(I) $$V_U(s)-V_L(s)\le\varepsilon$$

设 $\pi_{L\mathcal{G}}$ 为下界得到的贪婪策略,当下界满足一致性时,有 $V_{\pi_{L\mathcal{G}}}(s)\ge V_L(s)$,可得: $V^*(s)\le V_U(s)\le\varepsilon+V_L(s)\le\varepsilon+V_{\pi_{L\mathcal{G}}}(s)$, 即 $V^*(s)-V_{\pi_{L\mathcal{G}}}(s)\le\varepsilon$. 也就是说,当判据(I)满足时,已经找到状态 s 处的ε最优策略.

2.2 最优行动判据

本文结合上、下界技术,给出了一个新的最优行动判据.首先定义行动值函数上、下界 $Q_U(s,a)$ 和 $Q_L(s,a)$,有

$$\begin{cases}Q_U(s,a)=R(s,a)+\gamma\sum_{s'\in S}T^a(s,s')V_U(s')\\ Q_L(s,a)=R(s,a)+\gamma\sum_{s'\in S}T^a(s,s')V_L(s')\end{cases}\tag{6}$$

其中,$V_U(s)=\max_{a\in A}Q_U(s,a), V_L(s)=\max_{a\in A}Q_L(s,a)$, 上界行动 $a_U(s)=\arg\max_{a\in A}Q_U(s,a)$, 下界行动 $a_L(s)=\arg\max_{a\in A}Q_L(s,a)$, 同时有 $V_{\pi_{L\mathcal{G}}}(s)=Q_L(s,a_L(s))$.

事实上,应该注意到对给定状态,当某个行动值函数下界不小于其他所有行动值函数的上界时,其他行动都可以被排除.按照这一思路,首先定义 $V_U^{-a_L}$ 和 $a_U^{-L}(s)$ 两个量, $V_U^{-a_L}$ 为状态 s 处除 a_L 以外其他行动对应的最大行动值函数上界, $a_U^{-L}(s)$ 为相应行动,有: $V_U^{-a_L}(s)=\max_{a\in A,a\ne a_L}Q_U(s,a), a_U^{-L}(s)=\arg\max_{a\in A,a\ne a_L}Q_U(s,a)$. 在考虑近似最优的情况下,有下述定理:

定理 1. 当 $V_U^{-a_L}(s)-V_L(s)\le\varepsilon$ 时,总是 $\exists\pi$, 且满足 $\pi(s)=a_L(s)$, 使得 $V^*(s)-V_\pi(s)\le\varepsilon$.

证明:分两种情况来看:如果 $\pi^*(s)=a_L(s)$, 那么显然结论成立;如果 $\pi^*(s)\ne a_L(s)$, 可知 $V^*(s)\le V_U^{-a_L}(s)$, 进而有 $V^*(s)\le V_L(s)+\varepsilon=Q_L(s,a_L(s))+\varepsilon$, 即 $V^*-Q_L(s,a_L(s))\le\varepsilon$. 因此,可以得到结论,当满足判据:

(II) $$V_U^{-a_L}(s)-V_L(s)\le\varepsilon$$

时,执行 a_L 后,ε最优解仍然存在,则称判据(II)为最优行动判据. □

推论 1. 当判据(I)中的条件成立时,判据(II)中的条件总是成立,反之则未必.也就是说,作为条件,判据(II)比判据(I)更弱.

证明:由于 $V_U^{-a_L} \le V_U$, 立刻有 $V_U - V_L \le \varepsilon \Rightarrow V_U^{-a_L} - V_L \le \varepsilon$, 即当判据(I)成立时,判据(II)总成立.对于其逆命题,图 1 给出反例并示意了判据(I)与判据(II)的区别,判据(I)在图 1(a)中满足,在图 1(b)中不满足,但判据(II)在两种情况下均满足. □

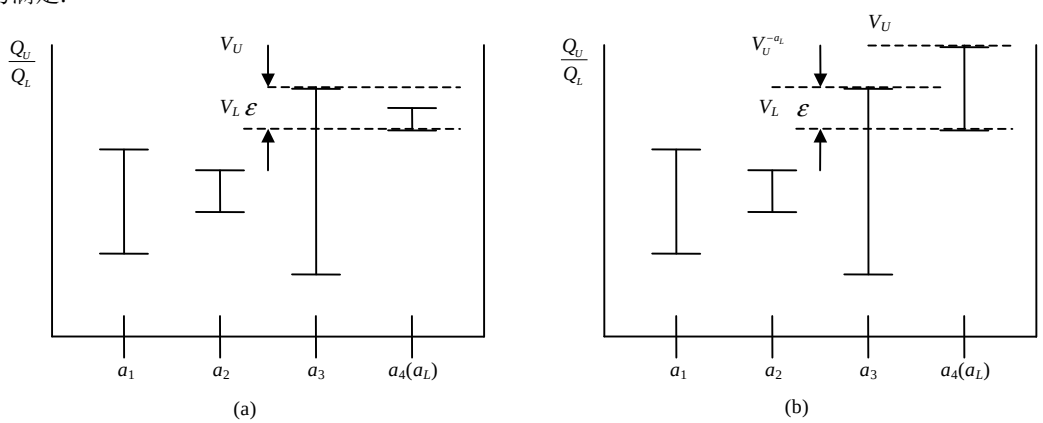

Fig.1 Difference of the criterion (I) and the criterion (II)

图 1 判据(I)和判据(II)的区别

我们考虑这样的问题:给定ε,从状态 s_0 开始,且当每步执行时都满足判据(II)且执行对应 a_L,整个过程获得的期望回报 $V^{a_L}(s_0)$ 与 $V^*(s_0)$ 一定满足 $V^*(s_0) - V^{a_L}(s_0) \le \varepsilon$ 吗?每步的这种误差会累加吗?考虑这样一个反例如图 2 所示,其中 S_0 为起始状态,S_g 为目标状态,行动 a_1 和 a_2 在任何状态处的立即回报均为–1.

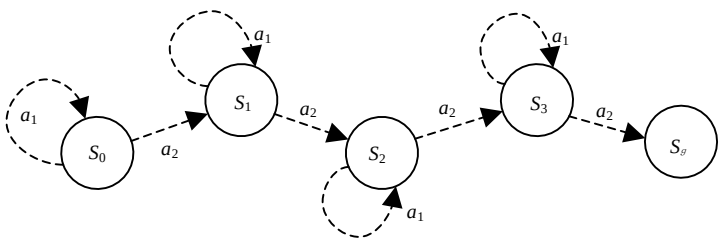

Fig.2 An example of multi-steps decision problem

图 2 一个多步决策问题的例子

由图 2 易知, $V^*(S_0) = -4,\ V^*(S_1) = -3$. 令 ε =1,可以初始化上界、下界如下:

$$V_U(s_0) = 0, V_L(s_0) = -4, V_U(s_1) = -3, V_L(s_1) = -100.$$

由式(7)可得: $Q_U^{a_1}(s_0) = -1, Q_L^{a_1}(s_0) = -5, Q_U^{a_2}(s_0) = -4, Q_L^{a_2}(s_0) = -101$.进而在起始状态 s_0,判据(II)满足且第 1 步可以执行 a_1,然后状态仍在 s_0 处,并可以这样重复下去.显然,整个策略不是ε最优的.出现这种结果的本质原因是初始化的下界不满足一致性,或者说假设能够获得下界期望回报的基础在后续执行过程中未能成立.

事实上,在一个同时使用上、下界的实时动态规划算法中,我们有下面的利用最优行动判据增量获取ε最优策略的定理.

定理 2. 给定一个 MDP 问题,满足:

(S1) 问题存在最优策略,且所有最优策略都能在有限步内结束.

(S2) 具有满足一致性条件的下界值函数: $V_L(s) \le \max_a \left[R(s,a) + \sum_{s' \in S} T^a(s,s') V_L(s') \right]$.

如果一个策略 π_{Inc} 以如下方式增量获取:在每步决策过程中,进行策略求解,假设当前状态为 s, 当最优行动判据 $V_U^{-a_L}(s) - V_L(s) \le \varepsilon$ 满足时,令 $\pi_{Inc}(s) = a_L$ 并且行动 a_L 被立即执行,进入下一步决策,重复此过程直到问题结束,那么,有 π_{Inc} 为 ε 最优.

证明:假设每步执行的 a_L 与最优策略对应的行动一致,条件(S1)保证过程在有限步内结束,且结果是 $V^* - V_{\pi_{Inc}} = 0$; 否则,假设在任意一步,执行的行动背离了最优策略,即 $V^*(s) \ne a_L$, 条件(S2)及定理1保证当前下界值函数对应的一步前瞻贪婪策略 π_{LG} 已经为 ε 最优,而继续求解都是对 π_{LG} 的改进,策略 π_{Inc} 不会比 π_{LG} 更差,即 $V^* - V_{\pi_{Inc}} \le \varepsilon$. □

按照定理2给出的方法,在软实时系统中,当前状态一旦满足最优行动判据,策略被固定,就可以马上执行一步.好处是执行后获得观察,而观察可以消除不确定性,比如执行行动后到底进入了哪个状态,那么此前有可能进入的其他一些分支对应的策略计算就可以避免了.注意到这里没有对状态空间是否有限的要求,换句话说,它可以被扩展到具有无限信念状态空间的 POMDP 问题上.

推论 2. 在对一个满足条件(S1)和条件(S2)的 MDP 问题的求解过程中,对任意状态 s,当 $V_U^{-a_L}(s) - V_L(s) \le \varepsilon$ 时,可以标定此状态已经被求解,令 $\pi(s) = a_L$.以这种方式最终确定的策略π为ε最优.

证明:假设 $\pi^*(s) = a_L$,将该状态处策略标定为 a_L 显然没有问题;假设 $\pi^*(s) \ne a_L$,状态 s 处期望回报最大为 $V_U^{-a_L}(s)$,由于 $V_U^{-a_L}(s) - V_L(s) \le \varepsilon$,且下界满足一致性,即 $V^*(s) - V_{\pi_{LG}}(s) \le \varepsilon$,也就是说,状态 s 处的一个ε策略已经获得了. □

在以这种方式求解策略时,任何一个状态处的计算分为两个阶段:在第1阶段,状态未被标定已求解,在此处进行值迭代需要考虑所有行动;第 2 阶段,最优行动判据满足,该状态处行动固定,值迭代时无须考虑其他行动.推论2给出了一种利用最优行动判据标定状态已被求解的方法.它在各类结合值函数上、下界的离线或在线算法中都可以应用.

实时规划的另一个重要特点是,当前状态处的行动是要被马上执行的,执行之后又会有计算时间进行后续决策,因此,系统对当前步策略的需求更加紧迫.事实上,利用新的判据,可以设计不同的分支选择策略,优先确定当前状态的行动.下一节我们讨论与此相关的内容.

3 分支选择策略

3.1 现有相关算法的分支选择策略

实时动态规划的分支选择包括行动选择和后继状态选择,每次Trail从初始状态开始,反复选择行动及行动后继状态直到边缘节点,扩展边缘节点,然后按Bellman公式反向值迭代回到初始状态,完成一次更新.RTDP与LRTDP的分支选择策略基本一致,行动选择由启发式信息采用一步前瞻的贪婪策略,后继状态的选择采用随机策略.BRTDP及FRTDP使用了区间估计的技术,使分支选择的标准从期望收益最大化变成了尽可能减小区间不确定性.在行动选择方面,FRTDP与基于POMDP的HSVI算法非常类似[13],基于判据(I),初始状态的不确定性 $\hat{V}$ 是由值函数上界V_U与下界V_L确定的.式(8)指出两者分别对应一个行动值函数Q_U和Q_L及行动a_U和a_L.而 Q_U^a 和 Q_L^a 又是由一组后继状态确定的.为了减小 $\hat{V}$,应该在这组后继状态中选择下面的分支.按照贪婪策略,最直接的做法就是在 a_U 或 a_L 的后继状态中选择.FRTDP 与 HSVI 在这一问题上都是选择 a_U.原因如下:首先需要知道两个结论,更新上界值函数只会使它变小,而对下界值函数的更新只会使其变大.如果选择总是采用 a_L 的策略,在树结构中,其他行动的值函数下界不会得到更新,由于 a_L 已经对应当前最大的行动值函数下界,以后在该状态处永远都只会选择这个 a_L,计算已经变成对这个策略的评价而不存在改进了.选择 a_U 则不存在这个问题,对上界更新会使其下降,在后面的过程中,其他行动值函数有可能成为上界进而被选中[13].事实上,在图结构中并非如此,这是因为不同的行动可能有相同的后继状态,即使选择 a_L,其他行动值函数下界也是有可能得到更新的.但无论如何,

FRTDP 这种行动选择策略都是一种高效的做法.对于后继状态的选择,使用区间估计算法的基本思路都是尽量选择值函数不确定性更大的后继状态,本文对此不作过多讨论.下面主要针对行动选择给出一个更适合判据(II)的分支策略.

3.2 针对实时决策的行动选择策略

相对于 $\hat{V}$, 判据(II)所描述的可以被称为行动不确定性,记为 $\hat{V}^a:\hat{V}^a=V_U^{-a_L}-V_L\le\varepsilon$. 为了减少 $\hat{V}^a$, 按照同样的思路,应该选择除 a_L 外拥有最大值函数的行动,记为 a_U^{-L}. 事实上,这一过程是在不断地尝试证明 a_U^{-L} 最乐观的期望回报也不比 a_L 的保守期望回报高出ε以上.并且,选择 a_U^{-L} 使得其上界下降的同时,其下界也会上升,有可能取代 a_L 成为新的对应最大下界的行动,进而其他行动也有机会被选定为 a_U^{-L} 继续更新.要说明的一点是,基于判据(II)的分支选择只在对应实时系统当前状态时使用,因为我们要确定的只是第 1 步.

这里涉及到的另一个问题是下界一致性.定理2要求下界一致才能保证ε最优.事实上,在实时系统中,即便初始化的下界不满足一致性,仍有解决方法.设初始化的下界为V_L,此时下界对应行动是未知的,不妨记为$a_?$,当 $V_L(s)>\max_a\left[R(s,a)+\sum_{s'\in S}T^a(s,s')V_L(s')\right]$时,下界没有得到更新,对应行动仍是$a_?$,此时选择 a_U^{-L} 其实就是a_U了.这与基于判据(I)的选择一致.由于$a_?$没有确定,算法也不会给出满足ε最优策略的行动供执行.只有当前状态一致性条件首次就得以满足后,V_L才被更新,事实上,此后该状态处一致性条件将会始终得到满足.

4 实时算法设计

有界增量最优实时动态规划算法(BI-RTDP)是以 FRTDP 为基础的,结合了新的基于最优行动判据的分支选择策略及已求解状态标定的技术.

每步决策时,当算法获得一个满足最优行动判据的解或者当前步决策时间用尽时,BI-RTDP 会返回一个行动供立刻执行.执行之后,行动的结果被观察到,算法继续下一步的决策.流程见算法 2.本节后面的内容集中介绍算法在 FRTDP 上扩展的部分,其他实现细节可以参考 FRTDP 算法实现本身[10].

算法 2. BIRTDP.

Function *trialRecurse*(*s*,*w*,*depth*)
  if *isTerminal*(*s*,*depth*) **or** *TIMEOUT*() **then return end if**
    trackUpdateQuality(*s*,*w*,*depth*)
    $a^*\leftarrow$*chooseGreedyAction*(*s*,*depth*)
    $s'\leftarrow$*chooseMaxPriSuccessor* (s,a^*)
    if *depth*<max_*depth* **then** *trialRecurse*$(s',\gamma T^{a^*}(s,s')w,depth+1)$ **end if**
    update (s)
Function *BIRTDP* (s_0,ε)
  while $V_U^{-a_L}-V_L>\varepsilon$
    if *TIMEOUT*() **then break end if**
    trialRecurse $(s_0, w=1, depth=0)$
    adjustTrailDepth()
  end while
  if *monotoniclowerbound*(s_0) **return** a_L
  else return a_U **end if**
Function *Run* (ε)
 (max_*depth*,*s*)$\leftarrow$(*INIT_DEPTH*, *INIT_STATE*)
  while !*isTerminal* (s)
    $a^*\leftarrow$*BIRTDP* (s,ε)
    $s\leftarrow$*execute* (a^*)

max_*depth*←max(*INIT_DEPTH*,max_*depth*−1)
end while

(1) *chooseGreedyAction*()在搜索深度为 1,即实时环境当前所在状态处,行动分支选择按第 4.2 节所述策略,是在最大下界对应行动之外的其他行动中选取的.当下界不满足不一致性或者对未来状态进行规划时,候选集合是所有行动.此外,对于已标记求解的状态,无须进行选择,直接返回对应状态的下界行动 a_L.

(2) *update*()操作基于推论 2 会标定已被求解的状态.若状态未被标定,则更新所有行动,否则只更新标定的行动.通过最优行动判据标记状态已解使用条件 $V_U^{-a_L}(s)-V_L(s)\le\varepsilon/2$.这里将ε替换为ε/2 的原因与 FRTDP 中一次 Trial 的返回条件使用 $V_U(s)-V_L(s)\le\varepsilon/2$ 类似,通常这样更有利于收敛.

(3) FRTDP 有一个自适应深度调整机制 *adjustTrailDepth*(),当前设定最大深度内更新效果不明显时,参数 max_*depth* 会增加.而在 BIRTDP 的实时运行模式中,每执行一步后,最大深度同时也作减 1 的调整.

(4) *BIRTDP*()返回当前状态最大值函数下界对应行动供执行,但在下界不满足一致性条件时,仍返回最大值函数上界对应的行动.原因是下界代表可行解,即当前已经找到的最好的解的情况,不满足一致性则意味着还未找到从当前状态达到终点状态的任何解,而上界代表启发式信息,在不得不执行行动时通常按贪婪策略基于上界选择是更好的方式.

(5) 在硬实时环境的实时计算中,*TIMEOUT*()也将引起递归调用返回.

(6) *execute*()与环境交互执行行动,并返回观察到的系统实际进入的状态.

5 实 验

实验采用被大量相关算法用作标准的测试问题 RaceTrack[6].RaceTrack 是一个赛车问题,在有障碍物的地图上给定初始位置与目标位置,状态由二维的位置及二维的速度 4 个量组成,记为 $(x,y,\dot{x},\dot{y})$. 行动是可以对赛车施加的一个加速度 $(\ddot{x},\ddot{y})$, $\ddot{x}$ 及 $\ddot{y}$ 取值为整数集合{0,−1,1}.每次加速有一定概率发生滑动,导致加速度为(0,0).赛车撞墙会被置于初始位置重新跑.在任何状态执行行动都会获得立即回报−1,赛车达到目标位置后过程结束.

测试选用标准地图 small-b 与 large-b[6].记地图左下角坐标为(0,0),这里选择固定起点,small-b 起点定为(0,5),large-b 为 (1,1) ,终点维持与标准地图不变.并有另外两个稍加修改的地图分别称为 small-b-m 与 large-b-m, small-b-m 是在地图坐标(7,15)处增加一个终点状态,large-b 是在正中部赛道一侧(33,11)−(33,17)处增加几个终点.这种修改不失一般性,目的在于使问题具有从起点到达终点解决问题之外更多的无关状态空间.这里统一使用默认滑动概率 0.1,上界初始为 0,下界初始为−1000.整个实验使用的计算环境为 Windows XP,AMD64 3200+ CPU,1G RAM.实验分为两部分,分别模拟软实时与硬实时两种不同的实时环境:

1. 软实时:每次实验开始,赛车被置为起点,算法给出自己有质量保证的策略后,提交并执行一步,系统模拟行动的不确定性结果进入下一状态,此过程循环直至到达终点状态.由于 RTDP 不处理停止条件,并不支持软实时测试,这里只选择 LRTDP,FRTDP 作为对比.每个算法共进行 500 次测试取平均值,实验结果见表 1.其中,Backup 为总迭代次数,它是算法效率最关键的指标;Time 为总决策时间,单位为秒;由于每步执行立即收益都是−1,Reward 取负值即为到达终点花费的总步数,ε 为精度要求.要说明的一点是,上下界在每步执行后都继续使用,直到一次测试到达终点才重置为初始值进行下次测试.
2. 硬实时:系统限定每步决策时间,TIMEOUT 时要求算法给出当前状态的行动供执行.采用 RTDP, LRTDP,FRTDP 进行对比实验,每个算法测试 1 000 次取平均,这里记录的是到达终点的平均步数 Steps, 在此问题中它就是 Reward 取负的值,实验结果如图 3 所示.

实验结果整体上显示 BI-RTDP 具有更好的实时特性.此外,从软实时测试结果可以看出,BI-RTDP 在两类情况下优势更加明显:1) 问题给定了更多的无关状态空间;2) 要求更高的精度.而这两种情况事实上都体现了 BI-RTDP 只把计算集中在寻找每步最优行动,并不要求获得对应的精确期望回报,也因而能够在环境中更早地行动并达到目标.从另一方面来看,使用最优行动判据的算法不必担心因给出了过多的无关状态空间或设定了过高的精度要求而花费过多额外的计算.

Table 1 Soft real-time test

表1 软实时测试

Algorithm	Backups	Time (s)	Reward	Backups	Time (s)	Reward
ε	0.1e−3			0		
small-b(9312s,9a)						
LRTDP	186 509	1.678	−13.410 0	N/A	N/A	N/A
FRTDP	141 583	1.429	**−13.216 0**	173021	1.476	−13.217 3
BI-RTDP	**130 846**	**1.372**	−13.227 8	**133 862**	**1.378**	**−13.279 3**
small-b-m(9312s,9a)						
LRTDP	317 958	1.812	−5.436 0	N/A	N/A	N/A
FRTDP	81 844	1.147	−5.466 0	118229	1.240	−5.700 0
BI-RTDP	**14 716**	**0.330**	**−5.394 0**	**14 879**	**0.334**	**−5.360 0**
large-b(23880s,9a)						
LRTDP	1 601 485	8.130	−23.338 0	N/A	N/A	N/A
FRTDP	591 606	3.724	−23.534 0	741 758	4.113	−23.720 0
BI-RTDP	**566 732**	**3.643**	**−22.874 0**	**604 322**	**3.647**	**−23.412 0**
large-b-m(23880s,9a)						
LRTDP	787 025	4.126	**−8.520 0**	N/A	N/A	N/A
FRTDP	53 660	1.102	−8.565 0	129 737	1.525	−8.526 2
BI-RTDP	**45 411**	**0.756**	−8.615 0	**98 801**	**0.974**	**−8.360 0**

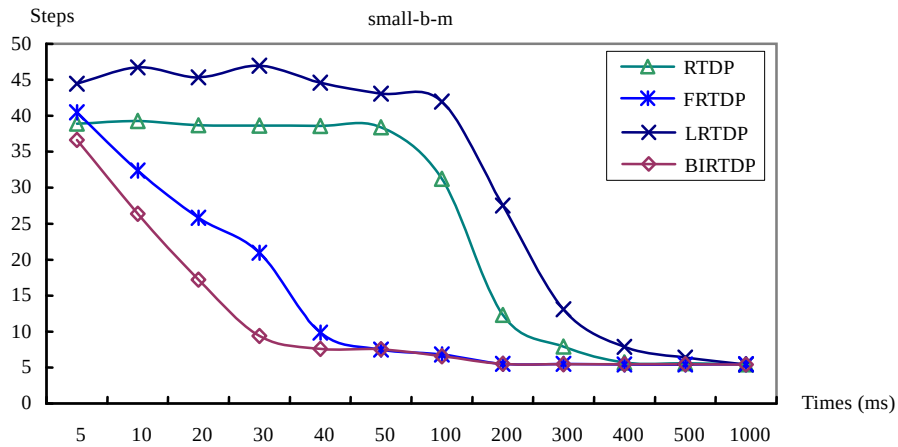

Fig.3 Hard real-time test

图3 硬实时测试

软实时测试中 Reward 有统计误差,但也可以看出精度要求下 BI-RTDP 解的质量不比相关算法差.硬实时测试显示 BI-RTDP 有明显更优的瞬时响应特性,如在每步决策只有 30ms 时,解的质量提高 1 倍以上,且达到接近最优解所需要的每步决策时间更短.这应该来源于 BI-RTDP 的分支选择策略,因为在给定的很短的每步决策时间内,各个判据在开始其实都得不到满足,行动的提交都是 TIMEOUT 引发的,所以解的质量的好坏最大程度上来源于分支选择策略.对于新的分支选择策略的一个更直观的理解是,相对于其他策略,它并不直接关心最优解能够好到多少,而是当前状态是否有哪个行动比其他行动已经足够好,以至于选择它之后未来仍能获得满足精度要求的最优策略,因而它的计算更集中于找出当前状态的最优行动.

6 结 论

本文的主要贡献是对现有实时动态规划算法的收敛判据及分支选择策略进行了对比分析,结合值函数上下界理论给出了一个新的最优行动判据,以及以此为基础的实时动态规划增量求解方法和一个更适合实时系统的分支选择策略.从本质上讲,增量求解更充分地利用了实时系统固有的决策与执行交替的特点,把求解过程的计算内容按策略被执行顺序作了更好的调整,利用新的判据优先确定当前步的行动,并通过一致性条件保证

此过程误差不会积累,每步执行依靠观察消除行动结果的不确定性,避免了相关的不必要的计算.

基于这一原理设计的 BI-RTDP 算法在两类典型的实时系统测试中都显示出优于当前最新相关算法的性能.BI-RTDP 受无关状态空间的影响更小,对精度的要求更不敏感.另外,在有严格响应时间限制的硬实时测试中,BI-RTDP 也显示了更好的实时响应特性.

本文给出的最优行动判据目前主要应用在实时决策中,用作在线计算.未来的工作希望能够测试最优行动判据标定已求解状态的方法在离线计算中的效果.另一方面,目前 FRTDP 算法采用的是一个自适应深度控制,这部分的分支控制仍然存在很大程度的盲目性,希望将来在结合上、下界做前向搜索的 RTDP 算法中尝试迭代加深(IDA*)的技术[14].

致谢 在此,我们向对本文的工作给予帮助与建议的 Hector Geffner 教授、Benjamin Johnston 博士以及讨论班上的宋志伟、吉建民、吴锋、徐文松、陈圆迷同学表示感谢.

References:

[1] Boutilier C, Dean T, Hanks S. Decision-Theoretic planning: Structural assumptions and computational leverage. Journal of Artificial Intelligence Research, 1999,11:1−94.

[2] Hansen EA, Zilberstein S. LAO*: A heuristic search algorithm that finds solutions with loops. Artificial Intelligence, 2001,129(1-2): 35−62.

[3] Bonet B, Geffner H. Faster heuristic search algorithms for planning with uncertainty and full feedback. In: Proc. of the 18th Int'l Joint Conf. on Artificial Intelligence. Acapulco: Morgan Kaufmann Publishers, 2003. 1233−1238.

[4] Dean T, Kaelbling LP, Kirman J, Nicholson A. Planning under time constraints in stochastic domains. Artificial Intelligence, 1995, 76(1-2):35−74.

[5] Ferguson D, Stentz A. 2004. Focused dynamic programming: Extensive comparative results, Technical Report, CMU-RI-TR-04-13, Pittsburgh: Robotics Institute, Carnegie Mellon University, 2004.

[6] Barto AG, Bradtke SJ, Singh SP. Learning to act using real-time dynamic programming. Artificial Intelligence, 1995,72(1-2): 81−138.

[7] Pemberton JC, Korf RE. Incremental search algorithms for real-time decision making. In: Proc. of the 2nd Artificial Intelligence Planning Systems Conf. 1994. 140−145.

[8] Bonet B, Geffner H. Labeled RTDP: Improving the convergence of real-time dynamic programming. In: Giunchiglia E, Muscettola N, Nau D, eds. Proc. of the ICAPS 2003. AAAI Press, 2003. 12−21.

[9] McMahan HB, Likhachev M, Gordon GJ. Bounded real-time dynamic programming: RTDP with monotone upper bounds and performance guarantees. In: Proc. of the 22nd Int'l Conf. on Machine learning. 2005.

[10] Smith T, Simmons R. Focused real-time dynamic programming for MDPs: Squeezing More Out of a Heuristic. In: Proc. of the 21st AAAI Conf. on Artificial Intelligence. AAAI Press, 2006.

[11] Liu CL, Layland JW. Scheduling algorithms for multiprogramming in a hard-real-time environment. Journal of the ACM, 1973, 20(1):46−61.

[12] Puterman ML. Markov Decision Processes—Discrete Stochastic Dynamic Programming. New York: John Wiley and Sons, 1994.

[13] Smith T, Simmons R. Heuristic search value iteration for POMDPs. In: Chickering M, Halpern J, eds. Proc. of the Int'l Conf. on Uncertainty in Artificial Intelligence (UAI). 2004. Arlington: AUAI Press, 2004.

[14] Bonet B, Geffner H. (2006). Learning depth-first search: A unified approach to heuristic search in deterministic and non-deterministic settings, and its application to MDPs. In: Proc. of the 16th Int'l Conf. on Automated Planning and Scheduling (ICAPS 2006). 2006.

范长杰(1981－),男,河南卫辉人,博士,主要研究领域为人工智能马尔可夫决策理论,多智能体合作及对抗.

陈小平(1955－),男,博士,教授,博士生导师,主要研究领域为人工智能逻辑,多智能体系统,自主机器人系统关键技术.