文章编号: 1001-0920 (2013) 06-0925-05

基于 Monte Carlo 粒子滤波的 POMDPs 在线算法

仵 博[1,2], 吴 敏[1]

(1. 中南大学 a. 信息科学与工程学院，b. 先进控制与智能自动化湖南省工程实验室，
长沙 410083；2. 深圳职业技术学院 教育技术与信息中心，广东 深圳 518055)

摘 要: 针对部分可观察马尔可夫决策过程 (POMDPs) 的信念状态空间是一个双指数规模问题, 提出一种基于 Monte Carlo 粒子滤波的 POMDPs 在线算法. 首先, 分别采用粒子滤波和粒子映射更新和扩展信念状态, 建立可达信念状态与或树; 然后, 采用分支界限裁剪方法对信念状态与或树进行裁剪, 降低求解规模. 实验结果表明, 所提出算法具有较低的误差率和较快的收敛性, 能够满足系统实时性的要求.

关键词: 部分可观察马尔可夫决策过程；信念状态；Monte Carlo；粒子滤波；在线算法

中图分类号: TP273 **文献标志码:** A

Online algorithm based on Monte Carlo particle filtering in POMDPs

WU Bo[1,2], *WU Min*[1]

(1a. School of Information Science and Engineering，1b. Hu'nan Engineering Laboratory for Advanced Control and Intelligent Automation，Central South University，Changsha 410083，China；2. Education Technology and Information Centre，Shenzhen Polytechnic，Shenzhen 518055，China．Correspondent：WU Bo，E-mail：wubo@szpt.edu.cn)

Abstract: In order to solve the double exponential size problem of belief states space in partially observable Markov decision processes(POMDPs), an online algorithm based on Monte Carlo particle filtering(MCPF) is proposed. Firstly, the methods of particle filtering and particle projection are used to update and expand the belief states respectively, and the and-or tree of reachable belief states is built. Then, a branch-and-bound pruning method is proposed to prune the tree to reduce computation. Finally, the experiment and simulation results show that the proposed algorithm has the effectiveness in retaining the quality of the policies and reducing the cost of computing policies, so it can meet the requirement of a real-time system.

Key words: partially observable Markov decision processes；belief states；Monte Carlo；particle filtering；online algorithm

0 引 言

部分可观察马尔可夫决策过程[1](POMDPs) 能够客观准确地描述真实世界, 是研究随机决策过程的重要分支, 并成为计算机、控制和管理等学科的研究热点. 综述现有算法, 按照年代可大致分为两个阶段: 1) 20 世纪末主要是精确求解算法, 代表算法为 Witness 算法[1]; 2) 21 世纪初, 由于精确求解 POMDPs 过程涉及到信念状态空间维数灾和迭代更新历史灾问题, 主要是近似求解算法, 代表算法为基于点的值迭代算法[2-3]. 但是在序贯决策中, 信念状态空间会随着时间的推移而呈指数形式爆炸增长[4-5].

近几年, 为了使理想的 POMDPs 模型能够满足实时系统的要求, 普遍采用在线近似算法来求解[6], 主要有蒙特卡罗采样算法、分支界限裁剪算法和启发式搜索算法. 与离线算法求解全局最优策略或全局近似最优策略不同, 在线算法只求解当前最优策略或当前近似最优策略, 根据时间约束条件, 将整个策略规划和策略执行分成若干个小的规划和执行, 每个时间片内分别进行策略规划和策略执行. Paquet[7]提出一种 RTBSS 算法, 其效率在很大程度上依赖于离线求解上下界的精确度, 当上下界紧密时, 查找算法高效. 但是该算法忽略了观察集合, 当观察集合较大时, 为了在有限时间内求解问题, 遍历深度应该很小, 但这又会增加最优策略的误差. Wolf 等[8]在 RTBSS 算

收稿日期: 2012-02-10；**修回日期:** 2012-04-25.

基金项目: 国家自然科学基金项目(61074058, 60874042)；教育部博士点基金项目(20090162120068)；广东省自然科学基金项目(S2011040004769).

作者简介: 仵博(1979−), 男, 副教授, 博士生, 从事序贯决策、机器学习的研究；吴敏(1963−), 男, 教授, 博士生导师, 从事过程控制、鲁棒控制等研究.

法的基础上提出了一种基于蒙特卡罗采样的MC-RTBSS算法. Ross等[9]提出了AEMS2算法, 该算法避免对观察或动作分支进行裁剪, 通过使用启发式方法选择最佳的扩展边缘结点, 从而查找出与决策相关性最高的可达信念状态点. 但是, AEMS2算法需要计算扩展的边缘结点, 还要在每次迭代中更新父亲结点的值, 因此比一般的深度优先和广度优先算法更为耗时.

本文借鉴MC-RTBSS的主要思想, 提出一种基于Monte Carlo粒子滤波(MCPF)的POMDPs在线算法. 采用粒子滤波更新信念状态, 使用粒子映射扩展信念状态, 在此基础上建立可达信念状态与或树(AOT). 采用深度优先方法遍历与或树, 将离线求解的值函数上下界作为判定条件, 对可达信念状态与或树采用分支界限裁剪方法进行裁剪, 从而降低求解规模, 加速求解过程.

本文除特殊说明外, 上标代表时间, 下标代表集合中的具体实例, 如s_i代表状态集合中的第i个状态, s^t代表t时刻的状态, $P(s^t = s_i)$代表t时刻状态为s_i时的概率. 有时根据描述问题的需要, 没有上下标则表示当前时刻的变量, 有上标$'$则表示下一时刻的变量, 如s表示当前时刻的状态, s'表示下一时刻的状态.

1 POMDPs

POMDPs通常描述为六元组$\langle S, A, T, R, Z, O\rangle$[10]. 状态集合$S = \{s_1, s_2, \cdots, s_n\}$、动作集合$A = \{a_1, a_2, \cdots, a_m\}$、观察集合$Z = \{z_1, z_2, \cdots, z_l\}$分别为所有可能的状态、动作和观察; 状态转移函数集合$T(s_i, a, s_j) = P(s_i|a, s_j)$为在状态$s_i$下采用动作$a$可能转移到状态$s_j$的概率; 观察函数集合$O(s_i, a_j, z_k) = P(z_k|s_i, a_j)$为在状态$s_i$下采用动作$a_j$观察为$z_k$的概率; 报酬函数集合$R(s, a) : S \times A \times Z \mapsto \boldsymbol{R}$. t时刻的信念状态b的更新方程为

$$\tau(b, a, z) = \eta \sum_{s' \in S} O(s', a, z) T(s, a, s') b(s), \tag{1}$$

$$P(z|b, a) = \sum_{s' \in S} O(s', a, z) \sum_{s \in S} T(s, a, s') b(s), \tag{2}$$

其中$\eta = 1/P(z^t|b^{t-1}, a^{t-1})$为归一化因子.

信念状态报酬函数$\rho(b, a) = \sum_{s \in S} b(s) R(s, a)$. 使用Bellman公式计算POMDPs的值函数V, 有

$$V^{t+1}(b) = \max_{a \in A} \Big[\sum_{s \in S} R(s, a) b(s) + \gamma \sum_{z \in Z} P(z|b, a) V^t(b) \Big]. \tag{3}$$

POMDPs决策的策略是将信念状态映射到动作, 即$\pi(b) \to a$, 同时满足使智能体选择的动作能够获得环境报酬的累计值最大, 最优策略是使折扣报酬值和的期望值最大, 即

$$\pi^* = \arg\max_{a \in A} \mathrm{E}\Big[\sum_{k=1}^{\infty} \gamma^t \sum_{s \in S} R(s, \pi(b)) b(s) \Big]. \tag{4}$$

2 Monte Carlo粒子滤波算法

2.1 在线算法思想和理论基础

POMDPs离线算法求解的是全局最优策略或全局近似最优策略, 分为策略规划和策略执行两个阶段. 假设给定一个初始的信念点, 要解出在这点达到理论上近似最优的值函数, 理论上只需通过无穷多次组合所有的行为和观测值, 得到所有能从初始信念点可达的信念点(这些点的集合称为可达信念状态空间), 然后在该可达信念状态空间上作迭代即可. 但这样的点可能有无穷多个, 所以无法进行操作. 在线算法求解的是当前最优策略或当前近似最优策略, 由于在线算法只考虑当前信念状态空间下的最优策略, 可达信念状态空间很小, 利用已有的基于点的迭代算法可以很快求解值函数.

在线算法将整个决策周期分为若干个小的时间片, 每个时间片内分别进行策略规划和策略执行. 在规划阶段, 根据当前信念状态计算出最优执行动作, 可分为两个步骤:

1) 建立可达信念状态与或树, 见图1. 在AOT中, 根节点为当前信念状态, 孩子节点为可达信念状态点, 可达信念状态点由式(1)计算. 信念结点为OR结点, 输出弧权重为$R_B(b, a)$. 两层信念点之间的结点为AND结点, 输出弧权重为$P(z|b, a)$, 由式(2)计算.

2) 由式(3)计算当前信念状态的值函数, 计算过程通过边缘结点的向上传播算法实现. 在执行阶段, 根据式(4)获得当前信念状态下的近似最优动作, 并根据观察更新当前信念状态和AOT.

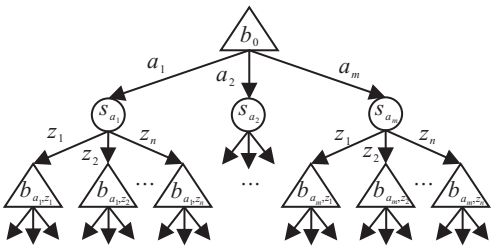

图1 可达信念状态与或树

定理1 令$\tilde{V}$为近似值函数, 用于评估边缘结点的值, V^*为最优值函数, b为信念状态, $b \in [0, 1]$, $\epsilon = \sup_b |V^*(b) - \tilde{V}(b)|$, D步迭代的近似值函数$\tilde{V}^D(b)$的误差边界为

$$|V^*(b) - \tilde{V}^D(b)| \leqslant \gamma^D \epsilon. \tag{5}$$

定理1的证明见文献[11]. 由定理1可知, 当D等于全局决策时间长度T时, 在线算法求解结果与

离线算法求解结果相同. 因此, 可以将在线算法看成离线算法的特例, 除时间长度不一样外, 其他均相同. 在线算法对与或树进行 D 次迭代后, 求出当前信念状态下最大的报酬值, 并返回该报酬值对应的最佳动作, 从而完成本时间片内的决策, 其时间复杂度为 $O(|A||Z|)^D$.

2.2 信念状态粒子滤波更新操作

为了处理连续的状态空间, 本文采用带权粒子来描述信念状态 $b=\langle W,S\rangle$. 其中: $S=\{s_1,s_2,\cdots,s_{N_p}\}$ 为样本状态集合, $W=\{w_1,w_2,\cdots,w_{N_p}\}$ 为权重集合, N_p 为信念状态的粒子数. 采用粒子滤波更新信念状态步骤如下.

Step 1: 当 $n<N_p$ 时, 执行样本采样操作.

Step 1.1: 在 b 中采样 s;

Step 1.2: 根据 $T(s,a,\cdot)$ 采样 s'_n;

Step 1.3: 赋值粒子权重 $w_n=O(s'_n,a,z)$;

Step 1.4: 将 $\langle s'_n,w'_n\rangle$ 添加到 b' 中;

Step 1.5: 令 $n=n+1$, 返回 Step 1.

Step 2: 对 b' 中的权重进行归一化操作.

Step 3: 得到更新后的信念状态 b'.

2.3 信念状态粒子映射扩展操作

信念状态的扩展决定信念状态空间的大小, 本文采用粒子映射方法, 具体步骤如下.

Step 1: 当 $n<N_p$ 时, 执行样本采样操作.

Step 1.1: 在 b 中采样 s;

Step 1.2: 根据 $T(s,a,\cdot)$ 采样 s'_n;

Step 1.3: 令 $n=n+1$, 返回 Step 1.

Step 2: 当 $m<N_{|Z|}$ 时, 执行样本采样操作.

Step 2.1: 在 b 中采样 s;

Step 2.2: 根据 $T(x,a,\cdot)$ 采样 x';

Step 2.3: 根据 $O(x',a,\cdot)$ 采样 z_m;

Step 2.4: 当 $n<N_p$ 时, 赋值粒子权重 $w_n=O(s'_n,a,z_m)$, 并将 $\langle s'_n,w'_n\rangle$ 添加到 b'_{a,z_m} 中;

Step 2.5: 对 b'_{a,z_m} 中的权重进行归一化操作;

Step 2.6: 令 $m=m+1$, 返回 Step 2.

Step 3: 得到新扩展的信念状态集合 $\{b'_{a,z_1},b'_{a,z_2},\cdots,b'_{a,z_{N_{|Z|}}}\}$.

2.4 MCPF 算法描述

为了将 Bellman 方程应用于在线算法, 将值函数和最优策略进行重新定义, 有

$$\pi^*(b,d)=\arg\max_{a\in A}[V^d(b,d)]. \tag{6}$$

$$V^d(b,d)=\begin{cases}\max\limits_{a\in A}\sum\limits_{s\in S}b(s)R(b,a),\ d=0;\\ \max\limits_{a\in A}\Big[\sum\limits_{s\in S}b(s)R(b,a)+\\ \gamma\sum\limits_{z\in Z}V^{d-1}(\tau(b,a,z),d-1)\Big],\ d>0.\end{cases} \tag{7}$$

通过信念状态粒子滤波更新操作和粒子映射扩展操作, 生成可达信念状态 AOT, 采用分支界限裁剪方法对 AOT 进行裁剪, 降低求解问题的规模. MCPF 算法主要步骤如下.

Step 1: 初始化输入参数操作. 输入参数为当前信念状态 b 和与或树 T 的深度 d. 其中: D 为最大扩展深度, $a_{\rm best}$ 为最佳动作, $R_{\max}(b)$ 为最大报酬值, R_c 为当前报酬值, $U_T(b)$ 为报酬值上界, b_c 为当前信念状态点. 输出为 $R_{\max}(b)$.

Step 2: 如果 $d=0$, 则计算边缘结点的最大报酬值为 $\max\limits_{a\in A}\sum\limits_{s\in S}b(s)R(s,a)$, 将计算结果赋值给 $R_c\leftarrow R_B(b,a)$.

Step 3: 在当前信念状态下, 根据 $U(b,a_i)\geqslant U(b,a_j)(i\leqslant j)$, 按照动作可能发生的概率大小对动作集合进行大根堆排序 HeapSort(b, A).

Step 4: 信念状态粒子映射扩展操作. 深度优先遍历与或树, 如果 $U_T(b)>R_{\max}(b)$, 则 $R_c=R_c+\gamma P(z|a,b)$MCPF$(\tau(b,a,z),d-1)$; 如果 $R_c>R_{\max}(b)$, 则 $R_{\max}(b)\leftarrow R_c$.

Step 5: 如果 $(d=D)\bigcup\|V^*(b)-R_{\max}(b)\|<\epsilon$ 成立, 终止条件满足, 则获得最佳动作 $a_{\rm best}\leftarrow a$; 否则, 返回 Step 2.

Step 6: 信念状态粒子滤波更新操作.

2.5 MCPF 算法复杂度分析

令 $|A|$ 为动作个数, $|Z|$ 为观察个数. 由定理 1 可知, 在最坏的情况下, 处在第 d 层的 AOT 有 $(N_{|Z|}|A|)^d$ 个信念结点. Step 3 根据当前信念状态对动作发生概率从大到小排序, 在 AOT 中, 首先深度遍历概率值大的动作, 可以加快与或树的裁剪速度. 本文采用大根堆排序算法, 其最坏时间复杂度为 $O(|A|\log|A|)$, 辅助存储空间为 $O(1)$. Step 4 深度优先遍历 AOT, 并使用分支界限裁剪方法对 AOT 进行裁剪, 当值函数上界不大于当前最大值时, 裁减该动作的子树. 假设被保留的动作子树个数为 N_b, 则采用粒子映射操作将扩展 N_b 个粒子. Step 6 是粒子滤波更新操作, 产生 N_p 个粒子. 综上可知, MCPF 算法在最坏情况下的时间复杂度为 $O(|A|\log|A|+(N_p+N_b|A|)(N_{|Z|}|A|)^D)$, 最坏情况下的空间复杂度为 $O((N_{|Z|}|A|)^{D-1})$, 其中 D 为 AOT 的最大深度.

2.6 误差分析

定理 2 对于任何信念状态 b 和 AOT 深度 d, 当效用函数 $U(b)=R_B(b)$ 时, MCPF 算法的最大误差为

$$|\mathrm{MCPF}(b,d)-V^*(b)|\leqslant$$
$$\gamma^d(|R_{\max}-V_{\min}|,|R_{\min}-V_{\max}|), \tag{8}$$

其中 $R_{\max}$ 和 $R_{\min}$ 分别为最大和最小报酬值, 并满足

$$V_{\max}=\sum_{i=0}^{\infty}\gamma^i R_{\max}=\frac{R_{\max}}{1-\gamma}, \tag{9}$$

$$V_{\min}=\sum_{i=0}^{\infty}\gamma^i R_{\min}=\frac{R_{\min}}{1-\gamma}. \tag{10}$$

证明 如果 $U(b)=R_B(b)$, 最优值函数为 $V_d^*(b)$, 则 MCPF $(b,d)=V_d^*(b)$. 由三角不等式可知

$$|\mathrm{MCPF}(b,d)-V^*(b)|\leqslant$$
$$|\mathrm{MCPF}(b,d)-V_d^*(b)|+|V_d^*(b)-V^*(b)|.$$

不等式右边的第 1 个绝对值的结果为 0, 根据定理 1 的结论可知

$$|V_d^*(b)-V^*(b)|\leqslant\gamma^d|V_0^*(b)-V^*(b)|=$$
$$\gamma^d|R_B(b)-V^*(b)|\leqslant\gamma^d(|R_{\max}-V_{\min}|,|R_{\min}-V_{\max}|).$$

即式 (8) 成立. □

当 $R_B(b)=R_{\max}$ 且 $V^*(b)=V_{\min}$, 或者 $R_B(b)=R_{\min}$ 且 $V^*(b)=V_{\max}$ 时, $|R_B(b)-V^*(b)|$ 最大, 该值即为 MCPF 算法在当前时刻的最大误差.

3 实验仿真

本文从两个角度评价 MCPF 算法的有效性. 第 1 是针对 RockSample 问题[12], 评价不同遍历深度对 MCPF 算法的影响, 获得其最优取值范围; 第 2 是针对 RoboCupRescue 仿真比赛进行算法有效性比较. 仿真平台为 MatlabR2010a, 运行环境为 32 位 Windows7, Intel(R) Core(TM) i3, CPU 为 3.07 GHz, RAM 为 4 GB.

3.1 RockSample 问题

RockSample 问题由 Smith 等于 2004 年提出, 它模拟机器人在火星上采集岩石, 通过获取采集岩石动作的报酬值来判断采集的岩石是否上乘. RockSample 问题的一般形式为 RockSample $[n,k]$, 表示 n 个格子, k 个岩石.

1) 状态集合 S. 每一个状态由 $k+1$ 个变量描述, X_p 为机器人所处的位置, 取值范围为 $\{(1,1),(1,2),\cdots,(n,n)\}$; k 个变量 $\{X_R^i\}$, 每个 X_R^i 表示一个岩石, 取值为 $\{\mathrm{Good},\mathrm{Bad}\}$, 终止状态为 EXIT. 因此, RockSample $[n,k]$ 问题有 $n^2\times 2^k+1$ 个状态.

2) 动作集合 A. 机器人有 $k+5$ 个动作, 取值范围为 {North, South, East, West, Sample, Check$_1$, $\cdots$, Check$_k$}. 前 4 个动作是确定的, 描述的是机器人的朝向; Sample 动作描述的是在当前位置下采集岩石, 当 $X_R^c=$ Good 时, 返回的报酬值 $R=10$, 并将 X_R^c 赋值为 Bad, 当 $X_R^c=$ Bad 时, $R=-10$, 当处在终止状态 EXIT 时, $R=10$; 其他动作不会有报酬值.

3) 观察集合 Z. Check$_i$ 动作是传感器对 X_R^i 进行 {Good, Bad} 评判, 返回带有噪音的观察值. 传感器越接近岩石, 观察信息越准确. 观察变量由效能因子 $\eta(X,i)$ 所决定, 有

$$\eta(X,i)=2^{-d(X_p,i)/d_0}. \tag{11}$$

其中: $d(X_p,i)$ 为 X_p 与 X_R^i 之间的欧几里得距离; d_0 为可调频常量, 称为半效能距离. 当 $d=d_0$ 时, $\eta=1/2$; 当 $\eta=1$ 时, 传感器始终返回正确值; 当 $\eta=0$ 时, 传感器返回 {Good} 和 {Bad} 值的几率均为 50%.

3.2 遍历深度实验

AOT 遍历深度 D 的取值对于在线算法至关重要, D 取值过大, 将影响算法的实时性; D 过小, 算法的误差较大. 图 2 是针对 MCPF 进行实验的结果, 实验对象为 RockSample [7, 8]. 由图 2 可知, 在遍历深度 $D=4$ 时, 算法基本上收敛; 当 $D>4$, 无法满足实时系统的时间约束条件; 当 $D<4$ 时, 算法的误差较大.

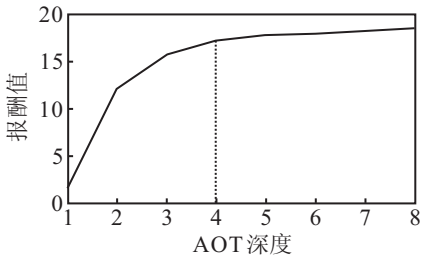

图 2 AOT 遍历深度实验

3.3 机器人救援系统实验

机器人救援系统是更为复杂的 POMDPs 问题, 其目标是尽可能地减少地震带来的破坏. 从警察的角度来看, 一个完整的状态由 1 500 个随机变量描述. 其中: 800 个 Roads 变量取值为 {blocked, cleared}; 700 个 Buildings 变量取值为 {fire, noFire}, Agent 位置变量 AgentsPosition, 它表示 30 ~ 40 个 Agents 可能处在 800 个道路上的任一位置. 警察 Agent 的状态空间 $|S|=2^{800}\times 2^{700}\times 800$[30~40], 信念状态空间更大. 衡量算法优劣的参数较多, 如营救的市民数量、着火点数量、受损房屋数量和被堵塞道路数量等.

CSU 队、SOS 队和 MRL 队市民死亡人数比较如图 3 所示. 实验结果表明, CSU 队在救援市民方面表现出色, 在 150 周期时完成所有市民的救援, 表明本文算法能够使整个队伍更高效地完成救援任务.

将本文算法应用于中南大学云麓队, 在 2011 年中国机器人大赛暨 RoboCup 公开赛中获得 RoboCup 机器人救援仿真组冠军. 从机器人救援系统的应用来

看, MCPF 算法可以应用于大规模的实时系统中.

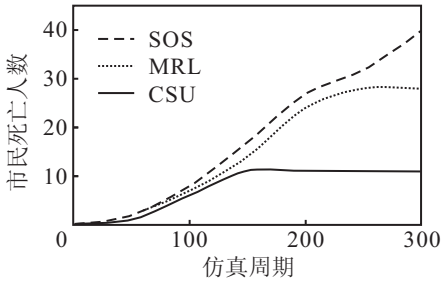

图 3　市民死亡人数比较

4 结　论

部分可观察马尔可夫是求解动态不确定随机环境下序贯决策的理想模型, 但由于其信念状态具有“维数灾”和“历史灾”问题, 只能针对小规模离线问题进行求解. 鉴于此提出一种 MCPF 算法, 实验结论表明, 所提出算法满足大规模实时系统的要求. 在实际 POMDPs 模型中, 信念状态往往具有较大的稀疏性. 如果在进行在线规划和执行前消除其稀疏性, 则会极大地提高 MCPF 的收敛速度. 可以对信念状态使用可分解描述简化求解过程, 并使用动态贝叶斯网络消除其稀疏性, 从而降低求解最优策略的计算量.

参考文献(References)

[1] Kaelbling L P, Littman M L, Cassandra A. Planning and acting in partially observable stochastic domains[J]. Artificial Intelligence, 1998, 101(2): 99-134.

[2] Kaplow R. Point-based POMDP solvers: Survey and comparative analysis[M]. Montreal: McGill University, 2010: 36-89.

[3] 孙湧, 仵博, 冯延蓬. 基于策略迭代值迭代的 POMDP 算法[J]. 计算机研究与发展, 2008, 45(10): 1763-1768.
(Sun Y, Wu B, Feng Y P. Policy-value iteration algorithm for POMDP[J]. J of Computer Research and Development, 2008, 45(10): 1763-1768.)

[4] Li X, Cheung W K, Liu J M. Improving POMDP tractability via belief compression and clustering[J]. IEEE Trans on Systems, Man and Cybernetics Part B: Cybernetics, 2010, 40(1): 125-136.

[5] 仵博, 吴敏. 一种基于信念状态压缩的实时 POMDP 算法[J]. 控制与决策, 2007, 22(12): 1417-1420.
(Wu B, Wu M. A real-time POMDP algorithm over belief states space compression[J]. Control and Decision, 2007, 22(12): 1417-1420.)

[6] Ross S, Pineau J, Paquet S, et al. Online planning algorithms for POMDPs[J]. J of Artificial Intellgence Research, 2008, 32(1): 663-704.

[7] Paquet S. Distributed decision making and task coordination in dynamic uncertain and real time multiagent environments[M]. Quebec: Laval University, 2006: 56-92.

[8] Wolf T B, Kochenderfer M J. Aircraft collision avoidance using Monte Carlo real-time belief sapce search[J]. J of Intelligent and Robotic Systems, 2011, 64(2): 277-298.

[9] Ross S, Pineau J, Chaib-draa B. Theoretical analysis of heuristic search methods for online POMDPs[C]. Proc of the NIPS. British Columbia: MIT Press, 2008: 1216-1225.

[10] Roy N, Gordon G. Finding approximate POMDPS solutions through belief compression[J]. J of Artificial Intelligence Research, 2005, 23(1): 1-40.

[11] Hauskrecht M. Value function approximations for partially observable Markov decision processes[J]. J of Artificial Intelligence Research, 2000, 13(1): 33-94.

[12] Smith T, Simmons R. Point-based POMDP algorithms: Improved analysis and implementation[C]. Proc of Int Conf on Uncertainty in Artificial Intelligence. Edinburgh: AAAI Press, 2005: 542-547.

(上接第924页)

[5] Fossen T I, Strand J P. Passive nonlinear observer design for ships using Lyapunov methods: Full scale experiments with a supply vessel[J]. Automatica, 1999, 35(1): 3-16.

[6] Stamnes O Y, Aamo O M, Kaasa G. Global output feedback tracking control of euler-lagrange systems[C]. Int Federation of Automatic Control. Milano, 2011: 215-220.

[7] Aamo O M, Arcak M, Fossen T I. Global output tracking control of a class of euler-lagrange systems with monotonic nonlinearities in the velocities[J]. Int J of Nonlinear Control, 2000, 74(7): 649-658.

[8] Roger S. The maneuvering problem[D]. Norway: Department of Engineering Cybernetics, Norwegian University of Science and Technology, 2005: 73-84.

[9] Wondergem M, Lefeber E, Pettersen K Y. Output feedback tracking of ships[J]. IEEE Trans on Control Systems Technology, 2011, 19(2): 442-448.

[10] Fossen T I, Grovlen A. Nonlinear output feedback control of dynamically positioned ships using vectorial observer backstepping[J]. IEEE Trans on Control Systems Technology, 1998, 1(6): 121-128.