基于可分解MDP模型的MAS协作策略优化及分布执行

王晓伶 慕德俊 刘哲元 袁 源

(西北工业大学自动化学院 西安 710072)

摘 要 不确定环境下MAS生成协作策略的复杂度关系到协作任务能否成功实现。为降低马尔可夫决策模型生成MAS协作策略的复杂度,减少协作通信量,改进了可分解MDP模型生成策略树的方法。利用Bayesian网络中agent状态之间存在的条件独立性与上下文独立性,分解并优化SPI算法生成的策略树,使得MAS中处于独立状态的agent可以分布独立运行,只有在需要同其他agent协商时才进行通信。通信时采用端对端的方式,agent不仅知道协商内容、协商时机,而且知道协作的目标。实验表明,采用该协作策略MAS在完成协作任务获得目标奖励的同时可以有效降低通信量。

关键词 多智能体系统,可分解马尔可夫决策过程,贝叶斯网络,上下文独立,条件独立

中图法分类号 TP393 **文献标识码** A

Optimization and Distributed Execution of MAS Cooperative Strategy Based on Factored Dec-MDP

WANG Xiao-ling MU De-jun LIU Zhe-yuan YUAN Yuan

(School of Automation, Northwestern Polytechnical University, Xi'an 710072, China)

Abstract The complexity of MAS cooperation strategy in uncertain environment determines the success of cooperation task. In order to reduce the complexity created by factored MDP model and the cooperation traffic, the method of creating strategy tree by the model was improved. Using the context-specific and conditional independence existing among the agent states in Bayesian network, the tree created by SPI algorithm was decomposed and optimized. This makes the independent agents in MAS running independently, and only communicating with each other when cooperation is needed, where the Peer-to-Peer method is applied. Simulation indicates that MAS applying the strategy not only accomplishes the task and gains the reward, but effectively reduces traffic simultaneously.

Keywords MAS, Factored Dec-MDP, Bayesian network, Context-specific independence, Conditional independence

1 引言

MAS如何在不确定环境下实现agent之间的协商协作是其实现中的一个难点,直接关系到系统任务能否顺利实现。在MAS执行中,处于协作的agent不仅要观察自身的状态,同时要推断合作者的状态、动作。近年来利用马尔可夫决策论对MAS建模已经成为MAS研究的一个重要方向,Dec-MDP(Decentralized Markov Decision Process)作为马尔可夫决策过程的一种扩充应用于在全部可观察(决策者可精确观测MAS系统状态)条件下建模MAS。但是Dec-MDP生成协作策略比较复杂,即使只有两个agent进行协作,其时间复杂度为NEXP-complete[1]。一种改进Dec-MDP高计算复杂度的方法是将Dec-MDP观察域分类。有一类Dec-MDP称为传输独立Dec-MDP[2,3],它的系统状态可以被分割为独立agent的状态,这些agent状态仅和自身的活动相关,独立于系统联合活动,传输独立Dec-MDP模型生成策略的复杂度为NP-complete。虽然这种方法大大降低了复杂度,但并不是所有的MAS系统观察域都是传输独立的。另外一种方法是使用可分解Dec-MDP模型[4],这种方法充分利用系统状态变量中存在的条件独立关系、上下文独立关系,简化生成协作策略的复杂度,复杂度可以减少为P-complete。但是这种方法在执行过程中不考虑通信的代价,即执行过程中协商双方需要随时进行协商通信。文献[3]提出了一种优化可分解Dec-MDP模型生成策略的方法,该方法分解、优化生成的策略树,使得MAS可以分布执行策略树,并且只有在agent需要协商时才进行通信,在一定程度上减少了通信的花销。但是这种方法在对单个agent生成策略时,没有优化相应的策略子树,在执行时会对系统产生通信冗余。

为减少文献[3]中冗余的通信量,进一步优化协作策略,本文利用Bayesian网络中的d-分离、上下文分离对分解后的策略子树进行优化,进一步提取子树中存在的条件独立性和上下文独立性关系,去除子树中冗余的节点状态特征,在通信中预测可能用到的协作agent状态、动作信息,从而减少协作过程的通信量。实验表明,经过优化后的策略可以分布执行

到稿日期:2008-05-05 本文受国防基础科研项目(C2720061361),教育部博士点基金(20020699026)资助。

王晓伶(1980—),男,博士生,主要研究方向为网络信息安全,E-mail:wang2004mail@gmail.com;**慕德俊**(1963—),男,博士生导师,主要研究方向为网络信息安全、并行计算;**刘哲元**(1982—),男,硕士生,主要研究方向为网络信息安全;**袁 源**(1979—),男,博士生,主要研究方向为网络信息安全。

并且能有效地减少通信代价。

本文第 2 节详细描述构建本文 MAS 可分解 MDP 模型的组织结构；第 3 节叙述策路树的优化、分解、分布执行的方法；证明算法的有效性的实验将在第 4 节介绍；最后是结论和未来的工作。

2 可分解 Dec-MDP 模型描述

可分解 Dec-MDP 模型（FDec-MDP）是马尔可夫决策过程（MDP）处理 MAS 协商协作任务模型的一种扩展，用来处理 MAS 中协作 agent 的联合动作、联合观察结果以及最终生成任务协作策略。

Dec-MDP 模型结构表示为$\langle N,S,X,A,P,R\rangle$，其中：

N 表示 agent 数目；

S 表示状态的有限集合；

$X=\{X_1,\cdots,X_n\}$，表示 agent 状态的向量空间，状态特征 X_i 对应于状态集合 S 中状态 S_i，特征可以是布尔型或者其他数据类型；

A 表示可能的联合动作集合；

P 定义状态变迁概率函数，表示当给定一个动作时由一个状态变迁到另一个状态的可能性。且对于 $\forall z\in Z, Z\subset X$，变迁概率函数为 z 中各状态属性变迁概率函数的连乘积：$P(z|X_i,a)=\prod_{i=1}^{|Z|}P(z_i|X_i,a)$；

$R:X\times A\to\mathcal{R}$，表示奖励函数。

在 FDec-MDP 模型中，状态特征和联合动作之间的关系可以用 Bayesian 网络进行描述。图 1 表示的是 t 时刻与 $t+1$ 时刻模型中状态变量之间的关系。$t+1$ 时刻模型的状态特征依赖于其父状态 t 时刻的特征值，条件独立于其他变量特征值，Bayesian 网络中变量之间的这种条件依赖关系用条件概率表表示。但是在多数情况下，变量只是依赖一些特定的父变量特征值，而不是全部的父特征值，称为上下文独立关系[3,5]，使用条件概率树描述。如图 1 树型结构所示，状态 Y 在 X 取值为 0 时，上下文独立于 Z。

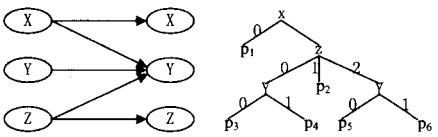

图 1 Bayesian 网络中的状态条件概率树

3 策略树分解执行

文献[4]提出了 SPI（Structured Policy Iteration）算法，用来在 Dec-MDP 模型中生成 MAS 协作策略树，其中策略树的叶子节点表示联合动作，非叶子节点表示 agent 状态。SPI 算法生成的策略树在 MAS 中是集中联合执行的，即使处于可独立执行状态，agent 也需等待其他 agent 状态。为使策略树在 MAS 中分布执行，单个 agent 需要一个可以匹配当前状态特征的执行策略。本节介绍一种将 SPI 生成的策略树转换为使各 agent 可分布执行策略树的方法。

3.1 将联合执行策略树转换为分布执行策略树

定义 1（动作独立） 如果在策略树叶子节点动作集合中存在一个动作，该动作同其他动作联合执行，且是多个联合动作的交集，在遍历该子树节点时会被同一个 agent 执行。例如联合动作集$\{ax,ay,bz\}$中，动作$\{a\}$独立；而在动作集$\{ax,by,cz\}$中没有独立的动作。

定义 2（交叉简化操作）[3] 用来合并策略树分支中动作集合交集非空的叶子节点或者子树。在策略树非叶子节点中递归调用该操作，如果节点中所有的孩子都是叶子节点，并且节点动作集合的交集非空，则该节点是冗余的，节点被一个包含交集的新叶子节点所代替；如果孩子中仅有一个是非叶子节点，该节点冗余，且如果该节点子树中包含于叶子节点动作集合的交集，则该节点被子树代替。算法如表 1 所示。

表 1 交叉简化操作算法

```
InterSectOpertion(policy)
  input:policy//树型结构的策略树
  output:policy//经过简化的策略树
1. 如果 policy 是叶子节点，则返回 policy；
2. 在 policy 孩子节点中递归调用 InterSectOpertion(policy)；
3. 如果 policy 所有的孩子都是叶子节点，则返回叶子节点动作交集(⋂i
   policy.childi.action)，并将该节点替换为动作交集；
4. 如果 policy 中只有一个孩子(childNL)不是叶子节点，则：
a)取得所有孩子中叶子节点动作集交集，记为 AL
b)取得 childNL 动作集交集，记为 ANL；
c)如果 ANL⊂AL，则返回 childNL
5. 返回 policy；
```

表 2 是将 SPI 生成的策略树转化为 agent 可分布执行的策略树的方法。该方法在 SPI 生成策略树基础上，对 MAS 中每个 agent 重组策略树，使 agent 分布执行。方法中第三步，使用预定义动作序列，目的是为了避免 agent 在分布执行中出现不协作或互相死等待的情况[3]。

表 2 分布策略树转化方法

1. 对每个 agent 重组策略树，使 agent 状态处于策略树根节点；
2. 对于策略树中叶子节点，找出所有独立动作；
3. 使用预定义动作序列解除联合动作之间的关系；
4. 将联合动作分解为个体动作；
5. 使用交叉简化操作简化策略树。

3.2 策略子树的优化

Bayesian 网络中策略树执行的复杂性取决于 Bayesian 网络分解后的对应的条件概率树大小，即由概率树的属性变量大小及其取值状态决定。在上一节中联合策略树经过转换后，对每个 agent 生成一个单独的策略树，在策略树中许多的节点状态之间存在着上下文独立性。可以利用这些节点的独立关系进一步优化每一个单独的子树。

定义 3（d-分离） X,Y,Z 是 Bayesian 网中有向非循环图 T 中三个互不相交的节点集，E d-分离 X 和 Y，记作 $I(X,Y|Z)_T$，当且仅当 X 中任一节点和 Y 中任一节点间的任一路径被节点集 Z 阻塞[5]。

定义 4 给定 Bayesian 网 T 和上下文 c。如果 $Ic(X,Y|\pi x\cap c)$，则称从 Y 到 X 的有向弧是无意义的。在上下文 c 条件下，删除 Bayesian 网 T 中的无意义弧得到的结果网络定义为 $T(c)$。在 $T(c)$ 中，如果 X 和 Y 被 $Z\cup C$ d-分离，则称上下文 c 及变量表 Z，X 和 Y 被上下文分离[5]。

$Ic(X,Y|\pi x\cap c)$ 是局部的上下文独立关系，可以从 X 节点的条件概率表中得出；使用局部的 Ic 删除无意义弧把 Bayesian 网 T 简化成 $T(c)$；在 $T(c)$ 利用图形判别 d-分离就

可以判断任意两个变量集是否被上下文分离。d-分离提供了一种判断全局任意两个变量集是否存在条件独立关系的方法，同样 $Ic(X,Y|\pi x \cap c)$ 也提供了判别两个变量集是否存在上下文独立的方法。利用这两个判别方法，提取策略子树中的条件独立性和上下文独立性，优化策略子树，形成最终的子策略树，如图 2 所示。树中根节点表示属于 agent i 的当前状态特征，非叶子状态表示协作 agent 的特征值。

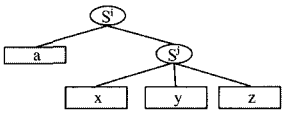

图 2　agent i 的分布执行策略树

3.3　策略的分布执行

对于单个 agent i 而言，经过分解、优化后的策略树是一个以自身状态为根节点的策略树，如图 2 所示，S_i，S_j 分别为 agent i，j 的观察状态。agent i 遍历策略树，根据自身状态选择分支。如果选择的分支是叶子节点，则执行叶子节点中的动作 a；当遍历到与其协作的 agent j 状态节点时，agent i 需要同 agent j 进行协商通信，以了解其状态，并据此选择要执行的动作。分布执行的递归算法如表 3 所示。

在实现 agent 协商通信时，以往的 Dec-MDP 模型使用的方法是：一个 agent 发起协商通信，所有的协商者广播它们所有包括历史结果的观察结果，采用这种方法协作 agent 需要辨别信息的有用性、必要性。本文采取的是端到端的协商方式，agent 仅向当前处于策略树中被遍历且状态特征未知的 agent 发起协商通信。采用这种方式，agent 不仅知道发起通信的时机、通信的内容，而且知道需要通信的对象。另外，为了避免出现 agent 就某协商信息的各个发起多次通信，agent 搜索处于策路树中需要通信的节点的所有子树的所有未知特征信息，并将之发送到协商 agent，协商 agent 求解后返回所需要信息特征值。

表 3　agent 分布执行策略的递归算法

IndividualExecute(policy, state, tm_state)
Input: policy; //图 2 中经过简化后的策略树
　　state; //agent 当前状态集合
　　tm_state; //已知的协作 agent 状态值集合
Output: action; //返回一个执行动作
define: index; //表示节点的第几个子树
1. 如果 policy 是叶子节点，则直接返回 policy 指向的动作；
2. 如果 policy 遍历的当前节点是非叶子节点，且 policy. value 包含于 state，则：
　a) index = state[policy. value];
　b) 返回 IndividualExecute
　　(policy. $child_{index}$, state, tm_state);
3. 如果 tm_state 为空，则：
　a) 找出所有 state 中需要同其他 agent 协商的状态特征 value；
　b) 同其他 agent 协商，并取得这些值；
　c) 创建 tm_state 值集合；
4. index = tm_state. value[policy. value];
5. 返回 IndividualExecute
　(policy. $child_{index}$, state, tm_state)。

4　实验与评估

本文实验是在 starlogo 中模拟两个海龟 agent 向目标坐标 T 移动，当到达目标时 agent 向对方发送到达信号 Signal。agent 动作集为{North, South, West, East, Stop, Signal}；agent 动作成功的可能性 P 为 90%；当双方 agent 成功发送 Signal 得到奖励 R 为 +20。反之，如果双方协作失败，则予以 −20 的惩罚；agent 每次移动消耗 1 的代价；在该试验中 agent 观察自己坐标 (x,y)，相互之间不知道对方的位置，只有在到达指定目标时才发送 Signal，之前 agent 则在上下文独立的环境中独立运行。环境如图 3 所示：标注中 A1，A2 分别表示 agent 1，2 初始位置，T 表示目标位置。

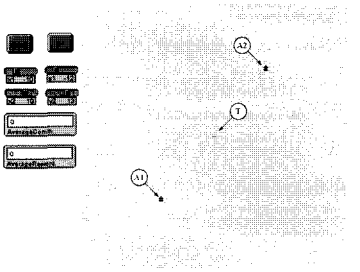

图 3　试验环境示例

试验中分别采用可自由通信的联合执行策略（表 4 中简称 JOINT-POLICY）、文献[3]中采用的 Dec-MDP 生成的策略、本文采用 FDec-MDP 模型生成的策略在上述试验环境中各运行 200 次。

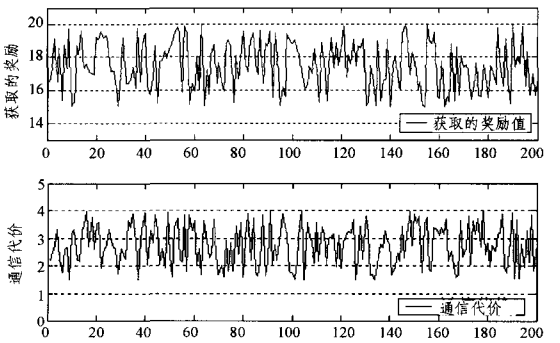

图 4　执行本文策略取得的奖励与通信代价

表 4　三种不同策略执行结果比较

	平均奖励(σ^2)	平均通信量(σ^2)
JOINT-POLICY	17.384(1.062)	7.032(1.059)
Dec-MDP	16.378(1.142)	3.332(1.042)
FDec-MDP	17.384(1.062)	2.253(1.053)

图 4 是采用本文经过优化后的策略树 MAS 执行取得的奖励及消耗的通信量。表 4 是分别采用 3 种不同策略执行取得的奖励值及通信量。从表 4 可以看出，本文策略取得的奖励与可自由通信的联合执行策略相同，优于 Dec-MDP 模型生成的策略，而且通信中需要的通信量低于前两者所需通信量。本文实验环境中 agent 是在 50×50 的矩阵空间中活动，agent 只有在到达目标坐标时才相互通信，之前则是各自独立运行，系统中存在大量的独立状态。这样，当 agent 在更大的矩阵空间中运行时，agent 运行过程中存在的独立状态将会更多，系统也将会节省更多的通信量。

结束语　本文利用 Bayesian 网络中 agent 状态之间的条件独立性、上下文独立性，分解、优化 SPI 生成的策略树，使得 MAS 中处于独立状态的 agent 可以分布独立运行，只有在需要同其他 agent 协商时才进行通信。agent 不仅知道协商内

容、协商时机，而且知道协作的目标，所以可以在通信时采取端对端的方式。试验表明，采用该方法生成的协作策略在完成协作任务获得目标奖励的同时可以有效降低通信量。

本文中 MAS 协作策略是在全部可观察的马尔可夫决策过程(Dec-MDP)模型中生成的，Dec-MDP 作为 Dec-POMDP 的一种特例。下一步的工作就如何将本文中的方法应用到 Dec-POMDP 模型中展开研究。

参 考 文 献

[1] Bernstein D S, Givan R, Immerman N, et al. The complexity of centralized control of Markov decision processes[J]. Mathematics of Operations Research, 2002

[2] Goldman C V, Zilberstein S. Decentralized control of cooperative systems: Categorization and complexity analysis[J]. Journal of AI Research, 2004

[3] Roth M, Simmons R, Veloso M. Exploiting Factored Representations for Decentralized[J]// AAMAS07. 2007: 469- 475

[4] Boutilier C, Dearden R, Goldszmidt M. Stochastic dynamic programming with factored representations[J]. Artificial Intelligence, 2000

[5] 王飞，刘大有. Bayesian 网中的独立关系[J]. 计算机科学，2001(12)：33-36

[6] 张新良，石纯一. M_POMDP 模型及其划分求解算法[J]. 清华大学学报：自然科学版，2005(10)：30-36

[7] 杨善林，胡小建. 复杂决策任务的建模与求解方法[M]. 北京：科学出版社，2007

[8] 方美琪，张树人. 复杂系统建模与仿真[M]. 北京：中国人民大学出版社，2005

(上接第 4 页)

了计算学科中的抽象、理论和设计 3 个概念作为学科最原始的概念，以学科中的“认知从感性认识(抽象)到理性认识(理论)，再由理性认识(理论)回到实践(设计)”作为惟一原始命题，构建了计算学科认知领域的理论体系，完成了计算学科认知模型框架的构建。

计算学科认知模型框架确定后，还应当有实质性的内容来填充。内容的充实，有两项工作要做：第一项是对学科概念的划分，也即是，将学科各个主领域原有的基本概念(集合)进行划分，采用集合论中等价类的划分原则，将集合划分为抽象、理论和设计 3 个子集；第二项是对 3 个子集的相互关系进行分析，以了解和掌握学科的发展规律。第一项任务《计算作为一门学科》报告做了，第二项任务文献[16-19]做了。

计算机方法论通过典型实例有序的引出学科的科学问题；它从一般方法论、计算机方法论和实例 3 个层面介绍学科的抽象、理论和设计；它从程序员的角度，以计算机语言的发展为主线，将自然语言、形式语言、图灵机、冯·诺依曼计算机，以及程序等内容有机地联系在一起，讲透程序与机器的关系，为学生深入学习和理解计算机系统打下基础；它介绍学科核心概念的来源，以及学科普遍采用的数学方法、形式化方法，以及系统化方法；它让学生了解社会与职业，为未来的职业生涯作准备；它介绍学科有争论的问题并对学科的未来进行展望。

计算机方法论中最原始的概念为：抽象、理论和设计。这 3 个概念与计算思维最基本的概念(抽象与自动化)反映的都是计算最根本的问题：什么能被有效地的自动进行。从教学的角度来说，计算机方法论中的这 3 个概念更易为人们所掌握。从研究的角度来说，计算机方法论遵循一般方法论的研究范式，更易展开研究工作。然而，对于具有丰富经验的计算机科学家来说，由于没有方法论研究中范式的约束，撰写有关计算思维方面的文章反到可能是他们的长处。

尽管计算思维与计算机方法论有着各自的研究内容与特色，但是，显而易见，它们的互补性很强，可以相互促进。比如，计算机方法论可以对计算思维研究方面取得的成果进行再研究和吸收，最终丰富计算机方法论的内容；反过来，计算思维能力的培养也可以通过计算机方法论的学习得到更好的提高。

参 考 文 献

[1] Wing J M. Computational Thinking. Communications of the ACM, 2006, 49(3)

[2] Wing J M. Computational Thinking and Thinking about Computing[EB/OL]. 2008. http://www.cs.cmu.edu/~wing/publications/Wing08a.pdf

[3] 周以真. 计算思维. 中国计算机学会通讯，2007，3(11)

[4] 王飞跃. 从计算思维到计算文化. 中国计算机学会通讯. 2007，3(11)

[5] http://www.cs4f.org/

[6] CS2001 Interim Review(draft). http://wiki.acm.org/cs2001/index.php? title=Main_Page，2008

[7] Philips P. Comptional Thinking: A problem-solving tool for every classroom[EB/OL]，2008. http://www.csta.acm.org/Resources/sub/ResourceFiles/ComputationalThinking.pdf

[8] Bundy A. Computational Thinking is Pervasive. Journal of Scientific and Practical Computing, Noted Reviews，1(2)

[9] BCS. The science of thinking: Europe's next policy challenge [EB/OL]. 2008. http://www.sciencebusiness.net/documents/thinking.pdf

[10] http://www.nsf.gov/crssprgm/cdi/

[11] Denning P J, et al. Computing as a discipline. Communications of the ACM，1989，32(1)

[12] 董荣胜. 向学术界推荐一个认知计算学科的工具——计算机科学与技术方法论(大会报告). 上海：新世纪计算机教育及 CC2001 研讨会，2001. 7

[13] ACM/IEEE-Curriculum 2001 Task Force. Computing Curricula 2001. Computer Science. IEEE Computer Society Press and ACM Press, 2001

[14] 中国计算机科学与技术教程 2002 研究组. 中国计算机科学与技术学科教程 2002. 北京：清华大学出版社，2002

[15] 全国高等学校计算机教育研究会. 全国“计算机科学与技术方法论”专题学术研讨会论文集. 计算机科学，2003，30(6，专辑)

[16] 董荣胜. 计算教育哲学初探[J]. 计算机科学，2000，27(1)

[17] 董荣胜，古天龙，等. 计算机科学与技术方法论[J]. 计算机科学，2002，29(1)

[18] 董荣胜，古天龙. 计算机科学与技术方法论[M]. 人民邮电出版社，2002(9)

[19] 董荣胜. 计算机科学导论——思想与方法[M]. 高等教育出版社，2007(9)

[20] 郑毓信，肖柏荣，熊萍. 数学思维与数学方法论. 四川教育出版社，2001