控制与决策
Control and Decision

基于蒙特卡洛Q值函数的多智能体决策方法

张健， 潘耀宗， 杨海涛， 孙舒， 赵洪利

引用本文:
张健, 潘耀宗, 杨海涛, 等. 基于蒙特卡洛Q值函数的多智能体决策方法[J]. 控制与决策, 2020, 35(3): 637−644.

在线阅读 View online: https://doi.org/10.13195/j.kzyjc.2018.0796

您可能感兴趣的其他文章

Articles you may be interested in

犹豫模糊语言PROMETHEE方法在川酒品牌评价中的应用

A hesitant fuzzy linguistic PROMETHEE method and its application in Sichuan liquor brand evaluation

控制与决策. 2019, 34(12): 2727−2736 https://doi.org/10.13195/j.kzyjc.2018.0335

基于前景理论的风险型混合模糊多准则群决策

Risk mixed multi−criteria fuzzy group decision−making approach based on prospect theory

控制与决策. 2017, 32(7): 1279−1285 https://doi.org/10.13195/j.kzyjc.2016.1221

基于后悔理论的灰色随机多准则决策方法

Method for grey−stochastic multi−criteria decision−making based on regret theory

控制与决策. 2017, 32(6): 1069−1074 https://doi.org/10.13195/j.kzyjc.2016.0295

一种改进的多agent分布式联盟形成算法

An improved distributed coalition formation algorithm in MAS

控制与决策. 2017, 32(4): 632−636 https://doi.org/10.13195/j.kzyjc.2016.0252

基于后悔理论的具有期望水平的直觉语言多准则决策方法

Approach to intuitionistic linguistic multi−criteria decision making with aspiration levels on criteria based on regret theory

控制与决策. 2016, 31(9): 1638−1644 https://doi.org/10.13195/j.kzyjc.2015.0848

基于前景理论的区间直觉模糊双向投影决策方法

Bidirectional projection method with interval−valued intuitionistic fuzzy information based on prospect theory

控制与决策. 2016, 31(6): 1143−1147 https://doi.org/10.13195/j.kzyjc.2015.0590

基于对立搜索和混沌变异的磷虾觅食优化算法

An improved krill herd algorithm based on oppositional searching and chaos mutation

控制与决策. 2015(9): 1617−1622 https://doi.org/10.13195/j.kzyjc.2014.1024

基于粒计算的最简决策规则挖掘算法

Mining algorithm for concise decision rules based on granular computing

控制与决策. 2015(1): 143−148 https://doi.org/10.13195/j.kzyjc.2014.0126

文章编号: 1001-0920(2020)03-0637-08 DOI: 10.13195/j.kzyjc.2018.0796

基于蒙特卡洛Q值函数的多智能体决策方法

张 健[1], 潘耀宗[1], 杨海涛[1†], 孙 舒[2], 赵洪利[1]

(1. 中国人民解放军战略支援部队航天工程大学，北京 101416；2. 中国人民解放军63628部队，河北 三河 065201；3. 中国人民解放军63919部队，北京 100089)

摘 要: 多智能体决策问题是人工智能领域的研究热点. 与单智能体决策问题相比,多智能体决策的策略搜索空间更大. 分布式局部感知马尔可夫决策过程(Dec-POMDPs)建立了不确定环境下多智能体决策问题的通用模型,自提出以来受到很大关注,但是求解Dec-POMDPs问题计算复杂度高,内存占用大. 基于此,提出一种新的Q值函数表示——蒙特卡洛Q值函数($Q_{\rm MC}$),并从理论上证明$Q_{\rm MC}$是最优Q值函数Q^*的上界,能够保证启发式搜索到最优解;运用自适应抽样方法,平衡收敛准确性和求解时间的关系;结合启发式搜索的精确性和蒙特卡洛方法随机抽样的一般性,提出一种基于$Q_{\rm MC}$的蒙特卡洛聚类/扩展算法(CEMC),CEMC整合了Q值函数求解和策略搜索过程,避免保存所有值函数,只按需求解. 实验结果表明,CEMC在时间和内存占用上超过目前性能最好的使用紧凑Q值函数的启发式方法.

关键词: 多智能体决策；蒙特卡洛；值函数；马尔可夫决策

中图分类号: TP301.6 **文献标志码:** A

Multi-agent decision making using Monte Carlo Q-value function

ZHANG Jian[1], *PAN Yao-zong*[1], *YANG Hai-tao*[1†], *SUN Shu*[2], *ZHAO Hong-li*[1]

(1. Space Engineering University，PAL Strategic Support Force，Beijing 101416，China；2. The 63628 Army of PLA，Sanhe 065201，China；3. The 63919 Army of PLA，Beijing 100089，China)

Abstract: Multi-agent decision making problems are very popular in artificial intelligence. Compared with single agent decision making problems, multi-agent decision making problems have larger policy space. Decentralized partially observable Markov decision processes (Dec-POMDPs) are general models for multi-agent decision making under uncertainty, which have caught much attention among researchers. Solving Dec-POMDPs has high computational complexity and takes much memory. This article presents a new Q-value function representation — Monte Carlo Q-value function($Q_{\rm MC}$), which is proved to be the upper bound of Q^*. This guarantees that the optimal policy can be found. An adaptive sampling method is used to balance the precision of convergence and solving time. And an algorighm called clustering and expansion for Monte Carlo (CEMC) based on $Q_{\rm MC}$ is proposed, which combines the precision of heuristic search with the generality of Monte Carlo random sampling. This algorithm integrates Q-value function solving with policy search and calculates value functions as needed, which avoids the need to backup all Q-value functions. The experiments show that the proposed method outperforms the state-of-the-art heuristic methods, with the compact Q-value function.

Keywords: multi-agent decision making；Monte Carlo；Q-value function；Markov decision

0 引 言

人工智能(AI)的一个重要目标是发展能够与环境交互的具有自主决策行为能力的智能体. 这种智能体通过感知(图像传感器、自然语言识别处理等)外界环境并通过决策产生行为. 目前已有很多服务型机器人应用于实际生活中. 尽管很多场景中单个智能体足以满足需求,但是仍有很多情况需要多个智能体协作完成某一任务,如机器人足球比赛[1]、机器人分拣快递等. 这些任务需要多智能体协调各自的行动,以达成特定目标. 在这些领域中,由传感器性能导致的感知不确定性和由于控制的不稳定导致行为的不确定性都是一些重要的特征.

分布式局部感知马尔可夫决策过程(Dec-POMDPs)[2]是用来描述不确定环境下多智能体序列决策问题的常用模型. Dec-POMDPs有很多应用场合,如: 1) 数据路由问题[3],该问题假定多个路由器

收稿日期: 2018-06-11；**修回日期:** 2018-09-18.
责任编委: 张化光.
†通讯作者. E-mail: haitaoyang79@126.com.

协作转发数据包,在每一时间步决定转发给哪个路由器,以最小化转发时间;2) 分布式拥塞控制[4]与负载均衡[5],在该问题中每个智能体相当于一个处理单元,每个智能体只能获得自己队列空余信息,并在接收到任务时决定是自己处理还是交给别的单元;3) 星球漫游问题[6],该问题描述的是多个智能机器人在未知星球上协作完成实验;4) 传感器网络[7-10]问题,该问题设定多个智能传感器协作以实现探测范围最大化;5) 交通流量控制问题[11-12],该问题认为每个交通灯是一个智能体,交通灯需要根据随机变化的交通流量作出改变以适应当前情况. 这些问题的主要特点是部分感知和多个智能体协作,Dec-POMDPs为其提供了很好的数学框架. 博弈论也适合解决类似问题,扩展范式博弈和马尔可夫博弈是两个重要的模型,可以描述多智能体的序列决策过程. Dec-POMDPs是在博弈框架下的多智能体具有相同效用函数时的一个特例,是当前多智能体决策研究的主流.

近几年,很多研究成果都将关注点放在求解Dec-POMDPs的最优解上. 然而,求解Dec-POMDPs是一件非常困难的事. 主要原因在于每个智能体需要依据自己的局部信息生成全局最优的策略,同时还要考虑其他智能体可能的行动. 另一个原因是联合策略的空间随着智能体数量和观察呈指数级增长,随着规划步数呈双指数增长. Bernstein等[2]指出计算最优规划解是NEXP-完全问题,在最坏情况下,几步之内就会占用大量内存,求解时间变得无法接受. 尽管如此,研究者们仍然提出很多新方法,希望在求解时间和求解深度上有所提升.

Dec-POMDPs的求解方法一般可以分为精确求解和近似求解两大类. 近几年,精确求解方法主要分为两大阵营:自底向上的动态规划[13]和自顶向下的启发式搜索[14]. 动态规划从$t = h - 1$开始,为每一个智能体循环构建策略树,直到$t = 0$. Boularias等[15]指出,造成Dec-POMDPs难以求解的根源在于信念空间的高维度特性,因此提出了无损策略压缩(LPC)算法用来对策略空间进行压缩. 增量策略生成(IPG)算法[16]考虑状态的可达性,通过计算可达状态集合来缩小空间. 启发式搜索算法从$t = 0$开始构造策略树,直到$t = h - 1$. Szer等[14]首先将A*方法用于Dec-POMDPs模型中,提出多智能体的A*(MAA*)算法. Oliehoek等[17]对MAA*算法进行了扩展,并显性地将整个过程分为:1) 基于历史联合策略池的迭代与裁剪;2) 节点选择;3) 扩展. 在此基础上,Oliehoek等[18]引入增量聚类和扩展技术(ICE)加速启发式方法的求解.

由于精确求解寻找最优策略,限制了其在求解深度上的发展,而近似求解方法则牺牲最优性换取求解深度. 部分近似算法受精确算法的启发,能够找到高质量的解,但是不保证最优性. 基于信念点的动态规划算法(PBDP)不考虑整个信念空间,而计算可达信念状态点,避免了大量徒劳计算从而节省了时间. 内存受限动态规划算法(MBDP)[19]将自顶向下的搜索和自底向上的动态规划相结合构建策略树,在每一时间步为每一个智能体保存一定数量的策略树,并用组合启发式函数选择最有希望的子策略. 在此基础上的改进包括用分支界限[20]或部分备份取代完全备份[21]的基于信念点的方法、对相似观察分支进行合并[22]、使用约束满足问题(CSP)[23]的方法解决备份耗时的高效的策略生成方法PBPG[24]等.

由于近似算法无法得知近似最优解有多接近最优解,且没有一个标准在近似和求解深度中取得平衡,最重要的是近似算法不能从理论上保证最优性,因此本文主要考虑精确算法在求解Dec-POMDPs的应用. 但是,现有算法在求解深度上有限,求解时间较长. 不管是动态规划还是启发式搜索,联合策略的备份或启发式函数的求解都耗费了大量的内存,使得算法很难应用到小内存的智能机器人上.

为了解决上述问题,本文提出一种全新的Q值函数表示——蒙特卡洛Q值函数$Q_{\rm MC}$,并从理论上证明了$Q_{\rm MC}$是最优Q值函数Q^*的上界. 为了平衡Q值函数的紧凑程度与求解时间的关系,本文提出一种自适应抽样的方法,依据求解问题本身的结构特征设定抽样次数. 最后提出一种基于$Q_{\rm MC}$的启发式方法——CEMC. 该方法借鉴启发式搜索的思想,整合启发式函数求解和最优解搜索过程,“按需求解”启发式函数,避免事先求解启发式函数对内存的大量占用;对于Dec-POMDPs模型而言,目前Q^*的最优上界是$Q_{\rm BG}$. 实验表明,CEMC方法在时间性能和求解深度上超过了使用$Q_{\rm BG}$作为值函数的启发式搜索方法.

1 背景知识

本节将简要叙述Dec-POMDPs模型和MCTS的相关知识,关于两者的详细介绍请参考文献[25]和文献[26].

1.1 Dec-POMDPs模型

Bernstein等[2]首次提出Dec-POMDPs模型,并分析了求解的复杂度. 该模型可以定义为一个元组

$\langle I, S, A, T, O, \boldsymbol{O}, R, b^0, h\rangle$. $I = \{1, \ldots, n\}$是有限智能体集合,n是智能体数量;$S = \{s_0, \ldots, s_{|S|-1}\}$是有限状态集合,$s_0$是初始状态;$A = \times_{i=1}^{n} A_i$是联合动作集合,其中$\vec{a} = \langle a_1, \ldots, a_n\rangle \in A$,$A_i$是智能体$i$的有限动作集合,联合动作中的元素$a_i$属于智能体$i$;$T$是转移函数,其中$P(s'|s, \vec{a})$表示在状态$s$采取联合动作$\vec{a}$转移到$s'$的概率;$O = \times_{i=1}^{n} o_i$是联合观察集合,其中$\vec{o} = \langle o_1, \ldots, o_n\rangle \in O$,$O_i$是智能体$i$的有限观察集合,联合观察中的元素$o_i$属于智能体$i$;$\boldsymbol{O}$是观察函数,其中$P(\vec{o}|\vec{a}, s')$表示采取联合动作$\vec{a}$转移到状态$s'$后获得观察$\vec{o}$的概率;$R$是奖励函数,其中$R(s, \vec{a})$表示在状态$s$采取动作$\vec{a}$获得的立即奖励;$b^0$是初始状态分布;$h$是有限步数.

在每一时间步t,环境处于某一状态s,每一智能体通过传感器感知环境获得各自观察分量o_i^t,观察分量共同组成联合观察$\vec{o}^t$. 智能体根据获得的观察选择各自的动作分量a_i^t,动作分量共同组成联合动作$\vec{a}^t$,作用于环境使其根据转移函数由状态s转移到状态s'. 每个智能体依据奖励函数获得奖励r. 规划的目标是计算一个从历史$\vec{\theta}$映射到动作$\vec{a}$的联合策略,该策略能够最大化累积期望奖励$\mathrm{E}\Big(\sum_{i=1}^{h-1} R(s^t, \vec{a}^t)b^0\Big)$. 时间步$t$的历史$\vec{\theta}^t$表示为动作和观察的序列$\langle \vec{o}^t, \vec{a}^t\rangle$,其中$\vec{o}^t$和$\vec{a}^t$分别表示联合观察和联合动作. 智能体$i$的策略为映射$\pi_i : \vec{\theta}_i \to A_i$. 所有智能体的联合策略为$\pi = \langle \pi_1, \ldots, \pi_n\rangle$. 最优策略$\pi^*$的值定义为

$$V^t(\pi^*) = \sum_{\vec{\theta}^t} P(\vec{\theta}^t|b^0) Q^*(\vec{\theta}^t, \pi^*(\vec{\theta}^t)). \tag{1}$$

其中:$\pi^*(\vec{\theta}^t)$表示由π^*为$\vec{\theta}^t$指定的联合动作,$Q^*(\vec{\theta}^t, \vec{a})$表示为

$$Q^*(\vec{\theta}^t, \vec{a}) = R(\vec{\theta}^t, \vec{a}) + \sum_{\vec{o}^{t+1} \in \boldsymbol{O}} P(\vec{o}^{t+1}|\vec{\theta}^t, \vec{a}) Q^*(\vec{\theta}^{t+1}, \pi^*(\vec{\theta}^{t+1})). \tag{2}$$

智能体i在时间步t的局部策略表示为φ_i^t,定义为决策规则的序列

$$\varphi_i^t = (\delta_i^0, \delta_i^1, \ldots, \delta_i^{t-1}), \tag{3}$$

其中决策规则δ_i^t是智能体i在时间步t的动作-观察历史到动作的映射,有

$$\delta_i^t = \vec{\Theta}_i^t \to A_i. \tag{4}$$

枚举法求解Dec-POMDPs在最坏情况时将评估$|A_*|^{\frac{n(|A_*||O_*|)^h - 1}{|A_*||O_*| - 1}}$个联合策略的值,由此可见,联合策略的数量与步数$h$呈双指数关系.

需要特别指出的是,本文所讨论的模型是有限步数,无限步数的模型需要一个折扣因子$\gamma \in (0, 1]$,表示当前动作对未来的影响程度. 无限步数的Dec-POMDPs不在本文的讨论范围之内.

1.2　蒙特卡洛树搜索方法

蒙特卡洛树搜索(MCTS)以增量、非对称的方式扩展搜索树. 树中的每一个节点表示一个状态,连边表示执行的动作. 该方法循环执行选择、扩展、模拟和回传操作,直至达到一些预先设定的量,如时间或者内存[27]. 选择阶段依据树策略,在当前状态下循环选择一个动作,直到结束或者存在未被访问子节点的节点. 树策略需在选择目前最优节点和探索未知节点之间取得平衡,即平衡开发(exploitation)和探索(exploration)的关系;扩展阶段在有未知子节点的节点上增加一个或多个节点,这些节点表示在状态s下执行动作$\vec{a}$到达状态s';模拟阶段在新扩展的节点上依据模拟策略递归地选择一个动作,直到结束;回传依据最终结束的结果更新本次迭代过程中访问的节点的值. Browne等[28]全面综述了MCTS的基本理论、核心算法及在置信区间上界[29-30]、树策略[31-32]、模拟策略[33−35]及回传[36]方法上的改进.

UCT是MCTS算法中最著名的[26,37],它以置信区间上界UCB1[29]为树策略来选择动作,即

$$\bar{x}_i + c\sqrt{\frac{2\ln N}{n_i}}. \tag{5}$$

其中:$\bar{x}_i$是均值;N是父节点被访问总次数;n_i是节点被访问次数;$c > 0$是一常数,表示exploration的权重. 模拟策略一般是快速地产生下一动作的策略——随机策略. 随机策略的好处是可以快速地选择动作,以便能够迅速地评估局面,但是没有利用领域知识. 本文蒙特卡洛Q值函数的求解借用了文献[36]的树策略. 参数c根据所求解问题的树结构进行设置,如果问题的树分支较多,可以设置c稍大;如果是深度较大的树,可以设置c稍小.

2　近似Q值函数表示

最优策略可以从最优Q值函数Q^*中提取,并且Q^*可以引导启发式函数以最快的速度寻找最优解,但是求解Q^*的难度等同于求解Dec-POMDPs本身,因此Q^*一般只有理论上的意义. Oliehoek等在文献[17]中指出目前比较常见的Q值函数有Q_{MDP}[14]、Q_{POMDP}[14]、Q_{BG}[17],它们都是最优Q值函数Q^*的

上界,并有如下关系:

$$Q^* < Q_{\rm BG} < Q_{\rm POMDP} < Q_{\rm MDP}. \tag{6}$$

Oliehoek等指出,近似Q值函数越接近Q^*,即越紧凑,启发式搜索越快. $Q_{\rm BG}$是比其他两个更加紧凑的Q值函数,但是求解$Q_{\rm BG}$本身比较耗时,而且通过实验发现,对于联合动作和观察较多的基准问题,存储$Q_{\rm BG}$将占用大量内存,随着步数的增加,将造成内存耗尽.

2.1 蒙特卡洛Q值函数

本文结合蒙特卡洛随机抽样的一般性特征,提出一种Q值函数的全新表示——蒙特卡洛Q值函数$Q_{\rm MC}$. $Q_{\rm MC}$的求解可以形象化地表示为历史-动作-观察的树结构,如图1所示.

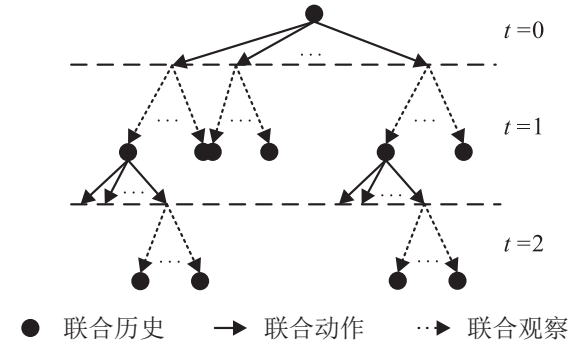


图1 蒙特卡洛Q值函数搜索树

在图1中,点表示联合历史,实线箭头表示在当前历史下执行的联合动作,虚线箭头表示在执行动作后获得的联合观察. 每一个节点维护一组信息: $N(\vec{\theta})$和$N(\vec{\theta},\vec{a})$分别表示访问节点$\vec{\theta}$的次数和在节点$\vec{\theta}$选择动作$\vec{a}$的次数.

$Q_{\rm MC}$在选择阶段使用UCB1为准则选择动作:

$$\vec{a} \leftarrow \mathop{\arg\max}_{\vec{a}\in A}\left(R+\hat{R}+c\sqrt{\frac{2\ln N(\vec{\theta})}{N(\vec{\theta},\vec{a})}}\right), \tag{7}$$

其中$R+\hat{R}$是立即奖励与未来奖励之和,可以认为是当前动作的平均奖励. 参数c可以根据求解问题的结构设定,但本文只是手动设定c值,并没有深入分析问题结构与c值的关系.

在扩展阶段,$Q_{\rm MC}$只扩展一个子节点(也可以选择扩展多个). 在模拟阶段,$Q_{\rm MC}$采用随机策略快速地选择动作直到$t=h-1$. 随机策略的优点是能够快速地选择动作进行模拟操作,以评估当前状态,这类似于计算机围棋里的快速走子策略. 回传阶段依据$t=h-1$时的立即奖励更新本次迭代中访问的节点. 因此,$Q_{\rm MC}$定义为

$$Q(\vec{\theta}^t,\vec{a}) = R(\vec{\theta}^t,\vec{a}) + \sum_{\vec{o}^{t+1}\in O} P(\vec{o}^{t+1}|\vec{\theta}^t,\vec{a})\times \max_{\rm MC} Q(\vec{\theta}^{t+1},{\rm MC}(\vec{o}^{t+1})), \tag{8}$$

其中MC是指蒙特卡洛Q值函数,它使用UCT方法返回最优动作MC$(\vec{o}^{t+1})$.

推论1 $Q_{\rm MC}$是最优Q值函数Q^*的上界.

证明 依据文献[17]的理论,Dec-POMDPs模型可以用k-步延迟通信系统表示. $Q_{\rm BG}$等价于1-步延迟通信系统. 如果$k=0$,k-$Q_{\rm BG}$退化为$Q_{\rm POMDP}$,可以表示为

$$Q(\langle\vec{\theta}^t,\phi\rangle,\vec{a}) = R(\vec{\theta}^t,\vec{a}) + \sum_{\vec{o}^{t+1}\in O} P(\vec{o}^{t+1}|\vec{\theta}^t,\vec{a})\times \max_{\vec{a}^{t+1}} Q(\vec{\theta}^{t+1},\vec{a}^{t+1}), \tag{9}$$

其中ϕ表示在$k=0$时的空策略,即0-步延时通信系统与$Q_{\rm POMDP}$等价. 比较式(8)和(9)可以看出,$Q_{\rm MC}$与0-步延时通信系统等价. $Q_{\rm POMDP}$是已证明的最优Q值函数的上界,所以$Q_{\rm MC}$是最优Q值函数Q^*的上界. □

从$Q_{\rm MC}$的数学表示可以看出,$Q_{\rm MC}$把Dec-POMDPs模型看作是一个拥有相同O、T、R但存在一个中心智能体来执行联合动作并获取联合观察的模型. 这种假设可以获得比基本Dec-POMDPs模型更多的信息. 因此,$Q_{\rm MC}$是Dec-POMDPs模型Q值函数的高估,可以确保启发式方法搜索到最优解.

2.2 自适应抽样

由于蒙特卡洛树搜索的任意时间特性,足够多的重复抽样可以使Q值函数收敛,但是也将耗费更长的时间. 然而,抽样次数过少显然会得到过于松散的Q值函数,甚至导致无法求出正确的策略. 所以本文根据所求解的问题自动设置抽样次数,即自适应抽样方法,以在收敛精度与求解时间上取得平衡.

本文中抽样次数K是联合动作数量和时间步的函数. K与动作、时间步的关系可以表示为

$$K = m\frac{|A|}{(t+1)^2}. \tag{10}$$

其中: m是与求解问题相关的常数,本文中根据经验设定; $|A|$是联合动作的数量; t是时间步.

由式(10)可以看出,K与时间步t的平方呈反比,与动作数呈正比. 也就是说,动作较多的问题需要较大的K来使Q值函数收敛. 同时,随着自上而下构造搜索树,抽样次数逐渐减小. 通过引入自适应抽样,可以动态调整算法中循环的次数,以使Q值函数达到收

敛的同时不浪费过多的抽样时间.

3 基于Q_{MC}的多智能体决策方法

Q_{BG}、Q_{POMDP}等Q值函数需要构造大小为$\vec{\theta}\times\vec{a}$的矩阵进行存储,以往实验表明大部分的$(\vec{\theta},\vec{a})$对都不可达. 为了避免备份所有$(\vec{\theta},\vec{a})$对,结合$Q_{\mathrm{MC}}$的任意时间特性,本文提出一种基于$Q_{\mathrm{MC}}$的多智能体决策方法(CEMC),该方法借鉴了ICE算法的启发式思想. 但与之不同的是,CEMC将启发式函数求解与策略搜索进行整合,不再显式地求解Q值函数,而是"按需求解",所以不必开辟空间存储所有$(\vec{\theta},\vec{a})$对,在策略求解过程中根据策略树的构造按需计算Q值函数. CEMC的伪代码如下所示.

输入: Dec-POMDP,空表L;

输出:最优联合策略π^*.

1) 初始化L

2) While $\neg\mathrm{empty}(L)$ do

3) $q\leftarrow\mathrm{Select}(L)$//选择最优节点

4) $B(\varphi^t)\leftarrow\mathrm{ClusterBG}()$//聚类

5) $\langle\beta^t,\hat{V}(\beta^t)\rangle\leftarrow B(\varphi^t).\mathrm{Solver}()$

6) $Q_{\mathrm{Expand}}\leftarrow\mathrm{ExpandIE}()$//节点增量扩展

7) if $\mathrm{depth}(q)=h-1$

8) $\langle\pi,v\rangle\leftarrow$ Compute then Policy and Value (Q_{Expand})//按需计算Q_{MC}

9) if $v>\underline{v}^{\mathrm{GMAA}}$ then

10) $\pi^*\leftarrow\pi,\underline{v}^{\mathrm{GMAA}}\leftarrow v$

11) $L.\mathrm{Prune}(\underline{v}^{\mathrm{GMAA}})$//策略池裁剪

12) end if

13) else

14) $L.\mathrm{Insert}(\{q'\in Q_{\mathrm{Expand}}|q',\hat{v}>\underline{v}^{\mathrm{GMAA}}\})$ //更新表

15) end if

16) end while

17) return π^*

CEMC算法使用$\hat{V}(\varphi^t)$作为启发式函数引导搜索算法,$\hat{V}(\varphi^t)$定义为

$$\hat{V}(\varphi^t)=V^{0\ldots t-1}(\varphi^t)+H^{0\ldots t-1}(\varphi^t). \tag{11}$$

其中:$\hat{V}(\varphi^t)$是局部策略φ在时间步t的启发值,$V^{0\ldots t-1}(\varphi^t)$是0到$t-1$的已获得的奖励,$H^{0\ldots t-1}(\varphi^t)$是用$Q_{\mathrm{MC}}$对未来奖励的估计. 如果$H^{0\ldots t-1}(\varphi^t)$是对真实值的高估,则该启发式函数会引导算法找到最优策略.

CEMC以增量方式扩展策略树,策略树中每个节点表示一个局部策略φ^t,连边表示联合动作. 同时,CEMC维护一个策略池,策略池中的元素即为策略树的节点. 策略池中节点定义为一个元组$\langle\varphi^t,\hat{v},h\rangle$. CEMC先扩展优先级最高的节点,节点优先级定义如下.

定义1　两个节点$q=\langle\varphi^t,\hat{v},h\rangle$和$q'=\langle\varphi^{t'},\hat{v}',h'\rangle$,其优先级$q<q'$,满足

$$\begin{cases}\hat{v}<\hat{v},\ \hat{v}\neq\hat{v}';\\ h<h',\ h\neq h'.\end{cases} \tag{12}$$

即,当两个节点启发值不相等时,启发值大的节点优先扩展;当启发值相等时,深度大的节点优先扩展.

4 实　验

本节从实验的角度验证算法的有效性,首先给出该领域的几个基准测试问题,实验统计求解时间、内存占用、最优策略值和随机策略值,并分析Q值函数的收敛情况,最后分析和讨论实验结果.

4.1 实验设置

本文的测试问题可以从http://masplan.org处下载,包括Dec-Tiger、Grid-Small、Box Pushing等. 如表1所示,所有的测试问题都有两个智能体,但是有不同的状态数、动作数和观察数. 表1中还列出了测试问题在不同步数时对应的策略数,可以看出,对于动作和观察较多的Box Pushing问题,在$t=5$时,其总策略数达到了惊人的10^{940}量级.

表1　基准问题

问题	参数				num of π for h		
	n	$\|S\|$	$\|A\|$	$\|O\|$	$H=2$	$H=5$	$H=10$
Dec-Tiger	2	2	3	2	6 561	3.34×10^{30}	1.39×10^{977}
Grid-Small	2	16	5	2	390 625	5.42×10^{44}	$3.09\times10^{1\,431}$
Box Pushing	2	100	4	5	3.34×10^{7}	5.23×10^{940}	$1.25\times10^{2\,939\,746}$

实验在Intel Core i5 CPU、6 GB内存、Ubuntu 16.04环境下进行. 代码以MADP工具箱[38]为基础实现,每次结果为运行100次取平均值. 将CEMC算法与目前性能最好的启发式算法(ICE,使用$Q_{\rm BG}$作为启发式函数)进行比较,分别统计计算Q值函数的时间t_Q、求解策略时间t_p、算法运行总时间t_o、策略价值V^*和随机策略价值V_r. 所有结果都是在同一环境下测试所得,因此时间上具有直接的可比性.

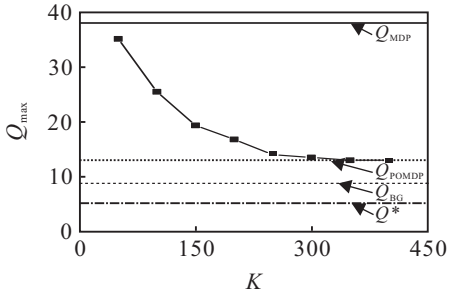
4.2 实验结果

表2给出了对3个基准问题的测试结果,“–”表示未知值,由超出时间限制(3 600 s)或者得出错误结果导致;“*”表示超出内存限制. 由于不需单独计算Q值函数,CEMC的$t_Q = 0$. 从表2的数据可以看出,CEMC比目前最好的启发式算法ICE在时间性能上有很大提高. 在Grid-Small和Box Pushing,$t = 5$时,ICE由于超出内存无法求出最优策略,这可以反映出CEMC能够求解比ICE步数更大的问题,内存使用要比ICE小.

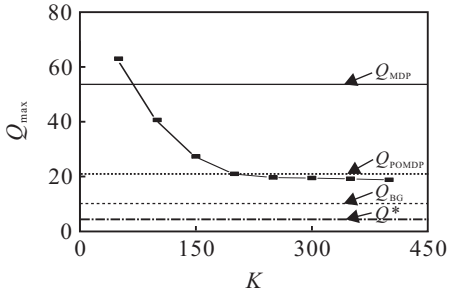
表 2 实验结果

测试问题	h	算法	t_Q	t_p	t_o	V^*	V_r
Dec-Tiger	2	ICE	<0.01	<0.01	0.27	−4.00	−93.05
		CEMC	0	<0.01	0.27	−4.00	−93.05
	3	ICE	0.06	0.02	0.43	5.19	−140.81
		CEMC	0	0.06	0.28	5.19	−140.81
	4	ICE	–	–	–	–	–
		CEMC	0	65.08	65.36	4.80	−186.67
Grid-Small	2	ICE	0.01	0.01	0.95	0.86	0.20
		CEMC	0	0.04	0.95	0.86	0.20
	3	ICE	3.88	0.03	4.84	1.37	0.38
		CEMC	0	0.91	1.81	1.37	0.38
	4	ICE	158.44	0.03	159.23	1.88	0.56
		CEMC	0	11.76	12.66	1.88	0.56
	5	ICE	*	*	*	*	*
		CEMC	0	50.23	51.33	2.53	–
Box Pushing	2	ICE	30.73	0.01	32.39	17.60	−0.01
		CEMC	0	0.54	1.81	17.60	−0.01
	3	ICE	492.58	21.76	540.00	66.08	−0.61
		CEMC	0	0.82	2.36	66.08	−0.61
	4	ICE	2 974.12	371.83	3 253.22	98.59	–
		CEMC	0	3.24	5.98	98.59	–
	5	ICE	*	*	*	*	*
		CEMC	0	21.49	30.79	107.72	–

为了验证$Q_{\rm MC}$的收敛性,本文统计了CEMC在求解测试问题时的最大Q值函数随抽样次数K的变化情况. 前文提到,足够多的抽样可以使近似Q值函数收敛到Q^*,但是将耗费更多时间求解. 图2显示了在$t = 3$和$t = 4$时,Dec-Tiger问题的最大Q值函数,它们都收敛于$Q_{\rm POMDP}$.



(a) Dec-Tiger for $h=3$



(b) Dec-Tiger for $h=4$

图 2 $Q_{\rm MC}$的收敛性

5 分析与讨论

本文提出的CEMC算法的两项重要指标(时间和内存占用)超过了目前最好的使用紧凑Q值函数$Q_{\rm BG}$的启发式算法ICE. CEMC良好的性能主要由于以下几点原因: 1) CEMC整合了Q值函数求解和最优策略搜索两大过程,不求解所有Q值函数. 这两个过程单独求解都是NEXP-完全问题,尽管ICE算法在求解Q值函数时使用了近似表示,但紧凑的Q值函数计算仍然需要很长时间; 2) CEMC“按需求解”在最优策略搜索过程中用到的Q值函数,不存储所有$(\vec{\theta},\vec{a})$对,节省了内存空间; 3) 计算$Q_{\rm MC}$时利用MCTS任意时间的特性,根据联合动作数量和时间步动态设定抽样次数K,平衡收敛精度与求解时间的关系.

本文选择了该领域比较有代表性的基准测试问题,Grid-Small具有较多的动作数,Box Pushing具有较多的状态数. 但是可以发现,对于同样步数的两个问题,求解Box Pushing要比Grid-Small时间短,这说明动作对求解复杂度的影响大于状态,这可以从总策略树$|A_*|^{\frac{n(|A_*||O_*|)^h-1}{|A_*||O_*|-1}}$看出,影响搜索的主要参数是$A$、$O$、$h$,且$h$与策略数是双指数关系. $Q_{\rm MC}$收敛于$Q_{\rm POMDP}$,尽管没有$Q_{\rm BG}$紧凑,但是从实验结果可以

看出稍稍放大的Q值函数获得了更好的性能,如同计算机围棋,需要快速评估当前局面,这就需要在精确与快速之间作平衡.

6 结 论

本文提出的CEMC算法结合了启发式搜索的精确性和蒙特卡洛树搜索随机抽样的一般性特征,整合了Q值函数求解和最优策略搜索,"按需求解"Q值函数,不存储所有$(\vec{\theta},\vec{a})$对,节省了内存空间,并在求解时间和求解步数上有所提升. 对于轻量的移动机器人,由于其资源有限(如携带能量较少,内存空间较少),先计算后存储的方式并不适用. 本文提出的多智能体的决策方法适用于多个轻量机器人在无法通信的场景中协作完成任务的情形. 在未来的工作中,将考虑蒙特卡洛方法中参数c的动态设定问题,以更好地适应问题结构,并考虑在有限通信的场景中求解最优策略的问题.

参考文献(References)

[1] RoboCup Federation. RoboCup Federation Official Website[EB/OL]. (2017-11-18)[2018-01-22]. http://www.robocup.org/.

[2] Bernstein D S, Givan R, Immerman N, et al. The complexity of decentralized control of markov decision processes[J]. Mathematics of Operations Research, 2002, 27(4): 819-840.

[3] Peshkin L M. Reinforcement learning by policy search[D]. Providence: Department of Computer Science, Brown University, 2001.

[4] Winstein K, Balakrishnan H. TCP Ex machina: Computer-generated congestion control[J]. Computer Communication Review, 2013, 43(4): 123-134.

[5] Ouyang Y, Teneketzis D. Balancing through signaling in decentralized routing[C]. 2014 IEEE 53rd Annual Conference on Decision and Control. Los Angeles: IEEE, 2015: 1675-1680.

[6] Zilberstein S, Washington R, Bernstein D S, et al. Decision-theoretic control of planetary rovers[C]. The International Seminar on Advances in Plan-Based Control of Robotic Agents. Heidelberg: Springer, 2001: 270-289.

[7] Nair R, Varakantham P, Tambe M, et al. Networked distributed POMDPs: A synergy of distributed constraint optimization and POMDPs[C]. Proceedings of the Nineteenth Intemational Joint Conference on Artificial Intelligence. Edinburgh: Professional Book Center, 2005: 1758-1760.

[8] Jain M, Taylor M, Tambe M, et al. DCOPs meet the real world: Exploring unknown reward matrices with applications to mobile sensor networks[C]. International Joint Conference on Artificial Intelligence. Pasadena, 2009: 181-186.

[9] Varakantham P, Marecki J, Yabu Y, et al. Letting loose a SPIDER on a network of POMDPs: Generating quality guaranteed policies[C]. International Joint Conference on Autonomous Agents and Multiagent Systems. Hawaii: ACM, 2007: 14-18.

[10] Kumar A, Zilberstein S. Constraint-based dynamic programming for decentralized POMDPs with structured interactions[C]. International Conference on Autonomous Agents and Multiagent Systems. Budapest: IFAAMAS, 2009: 561-568.

[11] Wu F, Zilberstein S, Jennings N R. Monte-Carlo expectation maximization for decentralized POMDPs[C]. International Joint Conference on Artificial Intelligence. Beijing, 2013: 397-403.

[12] Bazzan A L C, Oliveira D D, Silva B C D. Learning in groups of traffic signals[J]. Engineering Applications of Artificial Intelligence, 2010, 23(4): 560-568.

[13] Hansen E A, Bernstein D S, Zilberstein S. Dynamic programming for partially observable stochastic games[Z]. AAAI Workshop-Technical Report, 2004.

[14] Szer D, Charpillet F, Zilberstein S. MAA*: A heuristic search algorithm for solving decentralized POMDPs[C]. The 21st Conference on Uncertainty in Artificial Intelligence. Scotland: AUAI Press, 2005: 576-583.

[15] Boularias A, Chaib-Draa B. Exact dynamic programming for decentralized POMDPs with lossless policy compression[C]. The 18th International Conference on Automated Planning and Scheduling. Sydney: AUAI Press, 2008: 20-27.

[16] Amato C, Dibangoye J S, Zilberstein S. Incremental policy generation for finite-Horizon DEC-POMDPs[C]. Proceedings of the 19th International Conference on Automated Planning and Scheduling. Thessaloniki: IFAAMAS, 2009: 569-576.

[17] Oliehoek F A, Spaan M T, Vlassis N. Optimal and approximate Q-value functions for decentralized POMDPs[J]. Journal of Artificial Intelligence Research, 2008, 32(1): 289-353.

[18] Oliehoek F A, Spaan M T, Amato C, et al. Incremental clustering and expansion for faster optimal planning in DEC-POMDPs[J]. Journal of Artificial Intelligence Research, 2013, 46(1): 449-509.

[19] Seuken S, Zilberstein S. Memory-bounded dynamic programming for DEC-POMDPs[C]. International Joint Conference on Artificial Intelligence. Hyderabad: AAAI Press, 2007: 2009-2015.

[20] Dibangoye J S, Mouaddib A I, Chai-Draa B. Point-based incremental pruning heuristic for solving

finite-horizon DEC-POMDPs[C]. The 8th International Joint Conference on Autonomous Agents and Multiagent Systems. Budapest: IFAAMAS, 2009: 569-576.

[21] Seuken S, Zilberstein S. Improved memory-bounded dynamic programming for decentralized POMDPs[C]. Proceedings of the 23rd Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-07). Corvallis: AUAI Press, 2007: 344-351.

[22] Carlin A, Zilberstein S. Value-based observation compression for DEC-POMDPs[C]. International Joint Conference on Autonomous Agents and Multiagent Systems. Estoril: IFAAMAS, 2008: 501-508.

[23] Kumar A, Zilberstein S. Point-based backup for decentralized POMDPs: Complexity and New Algorithms[C]. International Conference on Autonomous Agents and Multiagent Systems. Toronto: IFAAMAS, 2010: 1315-1322.

[24] Wu F, Zilberstein S, Chen X. Point-based policy generation for decentralized POMDPs[C]. International Conference on Autonomous Agents and Multiagent Systems. Toronto: IFAAMAS, 2010: 1307-1314.

[25] Oliehoek F A, Amato C. A concise introduction to decentralized POMDPs[M]. New York: Springer, 2016: 14-17.

[26] Kocsis L, Szepesvári C. Bandit based monte-carlo planning[C]. The 17th European Conference on Machine Learning. Berlin: Springer, 2006: 282-293.

[27] Chaslot G, Bakkes S, Szita I, et al. Monte-carlo tree search: A new framework for game AI[C]. Proceedings of the 14th Artificial Intelligence and Interactive Digital Entertainment Conference. Stanford: The AAAI Press, 2008: 216-217.

[28] Browne C B, Powley E, Whitehouse D, et al. A survey of monte carlo tree search methods[J]. IEEE Transactions on Computational Intelligence & Ai in Games, 2012, 4(1): 1-43.

[29] Auer P, Cesa-Bianchi N, Fischer P. Finite-time analysis of the multiarmed bandit problem[J]. Machine Learning, 2002, 47(2/3): 235-256.

[30] Tesauro G, Rajan V T, Segal R. Bayesian inference in monte-carlo tree search[C]. Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence. Catalina Island: AVAI Press, 2010: 580-588.

[31] Chaslot G M J-B, Winands M H M, van den Herik H J, et al. Progressive strategies for monte-carlo tree search[J]. New Mathematics & Natural Computation, 2008, 4(3): 343-357.

[32] Gelly S, Wang Y. Exploration exploitation in go: UCT for monte-carlo go[C]. NIPS: Neural Information Processing Systems Conference on-line Trading of Exploration and Exploitation Workshop. Vancouver: Curran Associates, 2006: 1-8.

[33] Silver D, Tesauro G. Monte-carlo simulation balancing[C]. International Conference on Machine Learning. Montreal: ACM, 2009: 945-952.

[34] Hoock J B, Rimmel A, Teytaud F, et al. Intelligent agents for the game of go[J]. IEEE Computational Intelligence Magazine, 2010, 5(4): 28-42.

[35] Rimmel A, Teytaud F. Multiple overlapping tiles for contextual monte carlo tree search[C]. Applications of Evolutionary Computation. Torino: Springer, 2010: 201-210.

[36] Xie F, Liu Z. Backpropagation modification in monte-carlo game tree search[C]. International Symposium on Intelligent Information Technology Application. Shanghai: IEEE, 2009: 125-128.

[37] Kocsis L, Szepesvári C, Willemson J. Improved Monte-Carlo search[R]. Estonia: University of Tartu, 2006.

[38] Oliehoek F A, Spaan M T J, Terwijn B, et al. The MADP toolbox: An open source library for planning and learning in (multi-) agent systems[J]. Journal of Machine Learning Research, 2017, 18: 1-5.

作者简介

张健(1989–), 男, 博士, 从事多智能体决策的研究, E-mail:zjconquer@126.com;

潘耀宗(1984–), 男, 博士生, 从事智能规划的研究, E-mail: panyaozong1284@163.com;

杨海涛(1979–), 男, 副教授, 博士, 从事作战仿真系统的研究, E-mail: haitaoyang79@126.com;

孙舒(1982–), 女, 工程师, 博士, 从事航天搜救的研究, E-mail: sunshu_susan@163.com;

赵洪利(1964–), 男, 教授, 博士生导师, 从事作战筹划等研究, E-mail: zhlspace@sina.cn.

(责任编辑：齐 霁)