广东工业大学

硕士学位论文

基于马氏决策理论的智能体决策问题研究

姓名：郭靖

申请学位级别：硕士

专业：控制理论与控制工程

指导教师：陈玮

20120531

摘 要

马氏决策理论是智能体(agent)决策研究中有效的理论。马尔科夫决策过程(MDP)是马氏决策理论中最基础的一种模型，通常用来描述和解决大规模不确定性环境下智能体决策的问题。部分可观马尔科夫决策过程(POMDP)是 MDP 随着现实问题的扩展，当智能体决策过程中无法获得全局的信息的时候，POMDP 就是能够为决策过程提供可靠的模型和求解方法。随着人工智能研究的发展，越来越多的研究者开始考虑将多个智能体作为一个整体，也就是多智能体系统(Multi-agent System, MAS)，并开展对多智能体系统决策的研究，而分布式部分可观马尔科夫决策过程(DEC-POMDP)正是为解决 MAS 决策而提出的新模型。

本文首先介绍马氏决策理论中重要的三个模型 MDP、POMDP 和 DEC-POMDP 以及相应的求解算法，然后结合机器人足球 2D 仿真比赛中球员决策的问题，通过分析利用相关的模型和算法来提高球员决策的性能。本文主要工作可以分为以下三个方面：

首先，通过分析机器人足球 2D 仿真比赛中球员进攻决策存在的问题，发现球员在持球状态下决策的不足，然后利用 MDP 为球员持球状态下的进攻决策进行建模，同时提出值函数分解迭代的方法求解最佳进攻策略，通过实验数据证明利用本模型和相关求解算法能够让球队的进攻性能得到有效的提高。

其次，为提高球队守门员的表现，本文分析了守门员决策必须考虑决策实时性和信息不完整性，提出用 POMDP 为守门员决策进行建模，以提高守门员在紧急情况下及时有效防止对方进攻的表现。在求解过程中，为保证算法的实时性，文中提出基于了临界状态的求解方法，并将此方法应用到基于 POMDP 模型的守门员决策中。一系列检测守门员效果的实验数据表明，通过 POMDP 模型建模并求解后的守门员决策性能得到了较好的提升。

最后，我们展开了多智能体决策的研究。马氏决策过程中为解决多智能体系统决策，提出了 DEC-POMDP 模型，但是其相应的求解算法仍存在不足，尤其是目前大部分算法只能解决部分小规模问题，无法顺利地应用于机器人足球 2D 仿真比赛这种大规模多智能体系统的决策中。文中首先对多智能体系统决策的 DEC-POMDP 及相关算法进行分析，然后利用 MADP 工具箱对 DEC-POMDP 模型

的一系列标准测试问题进行测试和分析，阐述了 DEC-POMDP 离线规划求解过程中算法的重要性，然后提出分组有限空间的离线规划方法，并在 MADP 工具箱里几个有关 DEC-POMDP 标准测试问题中验证文中提出方法的有效性，通过几组实验对照发现分组有限空间的离线规划方法能够在一定程度降低标准测试问题的求解时间。

本文的工作是以机器人足球 2D 仿真比赛作为研究平台，利用马氏决策理论为球员(即智能体)决策进行建模和求解，通过设置的一系列实验数据统计，体现了本文工作的意义。基于文中的研究成果，GDUT_TiJi 队在 2011 年 RoboCup 中国公开赛获得全国一等奖，并顺利地首次通过机器人足球世界杯预选赛，将于 2012 年 6 月前往墨西哥参加 2012 年 RoboCup 机器人足球世界杯决赛。

关键词：机器人足球；智能体决策；多智能体系统；马尔科夫决策过程

ABSTRACT

Markov Decision Theory (MDT) is an effective tool for the research on the decision-making process of agent. Being the fundamental model, Markov Decision Processes (MDP) is often used to describe and solve the large-scale decision-making problems involving uncertainties. To deal with situations where only limited information from the environment is available to the agent, an extended version of MDP, called Partially Observable Markov Decision Processes (POMDP), was proposed in the literature. With the development of artificial intelligence, more and more researchers are focusing on the topic of decision-making problems of multiple agents, which is also called Multi-Agent System(MAS). Such research topic motivates the development of Decentralized Partially Observable Markov Decision Processes (DEC-POMDP).

This thesis purports to 1) introduce the above-mentioned three important models, MDP,POMDP and DEC-POMDP, contains in MDT;2) depict some algorithms with respect to these models;3)apply these models and associated algorithms to the development of strategies of the players(agents) in RoboCup according to the decision-making problems in RoboCup 2D simulation competition. Specifically, this thesis consists of three major parts, as follow.

In the first part, we firstly revealed the strategic deficiency when the players are in ball possession through analyzing the shortcoming of attack strategy during RoboCup 2D simulation. After re-modeling the attack strategy when in ball possession, we obtained the best strategy by an iterative method based on value function decomposition. Experiments results showed that our model and associated algorithm can significantly enhance the attack performance of our team.

The second part was devoted to improving the performance of the goalkeeper, whose decision must be made in real time given only incomplete information. As remarked before, under the circumstance of imperfect information, POMDP is most suitable for modeling the goalkeeper and perhaps can offer the best solution to block the opponents' attack in urgent situation. Moreover, to ensure the goalkeeper is making a decision. Experiment results suggested that our method can greatly enhance the goalkeeper' s performance.

In the third part we focused on decision-making problems involved in MAS. Although aiming at solving these problems, DEC-POMDP cannot be applied directly to the problems in RoboCup 2D simulation, since its corresponding algorithms can only solve small-scale problems. To overcome this disadvantage, we not only analyzed the DEC-POMDP model and related algorithms, but also performed a series of tests on some benchmark problems of DEC-POMDP using MADP tool box. We found that the bottleneck is the offline solution part of DEC-POMDP. We then proposed a grouping limited space offline planning method to reduce the scale of problem during the solving procedure. Experiment result showed that the running time was shortened compared with the original version based on some benchmark problems in the MADP tool box.

The work in this thesis was conducted on the platform of RoboCup 2D. Several related models and algorithms were employed to tackle the decision-making problems of the players in RoboCup 2D simulation competition. Experiment results verified the importance of our work. Based on the work in this thesis, our team, GDUT_TiJi, won the first prize in 2011 RoboCup China Open, and has already passed qualified for coming 2012 RoboCup World Championships to be held in Mexico in June, 2012.

Keywords: RoboCup; decision-making; multi-agent system; MDP

CONTENTS

第一章 绪论

1.1 引言

随着人工智能技术的发展，智能体与多智能体系统决策的研究已经成为当今控制决策研究领域的重要分支[10]。所谓智能体，不同学者有不同的定义：Shoham 在 1993 年认为，智能体是具有包括信念、能力、选择和承诺等精神状态的一个实体[1]；而 Lane 在 1994 年提出，智能体是一个具有控制问题求解机制的计算单元，可以代表一个机器人、一个专家系统、一个过程、一个模块或一个求解单元等[2]； Stan Franklin 在 1996 年将智能体定义为具有自治性的能够不断感知环境并作用于环境，同时完成其目标的系统[3,11]；Wooldridge 在 1997 年给出新的定义：智能体是封装在一些环境中的计算机系统，为了达到设计好的目标，它能够执行灵活自主的行为[4]。而多智能体系统(Multi-Agent System)则是由多个智能体组成的集合，是一种分布式自主系统，不同于单个智能体，多智能体系统一般具有以下特点：每个智能体只能获得不完全的外部信息和对问题求解的能力；系统不存在全局控制；系统计算的过程可以是异步、并发或者并行的；系统中的智能体可以是异构的[11]。因此多智能体系统与常见的集中式问题是完全不同的结构。

智能体决策的过程被认为是多步决策问题，即智能体不仅仅需要考虑其行为的当前效果，还需要考虑这个行为对智能体未来决策的影响。多步决策问题(Multi-step Decision-making Problem)也被称为序列化决策问题(Sequential Decision-making Problem)，解决此问题的过程也叫做规划(Planning)[12]。一般地说，规划的过程就是要通过一定的方法得到一个行为序列，让智能体能够按照这个行为序列执行并完成指定的任务。比如最短路径规划问题中，输入一个图以及起点和终点，规划的目标就是找到一条最短的路径以满足要求[12]。

人工智能研究的目标是让智能体能够像人一样思考并完成一些需要人类智能和推理才能完成的复杂任务[5]。人类在现实生活中做决策，往往需要对外界环境进行充分理解，通过综合分析并做出相应决策；智能体决策也需要对外界不确定环境进行理解和学习，并利用有限的观察信息等做出决策。通常地说，智能体的传感器收集到的信息是带有噪声的，同时智能体执行部件的控制也往往存有误差，这些不确定性给智能体的决策带来极大的挑战[12]。因此，针对不确定环境下的智能体规划研究

被提出来[12]。本文主要是研究智能体在不确定环境下的决策问题。解决这种问题一般的方法是利用马尔科夫决策理论[10,12]，这也是本文所关注的重点。

1.2 不确定环境下的智能体决策

经典规划问题是不考虑状态转移的不确定性[12]，然而在现实问题中，动作的不确定性是无处不在的。比如在机器人的移动中，由于电子脉冲或者其他机械原因，导致从一步移动到另外一步中出现误差的情况还是很普通的。机器人在做出决策时，还需要提前知道外部环境的具体情况，然而由于传感器通讯等因素，致使机器人无法准确获得外界全面信息，也会对机器人的决策造成影响[11,12]。

不确定环境下决策问题要比经典规划问题复杂的多，马尔科夫决策过程(MDP)为这种决策和规划提供了统一的模型和基础[6]，因此 MDP 是长期以来应用于决策研究的重点方面。目前人工智能的规划研究领域大部分关于非确定性规划的研究都集中在 MDP 框架下[7]。在 20 世纪中末期，MDP 被提出，其主要思想是用随机系统来描述一个规划问题，就是要充分考虑不确定的状态转移系统[8]。在动作结果方面，也用概率分配函数来进行描述。同时要定义反应动作效果的回报函数。MDP 规划的结果是得到一个状态到动作的映射函数：S→A。其规划的思想是找到映射函数能够使系统获得最大的回报累计值[8,18]。

对于单智能体决策问题，MDP 模型提供了较好的建模和求解方式。求解一个 MDP，所获得的决策就是用最少的步骤能够完成目标的规划[9]。而 MDP 决策的目标就是求解出一个能够在决策周期内最大化系统回报值的策略(Policy)，这个决策周期可以是有限的或者无限的，如果是在无限周期进行决策，还要引入一个折扣因子 γ，满足系统回报值折扣累计能够收敛[11]。

MDP 求解智能体决策的一个前提是智能体需要对环境的观察是完全的，而在实际问题中，智能体往往只观察到部分信息，于是 MDP 的一个更接近现实的模型被提出：部分可观马尔科夫决策过程(POMDP)[13]。 在 POMDP 中，智能体由于得到的局部信息被称为观察(observation)，因此就用观察函数(observation function)来对智能体信息的局部性和不完整性进行建模[25]。在 POMDP 决策过程中，智能体不仅需要考虑当前的观察信息，还需要考虑过去的历史信息，从而推断出当前所处状态的概率分布，这个概率分布又被称为信念状态。对 POMDP 问题进行求解的结果是得到

一个信念到动作的映射函数：$B \rightarrow A$。这个行为序列能够最大化智能体系统回报值累计[18,25]。

总体的说，马尔科夫决策理论中，转移函数的建模就是对不确定环境下动作的体现，观察函数的建模是对外界信息的不完整和不确定性的体现[14]。回报函数则根据具体的问题及目标进行描述。规划的过程就是要在有限或无限的决策周期能够得到使智能体累计回报值最大的动作行为序列[15]。由于充分考虑了对环境动作不确定性的分析，马尔科夫决策理论已经成为决策理论研究的基础[16]。

1.3 机器人足球 2D 仿真比赛

机器人足球(RoboCup)比赛是目前国际上一项为促进控制理论、分布式人工智能、智能机器人及相关领域研究和发展的大型比赛和学术活动，为智能机器人提供一个标准的平台来测试其研究与技术的水平[10]。RoboCup 的最初想法是由加拿大大不列颠哥伦比亚大学的 Alan Mackworth 教授在 1992 年正式提出，在 1993 年，通过对系统的调研和可行性分析，日本学者 Minoru Asada、Hiroaki Kitano 和 Yasuo Kuniyoshi 等创办了 RoboCup 机器人世界杯。与此同时，一些研究者也开始将机器人足球作为研究课题。1997 年在第 15 界人工智能联合大会上，机器人足球比赛(RoboCup)正式被列为人工智能研究的一项挑战，因此，机器人足球已经成为人工智能和智能机器人研究领域的标准平台[17]。

2D 仿真比赛时 RoboCup 所有比赛中发展最早，参加人数最多的项目之一。由于避免了物理环境和机器人制造技术的限制，只用几台计算用模拟的方式仿真出机器人足球比赛场景，RoboCup 2D 仿真比赛的研究重点集中在球队高层决策上，包括不确定环境下智能体决策、机器学习和智能规划、多智能体协作与对抗等人工智能研究的热点问题[10]。RoboCup 组委会提供标准 server，即比赛服务器平台，为机器人能够在电脑上进行足球比赛提供了标准的软件和后台服务，同时也对机器人控制、通讯、传感以及体能等方面做了类似于现实环境的限制和仿真。因此开发者只需要开发自己的球队程序，就可以在标准 server 平台上进行机器人足球比赛[25]。由于 2D 仿真比赛平台中并没有考虑高度的概念，因此在这个项目中的研究重点是智能机器人高层策略。

机器人足球队高层决策遇到的难点可以简单归纳如下[10,19]：

- 问题复杂，在比赛中，如果要对场面所有的信息进行描述，包括 22 名球员的位置和速度、球的位置和速度等，则状态空间非常巨大，一般计算机是无法完成这么大规模的计算；
- 实时决策，仿真比赛提供的是一个实时的环境，每个球员决策时间只有几十毫秒甚至更少，同时环境的变化很快也很突然，因此如何设计一个智能球员，能够适应变幻莫测的环境并实时进行决策也是必须考虑的情况；
- 信息不完全，在 2D 仿真比赛中，球员并不能随时获得场面所有的信息，server 平台已经根据实际情况对球员视觉、听觉等感知信息进行限制。球员只能根据自己获得的部分信息进行合理有效的决策；
- 通讯受限，server 平台对球员间的通讯也进行限制，因此球员间的通讯是不可靠的，同时通讯带宽也是有限的，球员间配合就需要考虑如何有效利用受限的通讯；
- 球员间合作与对抗，2D 仿真比赛中一个球队共有 11 个独立的球员，分别担任不同的角色并执行不同的动作；球队中的 11 个球员需要合作来完成比赛；而不同球队中的球员又存在对抗。对球员的合作和对抗进行规划和实现也是一个重要的研究方向。

1.4 GDUT_TiJi 机器人足球 2D 仿真队

GDUT_TiJi(广东工业大学太极队)机器人足球 2D 仿真队于 2006 年开始组队参赛，在 RoboCup 中国公开赛上多次获得全国二等奖和三等奖的好成绩。2011 年在甘肃兰州举行的 RoboCup 中国公开赛中，GDUT_TiJi 获得了全国一等奖(全国排名第四)，同时第一次被推选为 RoboCup 中国公开赛 2D 仿真组技术委员会委员。随后在 2012 年 3 月 RoboCup 世界赛预选赛中，凭借在中国赛区的优异成绩，GDUT_TiJi 顺利地首次通过世界赛预选赛，将于 2012 年 6 月参加在墨西哥举行的 2012 年 RoboCup 世界赛决赛。

GDUT_TiJi 机器人足球 2D 仿真队在备战中国赛和世界赛的过程中，一直致力于智能体决策的研究，尤其是单智能体不确定环境下决策和多智能体协作策略的研究。本文的工作是以决策论为基础，对机器人足球 2D 仿真比赛中产生的智能体决策问题进行建模和求解，同时开展智能体决策相关的研究。

1.5 论文的内容和组织结构

本文的内容和结构如下：

第一章：绪论部分，简单介绍了智能体决策的概念以及机器人足球 2D 仿真平台，为本文后期研究内容做铺垫。

第二章：简单介绍马氏决策理论中的 MDP，POMDP 模型及相关求解算法以及具有分布式结构的 DEC-POMDP 模型。

第三章：介绍机器人足球 2D 仿真比赛中的进攻策略，利用 MDP 对比赛中持球队员的决策进行研究，并提出了值分解的迭代求解方法，将这种求解方法应用于 GDUT_TiJi 2D 仿真球队，通过一系列实验和比赛证明了本方法能够有效地提高球队的进攻能力。

第四章：分析机器人足球仿真比赛中守门员决策的构造及存在问题，守门员决策是在动态不确定性环境中进行，往往需要对不完全的外界信息进行分析并作出决策，因此选用 POMDP 模型为守门员决策的过程进行建模。通过建模及求解分析，提出了基于临界状态的求解方法，然后将这种方法应用到机器人足球仿真球队的守门员决策中，并通过实验证明了本方法利用 POMDP 模型构造的守门员决策比常规的基于规则的方法表现得更好。

第五章：多智能体系统(MAS)的研究是目前人工智能研究领域的一项挑战。DEC-POMDP 为分布式多智能体系统在非确定环境下的决策提供了较好的模型，文中对 DEC-POMDP 模型及其相关的离线规划求解算法进行总结和分析，同时利用 MADP 工具箱中研究 DEC-POMDP 模型相关算法的一系列标准测试问题(benchmark problem)进行测试；然后提出了分组有限空间的离线规划方法，并在 MADP 工具箱中进行实验。通过实验数据的统计与分析验证了分组有限空间离线规划方法能够一定程度降低 DEC-POMDP 模型标准测试问题的求解时间。本章的结论也将有助于 DEC-POMDP 模型应用到很更现实的多智能体系统决策问题中。

第二章 马尔科夫决策理论

智能体的决策与过程是息息相关的，在智能体决策时，其相应的过程也会根据决策内容向有利的方向改变。马尔科夫过程是一种具有普遍共性的过程，其原始模型是马尔科夫链，于 1907 年被俄罗斯数学家 markov 提出。马尔科夫过程具有以下特性：某阶段的过程如果确定，则此后过程的演变将不与之前的过程有关，即后期过程的演变不依赖于前期的过程[10]。在现实生活中，具有马尔科夫过程性质的运动有很多，比如液体中微粒的布朗运动，传染病受感染的人数、车站的候车人数等。荷花池中青蛙的跳跃是马尔可夫过程的一个形象化的例子。青蛙依照它瞬间的念头从一片荷叶上跳往另一片荷叶上，因为青蛙是没有记忆的，当现在所处的位置已知时，它下一步跳往何处与它之前走过的路径无关。如果将荷叶编号并用 $X_0, X_1, X_2, \cdots$ 分别表示青蛙最初的荷叶号码及第 1 次、第 2 次、……跳跃后所处的荷叶号码，那么$\{Xn，n\geqslant 0\}$ 就是马尔可夫过程[20]。

2.1 马尔科夫决策过程

马尔科夫决策过程(Markov Decision Process, 简称 MDP)，就是由智能体即决策者参与的马尔科夫过程。在现实问题中，很多决策问题是带有极大的不确定性，马尔科夫决策过程的模型的提出就是为解决这种对环境动作不确定性的决策问题。在上个世纪 50 年代，bellman 首先提出了序列决策问题[8]，后来 Howard 就开始使用马尔科夫决策过程这个名词，通过系统地讨论这个模型从而提出了有效的算法：值迭代算法和策略迭代算法[21]。在马尔科夫决策过程发展初期的 30 多年时间内，大部分研究集中在最优方程和求解算法，也就是策略迭代和值迭代的方法。近 20 多年，马尔科夫决策过程的主要研究在多准则问题等方面。目前马尔科夫决策过程已经成功地应用到很多场合，比如生态科学、经济理论、通信工程及其他学科方面[18]。

2.1.1 基本模型

马尔科夫决策过程主要是描述智能体与环境之间相互作用的模型，如图 2.1 所示。智能体接受外界环境的状态作为输入，随后产生的动作作为输出，这些动作会

相应地影响环境的状态。需要指出的是，马尔科夫决策过程中的智能体是具有完全感知能力的，也就是说智能体能够得到全局的外界信息[10]。

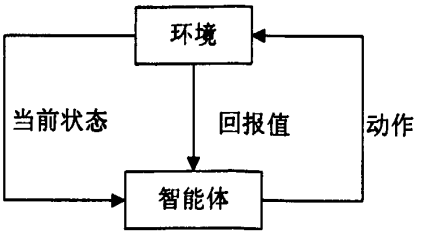


图 2.1 马尔科夫决策过程

Fig. 2.1 Markov Decision Processes

马尔科夫决策过程的模型一般用四元组〈S,A,T,R〉表示：

S：表示可能的外界环境状态的有限集合；

A：表示可能的动作的有限集合；

T:表示状态转移函数，用$T(s'|s,a)$表示在状态 s 执行动作 a 达到状态s'的概率；

R：表示回报函数，用$R(s,a)$表示智能体在状态 s 执行动作 a 可以获得的立即回报值。

2.1.2 外界环境状态

外界环境状态是在某时刻对该系统环境的描述。不同情况下，人们对外界环境状态的定义是不同的，但是一般地说，所定义的外界环境状态必须包含所有当前世界中可以让智能体做出决策的信息[10]。一般的描述采用平铺式的表示，即对所有可能的环境状态进行标号，用s_1，s_2，s_3...类似的方法进行描述。这时候，标号的环境状态的数目也就代表了状态空间的大小。Fan 等提出过因子化的表示方法，将每个状态看成由多个因子组成的多元组，也具有一定的意义和参考性[11]。

2.1.3 动作集

智能体的动作可以影响系统当前的环境状态，因此动作集是马尔科夫决策过程中智能体决策方面重要的部分，也是智能体所有可能采取的动作的集合。在决策过程中，当执行某一个动作时，当前的外界环境状态会发生改变，并按照一定的概率转换到下一个状态，其转换的概率与所执行的动作有关。马尔科夫决策过程中，任何动作都可以在任何状态下执行，不过这个动作或许会对某状态没有影响甚至有负

面影响[6]。在经典人工智能规划中，可以根据先决条件来决定哪些动作在该状态下是有意义的，这样的优点就是根据先决条件已经排除了很多无关的动作集，因此在计算的时候所需考虑的后继状态空间就相对小[21]。马尔科夫决策过程中则没有先决条件，是一种全局的综合决策过程，必须充分考虑在各种状态下采取所有动作的规划。如果能够在马尔科夫决策过程中引入经典人工智能规划的先决条件，尽可能地缩小状态空间，将在一定程度上提高决策过程的运算。

2.1.4 状态转移函数

状态转移函数是在马尔科夫决策过程中重要的部分，描述了系统的动态特性，一般分为确定环境和随机环境下动作的状态转移函数[11]：

确定环境下的动作：T：$S*A\rightarrow S$，在状态 s 下执行动作 a 可以直接到达一个确定的状态；

随机环境下的动作：T：$S*A\rightarrow Pr(S)$，在状态 s 下执行动作 a，得到一个状态的概率分布$Pr(S)$，也可记作$T(s'|s,a)$。

如图 2.2 所示，描述了执行某个动作后，系统中状态间的转移情况：当指定某动作后，状态 S_0 转移到 S_1 的概率为 0.6，转移到 S_4 的概率是 0.4[25]。

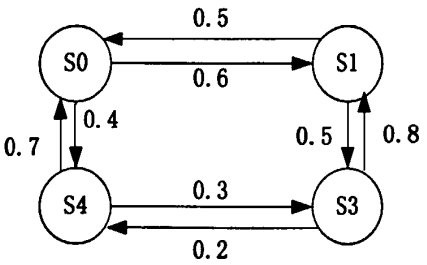


图 2.2 状态间的转移

Fig. 2.2 Transfer of state

2.1.5 回报函数

马尔科夫决策过程类似于非监督学习[48]，在整个决策过程中无法得到监督信号，只能根据已经确定好的回报函数来指导整体规划。每个动作都可以得到即时回报(记作 R，即 reward)，决策的过程总是希望能够按照某个标准来选择动作，进而使长期回报累计值最大。

计算回报累计值有两种常见的方式：一种是有限阶段的最优准则，要求决策过程中能够最大化有限阶段的总回报值，也就是$\max \mathrm{E}\left[\sum_{t=1}^{k} R_t\right]$，这里的$R_t$就表示智能体在第 t 步得到的收益，利用这种方式往往需要知道 k 的具体值。第二种方式就是考虑整个决策过程无限阶段的最优准则，要考虑智能体在整个过程中的总回报和，因此需要引入一个折扣因子 γ (0< γ <1)，此时整个过程中的总回报值就是 $\max \mathrm{E}[\sum_{t=1}^{\infty} \gamma R_t]$，折扣因子就是为了保证总回报值的收敛，便于找到一个最优的策略[9]。

马尔科夫决策过程的解被称为策略，描述为从状态集合到动作集合的映射，即 $\pi: S \rightarrow A$。求解策略的过程就是:首先智能体需要知道目前所处的状态 s，然后执行策略所对应的动作$\pi(s)$，并进入下一个状态，重复这个过程一直到问题结束。求解过程中的策略有如下区分：如果动作的选取只与当前状态有关，而与时间无关，则称为平稳策略；如果对于同样的状态，在过程不同的时刻可能会对应不同的动作，则称为非平稳策略[10,11,12,25]。

2.1.6 值函数与策略求解

在马尔科夫决策过程中，对于任何一个策略，都可以执行这个策略所能获得的长期期望回报来评价其优劣，首先定义值函数为智能体在状态 s 采取策略π时的期望回报[12]：

$$V^{\pi}(s) = \mathrm{E}[\sum_{t=1}^{\infty} R(s_t, \pi(s_t))] \tag{2.1}$$

其中s_t表示在 t 时刻的状态。

由于值函数是期望回报值的累积，因此可以用递归的方式来描述值函数：

$$V^{\pi}(S) = R\left(s, \pi(s)\right) + \gamma \sum_{s' \in S} T^{\pi(s)}(s, s') V^{\pi}(s') \tag{2.2}$$

通过公式 2.2 可知对于每个策略，其对应的值函数是一系列线性方程的唯一解。

上述方式为通过策略计算值函数的方法，但是在实际求解过程中，我们需要通过值函数求解出策略，因此需要引入一个在求解过程中常用到的中间变量，定义动作值函数$Q^{\pi}(s,a)$为智能体在状态 s 采取动作 a，其他状态下采取策略π时的期望回报：

$$Q^{\pi}(s,a) = R(s,a) + \gamma \sum_{s' \in S} T^{a}(s, s') V^{\pi}(s') \tag{2.3}$$

此时策略并未显示的记录，只有值函数 V 时，动作值函数记作 Q，策略π就可以通过下面公式计算：

$$\pi(s) = \arg\max_{a\in A} Q^{\pi}(s,a) \tag{2.4}$$

这里通常采用的是一步前瞻的贪婪搜索，因此这种计算方式也被称为贪婪搜索，即：

$$\pi(s) = \max_{a\in A}\{R(s,\pi(s)) + \gamma\sum_{s'\in S} T^{a}(s,s')V(s')\} \tag{2.5}$$

同时，值函数可以按照下式进行更新：

$$V^{\pi}(s) = \max_{a\in A}\{R(s,\pi(s)) + \gamma\sum_{s'\in S} T^{a}(s,s')V(s')\} \tag{2.6}$$

在计算过程中，将最优策略记为π^*，其对应的最优值函数记为V^*。如果一个策略π满足对状态 s，有$V^*(s) - V^{\pi}(s) \le \varepsilon$时，则称$\pi$为在状态 s 处的ε最优策略；如果策略π对所有的状态都满足上条件，则称π为该问题的ε最优策略[10]。

2.1.7 MDP 典型算法

MDP 模型早期求解算法有值迭代和策略迭代两种方法，随后，一些利用状态可达性的前向搜索算法，如 AO*，LAO*(Hansen E A, Zilberstein,2001)被相继提出，这些方法的特点是只求解从给定初始状态开始的最优策略，通常能够避免大量的不必要计算，获得更高的效率[22,25]。

从算法执行角度，MDP 模型求解算法又分为在线和离线求解两种类型。在很多现实世界中存在大规模问题，无论是否利用状态可达性，其问题的解都不可能以离线的方式一次性求出，因此需要一种在线算法，也称为实时算法，根据实际问题进行求解，避免产生大量无用计算从而影响问题求解[12]。实时算法中决策计算和执行交替进行，同时求得的解的效果也随着给定计算时间的增加而提高。最早基于动态规划的实时算法是 RTDP(Barto A G, et al. 1995)，RTDP 执行过程中首先确定一个从初始状态到目标状态的路径并进行反向的值迭代，然后不断地循环执行这个过程，直到将所有的策略执行完毕并找到一个最优的策略[23]。根据求解过程中是否已经达到要求并停止搜索以及降低搜索问题规模等，一些基于 RTDP 的算法相继被提出：Labeled RTDP(Bonet.B, Geffner,2003)，BRTDP(McMahan HB,2005)及 FRTDP(Smith T, Simmons R,2006)等[24,26]。

下面将介绍几种常用的 MDP 求解算法：

算法一：策略迭代算法，这种算法中策略是显式表示的，可以计算得到对应的值函数V^{π}，然后利用公式 2.3 和 2.4 改进策略。其中引入折扣因子γ，从而保证算法

经过有限步的迭代后能够收敛到最优策略，算法描述如下：

算法 1：策略迭代算法
1. 初始化策略π 2. 计算策略：利用策略π计算值函数V^{π}，计算公式如下： $V^{\pi}(s) = R(s,\pi(s)) + \gamma\sum_{s'\in S} T^{\pi(s)}(s,s')V^{\pi}(s')$ 3. 更新策略：利用公式 2.3 和 2.4 更新策略 4. 策略收敛：如果$\pi \approx \pi'$，则返回第 5 步；否则，令$\pi = \pi'$并返回第 2 步 5. 返回最优策略

算法二：值迭代算法，在值迭代中，策略是不能显式表示的，整个过程按照动态规划的 bellman 公式不断进行迭代来更新值函数：

$$V(s) = \max\{R(s,a) + \gamma\sum_{s'\in S} T^{a}(s,s')V(s')\} \tag{2.7}$$

值迭代算法中，策略是根据值函数的误差界限进行更新，从而收敛到最优策略。值函数误差界限一般通过在迭代过程中的 Bellman 误差和平均初过时间$\emptyset^{\pi}(s)$计算得到[8]。每次迭代所有状态值函数更新前后的最大差值成为 Bellman 误差 r：

$$r = \max_{s\in S}|V(s) - V'(s)| \tag{2.8}$$

平均初过时间是指从状态 s 开始，按照策略π执行，达到目标状态的期望步数，用$\emptyset^{\pi}(s)$表示，计算方式如下：

$$\emptyset^{\pi}(s) = 1 + \sum_{s'\in S} T(s,\pi(s),s')\emptyset^{\pi}(s') \tag{2.9}$$

通过公式 2.8 和 2.9 计算值函数的上界V_U和下界V_L：

$$V_L(s) = V^{\pi}(s) - \emptyset^{\pi}(s)r \tag{2.10}$$

$$V_U(s) = V^{\pi}(s) + \emptyset^{\pi}(s)r \tag{2.11}$$

在值迭代过程中，当前值函数就是最优值函数的下界，上界和下界的最大差值定义为σ，当$\sigma < \varepsilon$时，得到的策略就是ε最优。对于给定的任意$\varepsilon > 0$，迭代过程中经过有限步的迭代都可以收敛于ε最优[10]。下面是值迭代算法的描述：

算法 2：值迭代算法
1. 初始化值函数 V 和任意$\varepsilon > 0$ 2. 利用公式 2.7 更新值函数 V 3. 如果$\sigma < \varepsilon$，返回第 4 步；否则，令$V = V'$并返回第 2 步 4. 返回ε最优策略π

算法三：实时动态规划算法[23]，前面介绍的算法都可以认为是在一个状态空间的搜索问题，定义好一系列状态、动作和一个已定义的回报函数的情况下，找到从起点状态到目标状态最大收益的函数。这里的搜索往往不可避免地进行穷举式搜索，即对所有可能的情况进行分析和考虑。实时动态规划算法(real time dynamic

programming，简称 RTDP)则利用所有状态可达性的前向搜索结构来避免穷举所有的状态，从而降低运算量并取得较好的效果[23]。

RTDP 将计算组织成一系列试验测试(trials)，每次试验由多步组成，每一步的动作基于一步前瞻搜索选择，然后基于所选择的动作所有可能的结果对当前状态进行更新，并在达到目标状态或者经过一个指定步数的时候停止更新。这种方法的一个主要的特征是只更新那些基于当前值函数并采用贪婪策略的动作选择。因此 RTDP 方法可以省掉大量无关状态空间的计算，并且能渐近收敛于最优解[23]。下面是 RTDP 算法描述：

算法 3：实时动态规划算法
1. 初始化可达状态空间 2. 重复下步骤 n 次 Trails: 对当前状态 s 设置可达状态集，并更新 m 次直到达到目标状态: 对当前状态 s 进行更新(利用公式 2.7)，同时计算值函数 3. 在状态 s 下执行动作 a 并随机达到下一状态 s'，同时衡量动作 a 是否为最佳动作。 4. 通过公式 2.5 从值函数 V 中获取策略。

上面三种 MDP 求解算法主要是对 MDP 模型中的值函数或策略函数进行迭代，通过搜索的方式找到最优策略；在现实中，基于 MDP 模型的问题往往具有较大的状态空间，如果采用穷举搜索的方式来求最优策略，往往会因为运算量巨大而无法求解。目前在 MDP 求解的研究上，大部分研究者都集中考虑如何在保证模型完整性的基础上降低运算量从而得到最优策略。2012 年 AAMAS 上，Bai 提出一种基于 MAXQ 分解的方式对大规模 MDP 问题进行在线求解，采用启发式方法对大规模 MDP 问题的可能状态集进行分解，取得较好的效果，然而对大规模 MDP 问题的求解依旧是一个待解的问题[47]。

2.2 部分可观马尔科夫决策过程

部分可观马尔科夫决策过程(POMDP)是基于马尔科夫决策过程(MDP)提出的新模型。在现实世界中，智能体往往不一定能够即时获知外界信息，这就引起了部分可观察情况下智能体决策的问题，即智能体只能从外界环境获取部分信息，而不能知道全局的信息[11,12,18]。图 2.3 所示为 POMDP 模型：

简单地解释，POMDP 就是构造一个观察到动作的映射，用观察值取代在 MDP 中的状态值，然后按照一定的方法计算得到最优值函数，从而推导出最优策略。但

是 POMDP 模型并不是简单地对 MDP 添加观察值就可实现的。一个重要的问题是，观察值中是否包含历史信息，如果不包含有历史信息，那么假设观察值与现实状态差距太

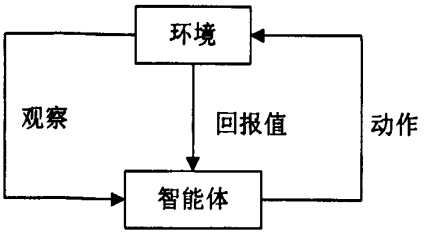


图 2.3 POMDP 的基本模型

Fig. 2.3 Simple Model of POMDP

大，这种方式求出的策略会非常差。为解决这个问题，一种采用随机的方法被提出，也就是用观察动作映射的形式来描述策略，但是用概率分布来替代确定的动作，智能体在决策时候就可以根据这个概率分布来随机地选择动作[28]。

在 POMDP 模型问题的求解中，观察与现实状态之间往往存在差距，智能体在决策过程中需要根据历史动作和观察来判断目前所处的状态，此时的判断带有极大不确定性，因此又引入信念状态的概念，描述智能体当前处于某状态的概率。如图 2.4 所示：

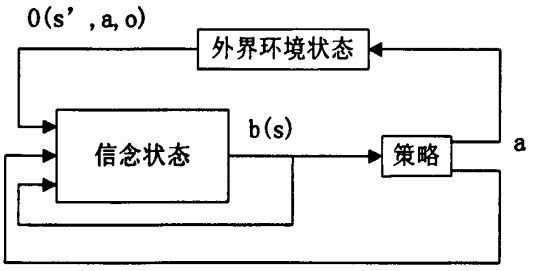


图 2.4 POMDP 模型

Fig. 2.4 POMDP model

与 MDP 模型类似，POMDP 同样包括一个状态集合、一个动作集合、状态转移函数和回报函数，除此之外，还包括有一个观察集合，因此一般地用一个八元组 $< S,A,\Omega,T,O,R,B,\gamma >$来描述 POMDP 模型：

其中 S,A,T,R,γ 与 MDP 中所定义的类似；

$\Omega = \{o_1, o_2, \ldots o_n\}$描述智能体所能得到的观察的有限集合；

O：$S * A \rightarrow \prod(\Omega)$称为观察函数，描述在给定动作和结果状态的情况下智能体可能得到的观察的概率分布。$O(o,a,s') = p(o|s',a)$表示执行动作 a 转换到状态 s' 得

到观察 o 的概率；

B：智能体的信念状态空间，用 $b(s)$来描述智能体当前处于状态 s 的概率。

在 POMDP 问题中，智能体的决策过程要根据从外界环境中得到的不确定因素即观察推断出信念状态并做出决策，从而转入下一状态并获得即时回报。POMDP 问题的决策过程如图 2.5 所示：

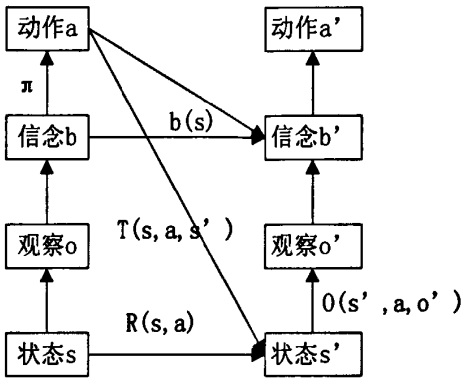


图 2.5 POMDP 问题决策过程

Fig. 2.5 Decision-making processes of POMDP

2.2.1 观察集合

观察集合描述的是智能体在决策过程中能从外界环境中得到的信息，当能够得到所有外界环境信息时，POMDP 模型就演变为 MDP 模型。观察值的描述与 MDP 中对状态的描述类似[29]。

2.2.2 信念状态

基于 POMDP 模型的智能体不能保证每步都获得全部的当前状态信息，因此引入了信念状态的概念。信念状态是智能体根据观察及历史信息计算得到对自己当前所处状态的概率分布，用 $b(s)$来描述，对任意的 s，有$0 \leq b(s) \leq 1$，且$\sum_{s \in S} b(s) = 1$。这个概率描述的是智能体主观上认为当前应该处于的状态，因此称为信念状态[30]。

在 POMDP 中，信念状态是智能体的主观概率，每当收到新的观察后，智能体就要更新信念状态，一般采用的是贝叶斯公式进行更新[28,29]：

$$b'(s') = \Pr(s'|o,a,b) = \frac{Pr(o|s',a,b)\Pr(s'|a,b)}{Pr(o|a,b)}$$
$$= \frac{Pr(o|s',a,b)\sum_{s\in S} Pr(s'|a,b,s)Pr(s|a,b)}{Pr(o|a,b)}$$
$$= \frac{Pr(o|s'a)\sum_{s\in S} Pr(s'|a,s)Pr(s|b)}{Pr(o|a,b)} = \frac{O(s',a,o)\sum_{s\in S} T(s,a,s')b(s)}{Pr(o|a,b)} \tag{2.12}$$

其中有:

$$Pr(o|a,b) = \sum_{s'\in S} Pr(o,s'|a,b)$$
$$= \sum_{s'\in S} Pr(s'|a,b)\,Pr(o|s',a,b)$$
$$= \sum_{s'\in S}\sum_{s\in S} Pr(s'|a,s)\,b(s)\,Pr(o|s',a) = \sum_{s'\in S} O(s',a,o)\sum_{s\in S} T(s,a,s')b(s) \tag{2.13}$$

通过对贝叶斯公式进行反复迭代，就可以得到信念状态的更新公式。信念状态的更新就可以根据当前信念状态，转移函数 T 集观察函数 O 得到新的信念状态的概率分布。

2.2.3 策略与值函数表示

在 MDP 问题中，策略通常可以用状态到动作的映射来表示，但是在面对连续的信念空间的 POMDP 问题时，智能体的 t 步策略通常会用策略数的形式来表示。根节点决定要采取的第一个动作，通过观察，选择一条边到达下一个节点，这个节点就会决定下一步的动作。

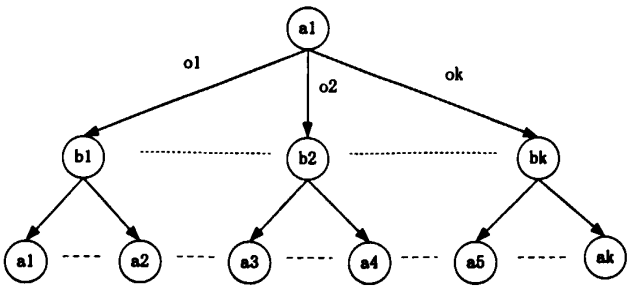


图 2.6 POMDP 策略树

Fig. 2.6 Decision Tree of POMDP

同 MDP 中类似，POMDP 在执行动作后也会有一个回报函数返回即时回报值。

POMDP 按照某策略π执行，为衡量策略的优劣，同样引入期望回报的值函数，如下：

$$V^{\pi} = \sum_{t=0}^{\infty} \gamma \sum_{s \in S} b_t'(s) R(s, \pi(b_t)) \tag{2.14}$$

其中γ是折扣因子，且$0 < \gamma < 1$，$b_t'(s)$是在 t 时刻智能体处于状态 s 的概率，而$\pi(b_t)$则是由策略和信念状态决定的动作。求解 POMDP 问题就是要找到一个最优策略π^*，该策略决定在每一步智能体应该选择的动作，并且保证在智能体执行的总周期内能够获得最大的回报。最优策略π^*的求解方式如下：

$$\pi^* = \arg\max_{\pi} V^{\pi} \tag{2.15}$$

前面介绍策略可以用树型结构进行描述，假设Π表示所有策略，那么策略的数量级可以表述为：

$$\left|\Pi_t\right| = |A|^{\frac{|\Omega|^t - 1}{|\Omega| - 1}} \tag{2.16}$$

因此策略的数量会随着 t 的增加而呈指数级的增加。若 POMDP 问题规模增大，则值函数的计算量将超出想象[18]。在 2.14 式中知道，值函数可以表示为 b 上的线性函数，即可用下式描述：

$$V = \alpha^{\pi} \cdot b \tag{2.17}$$

此时每个策略就对应一个α向量，Smallwood 和 Sondik 证明了最优策略下的值函数是 b 上的分段线性凸函数[31]，即可以用一个α向量集Γ来表示最优值函数：

$$V^* = \max_{\alpha \in \Gamma} \sum_{s \in S} \alpha(s) b(s) \tag{2.18}$$

因此将 POMDP 问题的策略求解的问题转变为寻找最优向量集Γ。

2.2.4 POMDP 策略求解

POMDP 策略的求解，与 MDP 问题中策略求解类似，也分为值迭代和策略迭代两种。值迭代的主要思想是用值函数来评价执行某策略的优劣，一步一步构造策略，利用上一阶段的值函数更新下一阶段的值函数。上面已经介绍用α向量的方式表示值函数，求解最优策略的过程其实也就是不断更新α向量集的过程[31]。

值迭代方法是从初始值函数开始，通过迭代的方式更新值函数，当两个连续的值函数之间的差别小于预定的阈值时，终止算法，并输出最终得到的值函数与策略[18]，描述如下：

算法 4：POMDP 值迭代算法(V, ε)
1. 初始化 V， ε 和 t 2. 更新值函数 $V_{t+1} = TV_t$，t=t+1 当$\max_{b\in B}\|V_{t+1}(b) - V_t(b)\| \le \varepsilon$，则返回第 3 步 3. 返回$V_t$

在更新值函数的过程中，可以通过不同的方法来选取和计算信念状态与动作，基于这种思想目前已经有很多求解 POMDP 问题的算法：Sondik 和 Monahan 提出的枚举方法[29,31]，Smallwood 和 Sondik 提出的 One-pass 算法[31]，Cheng 提出的线性支持法[32]，Littman 提出的证据算法[28]，以及 Zhang，Liu 和 Cassandra 提出的增量修剪算法等[33,34]。这些算法都能够精确地求解有限范围内的 POMDP 问题，但对于大规模 POMDP 问题则无法顺利求解[18]。

策略迭代是先从策略开始，每次迭代的过程中，用一个新的策略来取代初始的策略，如果两个连续的策略得到的效果差别不大，则迭代终止并输出策略[18]。描述如下：

算法 5：POMDP 策略迭代算法(π, ε)
1. 初始化策略π，阈值 ε 和 t 2. 更新策略 $\pi_{t+1} = T\pi_t$，t=t+1 3. 当两个策略的差别不大时，返回π_t

策略迭代的方式求解 POMDP 问题采用的是直接优化策略的方法，也可以理解为在策略树上寻找一个最佳的路径。策略执行的过程就是选择当前节点应该对应的动作，沿着观察对应的边，到达下一个节点。目前基于策略迭代的算法有 Aberdeen 等人提出了 GA(gradient ascent)算法[35]，但容易陷入局部最优；Braziunas 提出 BBSLS(belief-based stochasyic local search)算法[36]，能够一定程度避免出现局部最优的情况。策略迭代是一种比值迭代更有效率的方法，具有比较高的收敛速度，能够相对快速地寻找到最优策略，但是不能保证对每个问题找到最优的策略，同时很容易陷进局部最优[18,25]。

2.3 分布式部分可观马尔科夫决策过程

近年来在人工智能领域，大部分研究者对多智能体协作完成共同目标的问题产生了浓厚的兴趣，在这类问题中，智能体之间通常不一定有可靠的通讯，同时智能体也并不一定清楚其他合作者的信息。也就是说在同样的环境中，不同的智能体对

自身和外界状态的认识是不同的。这类问题比单智能体和集中式信息共享的多智能体协作问题更难解决。早期的研究一定程度上依赖于合作协议的设计，即用信念、愿望或意图等对智能体的动作进行表示和推理[11,12]。分布式部分可观马尔科夫决策过(DEC-POMDP)为解决这类问题提供了较好的模型。

DEC-POMDP 建模的是一组智能体在非确定环境下决策的过程，可以形式化地定义为一个多元组$< I, S, \{A_i\}, \{\Omega_i\}, P, O, R, b^0 >$，每一部分含义如下：

- I是指智能体的集合，即为智能体进行编号，若I=1，则模型等价于 POMDP 模型；
- S表示有限的状态集合。其定义与马尔科夫决策过程中一样，涵盖了智能体决策所需要的所有信息。
- A_i是智能体 i 可以采取的动作的集合。$\vec{A} = \times_{i \in I} A_i$描述的是所有智能体的联合动作集，其中$\vec{a} = < a_1, a_2, \cdots a_n >$则表示一个联合动作。
- Ω_i表示智能体 i 获得的观察集合。同理$\vec{\Omega} = \times_{i \in I} \Omega_i$表示所有智能体的联合观察集，其中$\vec{o} = < o_1, o_2, \cdots o_n >$表示一个联合观察。
- P：$S \times \vec{A} \times S \rightarrow [0,1]$表示系统的转移函数。$P(s'|s, \vec{a})$表示在状态 s 下系统采取联合动作$\vec{a}$后转移到状态 s' 的概率。
- O：$S \times \vec{A} \times \vec{\Omega} \rightarrow [0,1]$表示系统的观察函数。$O(\vec{o}|s', \vec{a})$表示系统中采取联合动作$\vec{a}$后转移到新状态 s' 时获得联合观察$\vec{o}$的概率。
- R:$S \times \vec{A} \rightarrow R$表示系统的回报函数，描述了在状态 s 下采取联合动作$\vec{a}$后系统所获得回报。
- $b^0 \in \Delta(s)$表示系统的初始状态分布。

考虑到决策周期是离散的，用 T 代表决策周期，则在每个决策周期 t=1,2,3…,T-1 内，每个智能体做出决策并执行相应的动作，所有智能体的联合动作使系统从一个状态转移到另一个状态，同时系统也进入下一个决策周期[11]。在下一个决策周期中，智能体从系统再次获得观察，然后做出决策并执行新的动作，如此反复。在每个决策周期，根据对回报函数的定义，智能体执行完动作都会有一个即时回报 $r(t) = R(s, \vec{a})$。用累计回报值来衡量决策的效果。因此 DEC-POMDP 最优策略的依据就是找出一个联合策略$\vec{\pi}$使整个决策过程的累计回报值最大，即：

$$V(\vec{q}) = \max[\sum_{t=1}^{T} R(s, \vec{a}) | \vec{\pi}, b^0] \tag{2.19}$$

DEC-POMDP 是多智能体系统决策的模型，其策略集相应地表示为观察的历史序列集Ω_i^*到动作集A_i的映射，即Q_i：$\Omega_i^* \to A_i$。给定一个策略$q_l \in Q_i$，智能体 i 就可以根据其观察的历史信息$o_i^* =< o_i^1, o_i^2 \cdots o_i^t >$，选择动作$a_l \in A_i$。所有智能体策略的向量集合则被定义为联合策略，即$\vec{q} =< q_1, q_2, \cdots q_n >$。在联合策略下，所有智能体的期望累计回报值又称为策略的值函数，即在给定状态 s 下，所有智能体在策略$\vec{q}$下的值函数可以利用 bellman 公式迭代得到：

$$V(s,\vec{q}) = R(s,\vec{a}) + \sum_{s' \in S} \sum_{\vec{o} \in \vec{\Omega}} P(s'|s,\vec{a}) O(\vec{o}|s',\vec{a}) V(s', \overrightarrow{q_{\vec{o}}}) \tag{2.20}$$

其中$\vec{a}$是根据策略$\vec{q}$得到的联合动作，而$\overrightarrow{q_{\vec{o}}}$是执行动作后根据$\vec{q}$和所获得观察信息$\vec{o}$得到的联合子策略。在 DEC-POMDP 模型中，通常不会给定状态 s，而通过信念状态 b 来描述智能体当前可能处于的状态，这种情况下，策略的值函数就相应地变换为：

$$V(b,\vec{q}) = \sum_{s \in S} b(s) V(s,\vec{q}) \tag{2.21}$$

求解一个 DEC-POMDP 问题的过程就是找到一个联合策略$\vec{q}$使得在给定的初始状态分布b^0的情况下，其值函数$V(b^0,\vec{q})$最大[12].

多智能体系统协作问题的研究上，DEC-POMDP 是一个很常用的模型。从理论上讲，DEC-POMDP 模型确实能为多智能体系统的决策建模提供完整的依据，但是其问题复杂度极高，求解这类模型还是非常困难的。尤其要为实际问题进行建模，其状态描述方式、策略表现形式以及状态、观察转移函数的确定非常困难的[11]。

在 MDP 和 POMDP 求解中分析过，问题的规模要随着决策周期呈指数级增加，而在 DEC-POMDP 问题中，参与决策的智能体数目会不断增加，其问题规模将会随着决策周期呈双指数级增加[12]。假设用离线的枚举法来对每个智能体的决策过程进行描述，巨大的状态和策略空间将使小规模 DEC-POMDP 问题的求解都无法实现。

2.4 本章小结

本章要介绍了马尔科夫决策理论相关模型，对马尔科夫决策过程(MDP)和部分可观马尔科夫决策过程(POMDP)模型和策略求解算法进行详细介绍，同时简单介绍了分布式部分可观马尔科夫决策过程(DEC-POMDP)模型。这几个模型都是马尔科夫决策理论中重要的理论模型，为智能体决策过程的求解提供了有力的工具和方法。本文研究内容基本上是围绕本章所介绍的模型和方法展开。

第三章 基于 MDP 模型的智能体决策

MDP 适用于非确定环境下智能体决策问题的研究，同时应用于多个领域，如智能机器人领域、经济学等方面。目前对 MDP 的研究主要包括现实问题的 MDP 建模和大规模 MDP 问题的求解两方面。建模是应用 MDP 关键的一步，建模过程如果不完整，再优秀的算法也很难有较好的表现；充分完整的状态空间在一定程度上能够有助于智能体决策，但是太大的状态空间又往往导致运算量过大，在实时决策环境下依然不能得到很好的表现。状态转移函数是系统不确定性的表现，通过什么途径和方式描述状态转移函数以及状态转移函数如何能够反映真实环境的不确定性也是建模过程的重点。

本章的工作是将 MDP 模型应用于机器人足球 2D 仿真球队中的进攻决策。首先介绍机器人足球 2D 仿真球队的决策体系，然后对进攻决策存在问题进行分析，并提出 MDP 对球员进攻决策进行建模与求解。

3.1 机器人足球 2D 仿真决策体系

机器人足球 2D 仿真比赛是由球队开发者自己设计球队，通过比赛服务器 server 和比赛仿真平台 monitor 实时更新并播放比赛情况。任何球队都需要基于 server 规定的足球规则进行开发，首先保证球队能够与服务器 server 正常通讯并可以完成部分底层动作，在此基础上，球队开发者可以进行球队高层决策的开发与研究[10,19,25]。如图 3.1 所示，球队的架构介绍：

高层决策是一个球队的灵魂，左右着球队的整体表现。完成通讯和底层动作的设计，球队并不能进行正常的比赛，必须在一个大脑，就是高层决策的指导下，球队中球员才能合理选择相应的动作或战术。高层决策实时地对每个球员的动作进行指导，因此球员能够根据高层决策对自己的情形进行分析然后调用动作发生器并执行相应的动作[38]。在高层决策中，需要根据球员的角色和位置对球员的动作进行分析，如守门员有自己固定的活动范围和独特的动作，在禁区外，守门员不能扑球；前锋、中场和后卫等角色活动范围稍有区别，主要动作如传球、带球、跑位等式一致的，但是只有在对方禁区附近的一个范围内才可以执行射门动作[19]。机器人足球

2D 仿真球队的决策体系如图 3.2 介绍：

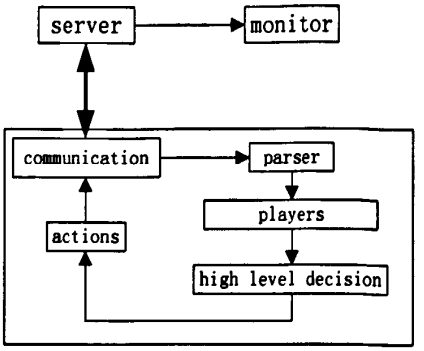


图 3.1 机器人足球 2D 仿真球队构架

Fig. 3.1 Framework of RoboCup 2D simulation

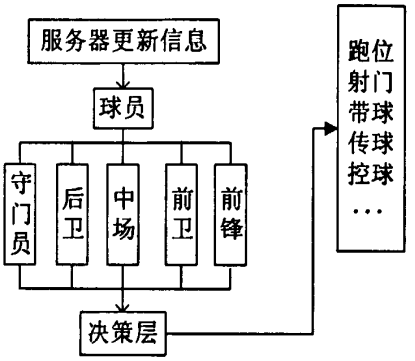


图 3.2 决策体系

Fig.3.2 Decision system of RoboCup 2D simulation

3.2 机器人足球 2D 仿真中进攻策略

机器人足球 2D 仿真比赛中，球队的进攻是决定球队整体成绩的标准之一。在过去 15 年的 RoboCup 各项赛事中，进攻优秀的球队往往能用进攻的优势来弥补防守方面的缺陷，并能在比赛中获得较好的名次。足球比赛中进攻包括带球、传球、射门、跑位等动作，在比赛中最常见的进攻策略的研究是分别对进攻动作进行建模和分析，依据经验设置动作触发器，在特定情形和环境下执行特定的动作。这种方式是最简单的，但是必须依据手工编码实现，工作量大；同时由于 2D 仿真状态空间巨大，球员能获得信息不完全等原因[19]，经验和手工编码往往不能很好地提高球队进攻能力。

进攻策略中对球员单个动作进行优化有助于球员在特定情形下有效地执行某一动作，但是当出现更有利的情况时，此球员将无法找到最优方法，而在对单个动作

进行优化的同时又不能保证考虑所有可能存在的更有利情况。如图 3.3 所示，蓝方球队的 7 号球员持球到红方球门附近，按照常规的策略，此时蓝方 7 号球员基本都会选择直接射门，但是红方 1 号球员和 3 号球员有一定概率能够化解蓝方 7 号球员的射门。因此此时最佳的策略应该是蓝方 7 号球员直接传球给 8 号球员，由 8 号球员直接完成射门动作并得分。

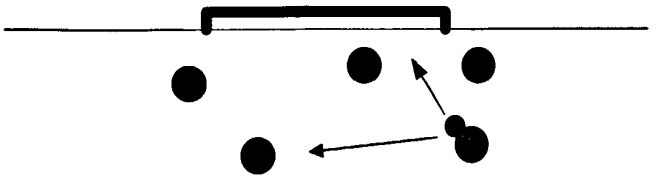


图 3.3 球员进攻

Fig. 3.3 Offensive

同样的情况在比赛中经常出现，这种情况出现的原因就是在球员决策时忽略其他有利的因素而一味对目前的动作进行优化。对对方和己方球员进行建模理论上能够解决这个问题，如图 3.4 所示，蓝方球员 7 号可以通过对对方和己方球员反建模的方式预测他们可能做出的动作及在一定周期内可能到达的位置，如图中所示的花纹型红方 1 号和 3 号球员位置以及花纹型蓝方 8 号球员的位置，正是蓝方 7 号球员预测在最近几个决策周期他们可能到达的位置。此时 7 号球员就可以根据这些反建模的预测来帮助自己在此决策周期的动作选择。这种方法在一定程度上可以解决决策问题出现的单个动作优化的问题。但是在 2D 仿真比赛中，每个周期只有 100 毫秒，在每个周期球员必须完成决策和执行动作两个步骤，因此反建模预测形式很容易让球员陷入计算中而错失最佳的决策和执行时间[10]。

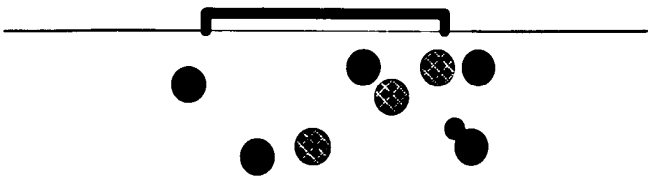


图 3.4 对外界球员反建模预测的进攻策略

Fig. 3.4 Offensive strategy by re-model of other player

综上所述，机器人足球 2D 仿真比赛中，球员必须在有限的时间内做出决策并执行相应的动作，这就是所谓的实时性决策，任何多余的计算都会让球员错失决策周期反而陷入不利的状况。尤其在进攻型球队中，进攻必须快和准，只有这样才能保证球队在比赛中能够始终处于有利的形势。

3.3 基于 MDP 的进攻策略问题建模

第二章已经介绍了 MDP 模型及求解算法，由于简单的模型和求解算法的多样化，MDP 已经成为智能体决策、金融建模以及博弈论的研究上重要的模型。MDP 是对智能体求解出一个从状态到动作的映射，智能体按照这个映射执行相应的动作能够得到最大的累计回报值。

机器人足球 2D 仿真比赛中，由 server 给每个球员发送并更新场面信息，智能体通过这些信息来做出相应的决策。Server 发送给球员的信息包括球的位置和速度以及部分球员的位置与速度等，这些信息正是球员决策所必须的。由于 server 在每个周期会根据球员决策产生的动作更新比赛场面情况，即比赛环境的状态会根据球员决策进行更新。这正好符合 MDP 模型的状态空间转移的要求。

在比赛中，球员都有一系列原子动作如踢球动作 kick、加速动作 dash、转身动作 turn 等，这些动作都是在机器人足球 2D 仿真平台中已经提前设置好，比赛中每个周期只能执行一次原子动作。基于这些原子动作，一些高级动作如带球，就是在多个周期执行多次转身和加速动作而实现[37]。因此根据这些原子动作和高级动作，可以为 2D 仿真球队中球员建立一个动作集，根据球员不同角色执行其相应的动作集合中的动作。

足球比赛球员进攻包括传球、带球、射门、跑位、控球等动作，其中传球、带球、射门和控球都是球员直接在持球状态下的动作，而跑位则是根据场面形势寻找最佳位置接应队友的传球等。持球状态下的动作，可以将球的位置与持球队员的位置进行等同化处理，在一定程度可以降低状态空间。

3.3.1 状态空间表示

状态空间应该包括球员决策时所需的所有外部信息，在持球状态下，球员需要考虑是带球、传球、控球或射门，所有可能影响球员决策的就是当前自己的位置和速度，附近对方球员的位置和速度以及附近己方球员的位置和速度等。由于比赛场面是实时动态的，同时球员从服务器获得的信息是不完全的，因此不可能在每个决策周期得到所有比赛场地球员的信息。服务器会在每个周期对每个球员发送其可以得到的状态信息，里面就包括每个球员在当前状态的位置和速度以及当前状态此球

员可以得到的场面上其他球员和球的位置速度等信息。在机器人足球 2D 仿真平台中，状态 s 描述了每个球员在每个周期可以得到的状态信息，状态 s 涵盖内容用因子化形式表述：

$$s = < x_b, y_b, x_1, y_1, v_1^x, v_1^y, x_2, y_2, v_1^x, v_2^y \cdots > \tag{3.1}$$

其中(x_b, y_b)表示球的位置，(x_1, y_1)表示 1 号球员的位置，(v_1^x, v_1^y)表示 1 号球员的速度。

在持球状态下进攻策略研究中，球员的决策研究是在已经获得球权，因此状态空间的描述可以精简为对球的位置速度和持球方。因此利用因子化描述状态空间为：

$$s = < x_b, y_b, v_b^x, v_b^y, true > \tag{3.2}$$

其中(x_b, y_b)表示球的位置，(v_b^x, v_b^y)表示球的速度，*true* 表示由己方控球，同理对方控球则用 *false* 表示。

3.3.2 动作空间表示

上面已经分析，进攻的持球状态下，球员动作包括带球、传球、控球和射门。每个动作都是由系统提供的原子动作集合生成：

- 带球动作 dribble：带球动作是一系列 kick、turn、dash 原子动作的集合，球员在能够带球的情况下，将带球动作分解为多个原子动作并连续执行，如图 3.5 所示：

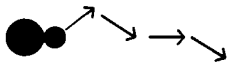


图 3.5 球员带球示意图

Fig. 3.5 Dribble

- 传球动作 pass：传球动作是由原子动作 kick 生成，通过决策过程中计算的角度和力度的不同，传球动作又分为短传、长传和直传球。每种传球都是基于不同的策略：短传是在小范围内的传球，保证球能有效被控于己方，因此需要给球一个比较小的 kick 力度完成；长传是大范围转移，需要考虑球到达目标点的距离和时间，用合理的 kick 动作让球达到一定的速度并有效地传出球；直传球是策略的体现，传球队员要对场面形势进行判断，将球传向某个有利的位置，然后己方球员需要一段时间的追赶才能达到此位置并获得持球权。图 3.6、3.7 和 3.8 中分别为 3 种传球方式：

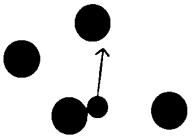


图 3.6 短传方式

Fig. 3.6 Short Passing

- 控球动作 holdball：控球就是球员直接停住球并保护球不被对方球员抢走的动作。控球动作在球员无法直接执行传球、带球等动作的情况下可以有效地继续持有球并寻找更好的机会进攻。

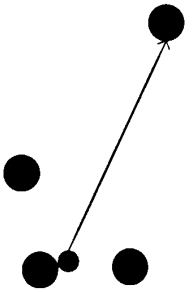


图 3.7 长传方式

Fig. 3.7 Long passing

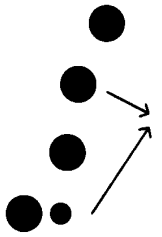


图 3.8 直传方式

Fig. 3.8 Straight passing

- 射门动作 shoot：射门动作是在固定区域执行，球员在判断具有射门可行性的情况下，经过判断找到合适的目标点，并计算相应 kick 的角度和力度，然后执行 kick 动作完成。射门动作与传球动作大致相同，只是在不同的位置应用于不同的场合。

综上所述，所有的动作均可以由原子动作或者原子动作组合而成。即 kick(power, angle)，turn(angle)，dash(power)的形式进行描述。

3.3.3 状态转移函数

在机器人足球 2D 仿真比赛中，每个仿真周期是 100 毫秒，服务器会在每个周

期根据所有球员的决策和动作更新比赛场面信息，即每个球员在此周期应该处于的位置和速度以及球受到外界因素影响应该处于的位置和速度等。在球员做出动作决策后，向服务器发出执行动作命令，服务器就相应地更新场地信息，也就是进入下一个状态[10,25]。在 2D 仿真 server 中会对每个原子动作 kick，dash，turn 等加上随机误差，在执行时候会按照此误差进行；同时，server 还对球员和球的速度向量加上随机误差[19]。因此按照 server 所提供的误差模型可以计算出动作执行后所转移到得后继状态。

3.3.4 回报函数

持球情况下进攻决策，就是从球员获得球权到失去球权的一段时间内的动作决策。球员在这个阶段可以选择带球、传球、控球动作；当处于较好位置能够射门时候，则直接选择射门动作。球员每选择一个动作，则有相应的回报值。在进攻时，要尽量向对方半场推进，因此要尽可能地将球保持在对方的半场并寻找机会进球。回报函数就以球的位置作为参考，球离对方球门中点越近，回报值越大，反之，回报值越小。同时在回报值中要考虑球在己方控制下，回报值为正；若球在对方控制下，则回报值为负。

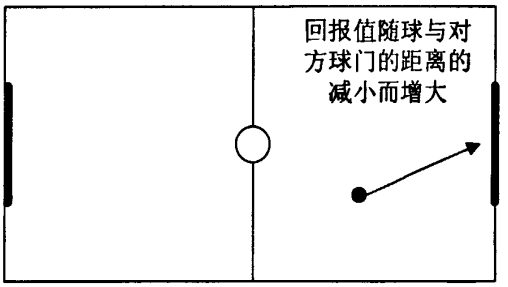


图 3.9 回报函数

Fig. 3.9 Reward Function

综上，用 MDP 模型对持球状态下球员进攻策略的研究进行建模。该问题求解的目标是在己方不丢球的情况下使球员快速进攻并进球。

3.4 持球进攻策略的求解

在第二章中介绍了 MDP 问题求解的几种算法。在基于 MDP 的球员进攻策略研究中，求解的目标是找到最佳动作集，有利于持球状态下球员的进攻。根据回报函

数的定义，策略的目标是要找到在己方球员持球时刻到己方球员失去对球的控制一段时间内的最大累计回报值：

$$V_\pi = max_\pi \sum_{t=1}^{t=T} R_t^\pi \tag{3.3}$$

由于在机器人足球仿真 2D 比赛中，整个系统的状态空间非常大，同时具有实时动态性，球员可以在每个周期进行决策，如果决策时间过长，球员可能会浪费本周期执行动作的时间，因此无法按照预先决策执行相应的动作。如果按照上一节建模方式求解最佳的动作，每个球员在持球状态以最大的速度朝对方球门将球踢出，此时可以获得最大的即时回报值，然后这种决策方式并不能对球队的进攻有所改善，反而一直在浪费进攻机会。在对本问题求解时，需要考虑到长期目标。球队的长期目标是进球，进球就需要寻找射门机会，长久的持球时间则为寻找射门提供了更大的机会。同时求解此问题的累积回报值应该是每个参与进攻的持球队员获得的累积回报值之和，而不是单一球员的累积回报值。因此将本模型下累积回报值分解为每个参与进攻的持球队员所获得的累计回报值之和：

$$V_\pi^* = max_\pi\{V_1^{\pi 1} + V_2^{\pi 2} + \cdots V_n^{\pi n}\} \tag{3.4}$$

其中V_n^π代表在本次进攻中第 n 个球员所获得的累积回报值，表示如下：

$$V_n^{\pi n} = max_{\pi n}[\sum_t R_t^{\pi n}] \tag{3.5}$$

基于 MDP 的持球状态下进攻策略的研究就可以分解为单个球员持球情况下决策问题的研究，求解方法就是在己方球员持球时刻开始，到队友持球的一段时间内，最大化所有进攻球员获得的累积回报值。分解值函数求解的方式仍不能完全避免单个球员追求最大回报值从而一味将球以最大速度踢向对方球门的可能。严格限制射门和长传动作，在理论上可以避免出现如上情况，但是容易让球队陷入“进攻无效”的状态，即球员只会短传和带球，不会把握时机射门并得分，这样的进攻体系是无用的。为解决这个问题，在求解过程中引入动态折扣因子 ζ，对单个球员任意追求最大回报值的情况进行限定，在一定能够程度均衡回报值，定义如下：

$$\zeta = \frac{l_s}{l_o + l_{og}} \tag{3.6}$$

其中l_s表示持球队员与最近队友的距离；l_o表示持球队员与最近对方球员的距离；l_{og}表示持球队员与对方球门的距离。

此时 3.4 式就可以表示为：

$$V_n^{\pi n} = max_{\pi n}[\sum_t \zeta R_t^{\pi n}] \tag{3.7}$$

如上分析，可以用值函数分解迭代的方法求解基于 MDP 的持球状态下进攻决

策的研究。具体算法如下描述：

算法 6：值函数分解的迭代求解方法
1. 初始化持球状态值函数V_π^*与单个球员的值函数$V_n^{\pi n}$ 2. 对持球状态下值函数，有： $V_\pi^* = max_\pi\{V_1^{\pi 1} + V_2^{\pi 2} + \cdots V_n^{\pi n}\}$ 3. 对单个球员的值函数，有 $V_n^{\pi n} = \sum_t \zeta R_t^{\pi n}$ 根据策略π更新单个球员值函数 同时更新持球状态下值函数V_π^* 当$V_{\pi 1}^*$与$V_{\pi 2}^*$无差别时，返回第 4 步 4. 返回最佳策略

3.5 实验及分析

前面已经对持球状态下球员进攻策略进行 MDP 建模，并提出值函数分解的迭代算法，将这种方法应用于 GDUT_TiJi 球队中，并设计一组实验：第一组为 GDUT_TiJi 与机器人足球 2D 仿真 2011 年世界赛冠军中科大 WE 球队进行 50 场比赛；第二组为 GDUT_TiJi 与机器人足球 2D 仿真 2011 年世界赛亚军日本 HELIOS 球队进行 50 场比赛；第三组为 GDUT_TiJi 与其开源底层球队 agent2d 进行 50 场比赛；为做比较，同时设计 agent2d 分别与 WE 和 HELIOS 进行 50 场比赛；然后分别统计出每组比赛中 GDUT_TiJi 的胜率与平均每场进球数，同时统计 agent2d 比赛中的胜率与平均每场进球数，如下表所示(本文所有实验均运行在 ubuntu 10.04 系统下，比赛服务器版本为 soccerserver 15.0.1，硬件环境是 AMD Athlon P360 双核处理器，2.30GHz)：

表 3.1 实验统计数据

Table. 3.1 Statistic of experiments

	WE2011		HELIOS2011		agent2d	
	胜率	平均每场进球数	胜率	平均每场进球数	胜率	平均每场进球数
agent2d	4%	1.54	10%	2.68	X	
GDUT_TiJi	11%	**2.49**	23%	**3.75**	65%	**4.35**

实验证明，改进后的 GDUT_TiJi 在对 WE 的比赛中，胜率由 agent2d 的 4%提升到 11%，平均每场进球数也得到提高；同时 GDUT_TiJi 在对 HELISO 球队的比赛中，胜率由 agent2d 的 10%提升到 23%， 平均每场进球数也得到提高。而在 GDUT_TiJi

与 agent2d 的比赛中，胜率保持在 65%以上。

基于本章的内容，我们的球队 gdut2011 参加了 2011 年 RoboCup 中国公开赛，在第一轮小组赛中对手为：中国的中南大学(CSU_Yunlu)，厦门大学(AmoyNQ)，北京理工大学(ABIT)和伊朗的 Shahid Rajaee 师范大学(Riton)。第一轮小组赛比赛中，gdut2011 以 3 胜 1 负，净胜球 11 个排名小组第一，如图 3.10 所示。


RoboCup ChinaOpen 2011 First Round, Group B

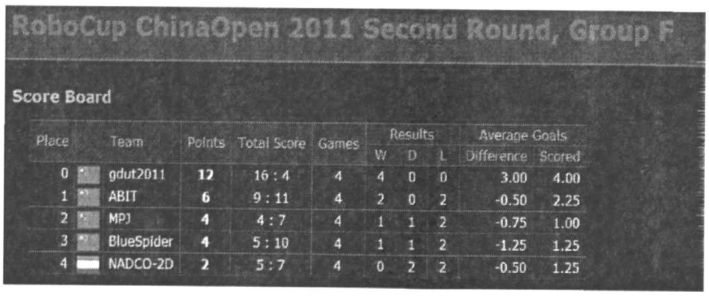
Score Board

Place	Team	Points	Total Score	Games	Results			Average Goals	
					W	D	L	Difference	Scored
0	gdut2011	9	18 : 7	4	3	0	1	2.75	4.50
1	CSU_Yunlu	9	16 : 11	4	3	0	1	1.25	4.00
2	AmoyNQ	9	5 : 3	4	3	0	1	0.50	1.25
3	Riton	1	2 : 8	4	0	1	3	-1.50	0.50
4	ABIT	1	5 : 17	4	0	1	3	-3.00	1.25

图 3.10 2011 中国公开赛小组赛第一轮

Fig. 3.10 First Round group game during RoboCup China Open 2011

第二轮小组赛中，我们的对手为：中国的北京理工大学(ABIT)，安徽滁州学院(MPJ)，西安交大(BlueSpider)和伊朗的北德黑兰阿扎德大学(NADCO-2D)。gdut2011 凭借良好的进攻体系，以 4 场全胜，净胜球 12 个的成绩继续排名第二轮小组赛第一，并顺利进入 RoboCup 中国公开赛八强的成绩，如图 3.11 所示。


RoboCup ChinaOpen 2011 Second Round, Group F

Score Board

Place	Team	Points	Total Score	Games	Results			Average Goals	
					W	D	L	Difference	Scored
0	gdut2011	12	16 : 4	4	4	0	0	3.00	4.00
1	ABIT	6	9 : 11	4	2	0	2	-0.50	2.25
2	MPJ	4	4 : 7	4	1	1	2	-0.75	1.00
3	BlueSpider	4	5 : 10	4	1	1	2	-1.25	1.25
4	NADCO-2D	2	5 : 7	4	0	2	2	-0.50	1.25

图 3.11 2011 中国公开赛小组赛第二轮

Fig. 3.13 Second Round group game during RoboCup China Open 2011

最终 gdut2011 在 2011 年 RoboCup 中国公开赛 2D 仿真比赛中获得全国第四名的好成绩。这也证明我们在本章所利用的模型及求解算法在实际比赛中能够有效提高球队的进攻能力。

3.6 小结

本章介绍的是利用 MDP 模型为机器人足球 2D 仿真球队中进攻策略进行建模和求解的方法。首先对机器人足球 2D 仿真球队决策体系进行分析，然后针对进攻策

略存在问题进行探讨，并提出利用 MDP 模型对进攻决策改进的可能性；建模过程是解决问题的第一步，通过大量的分析，本章以球的位置构建状态空间，将带球、传球、控球和射门作为球员基本动作，并以球员执行动作后球的位置作为回报函数进行建模。本章就 MDP 模型的求解提出了值函数分解迭代求解算法，并在 GDUT_TiJi 球队中实现基于 MDP 模型的持球球员进攻决策体系。文中设计几组改进后的球队 GDUT_TiJi 和未改进球队 agent2d 分别 2011 年世界赛冠军 WE 球队和亚军 HELIOS 球队的比赛，通过比赛数据的统计和分析表明经过改进的球队整体在进攻方面得到一定的提高。

基于本章工作实现的球队 gdut2011，代表本校参加 2011 年 RoboCup 中国公开赛 2D 仿真比赛，获得了全国一等奖的好成绩，是本校机器人足球仿真 2D 比赛参赛以来最好的成绩。

第四章 基于 POMDP 的守门员策略研究

本章介绍利用 POMDP 模型对机器人足球 2D 仿真球队的守门员策略进行改善。首先分析机器人足球 2D 仿真球队中常规守门员策略的设计和存在的不足，然后分析利用 POMDP 为守门策略建模的可行性和意义，并对守门员决策进行建模，通过建模提出针对守门员策略特定情况的算法，最终在 GDUT_TiJi 球队中实现并通过试验验证其有效性。

4.1 机器人足球 2D 仿真守门员决策

守门员是一类特殊角色的球员，其动作与普通球员有所不同，除了 kick、turn、dash 动作外，守门员在禁区范围内还有 catch 动作，也就是扑球动作，这是守门员角色独特的动作，也必须是守门员在禁区的范围内才能使用。

守门员决策就是当对方球员进攻时，实时地做出判断并与自己球队防守球员一起化解对方的进攻，这个过程中需要跑到合适的位置，用扑球或者踢球动作获得球权或者将球踢出禁区。对方球员的进攻主要包括正常进攻、对方角球、对方任意球等情况，每种情况下，对方都会有几名进攻球员随机分布在禁区附近的区域，主动寻找或等待机会发起进攻，因此守门员必须迅速准确地做出判断。

在机器人足球比赛中，合理地选择动作是非常重要的，尤其对守门员这类特殊又重要的角色，动作的选择稍有闪失就可能会造成严重的后果。守门员决策和普通球员的决策是类似的，首先从比赛服务器 server 获得外界环境的信息，然后转换为内部状态表示的信息，再通过内部的动作选择模块确定具体的动作，动作执行模块就将相应的动作描述发送给 server，由 server 实时地执行并改变外界环境[19,25]。如图 4.1 所示。

守门员策略动作选择的方法主要有基于规则的方法和基于目标的决策等。基于规则的方法就是由球队开发者提前设计好条件-动作规则(if-then 规则)，与当前场景进行合适的匹配，并选择相应的动作。采用基于规则的方法进行动作选择，就要尽量地扩大 if-then 规则库，利用经验或理论分析对更多场景进行描述。毫无疑问，这种方法比较简单，执行速度也比较快，但是建立规则的过程就很复杂，尤其涉及到对场景等分析；同时要想达到预期效果，规则数量将及其庞大[25]。

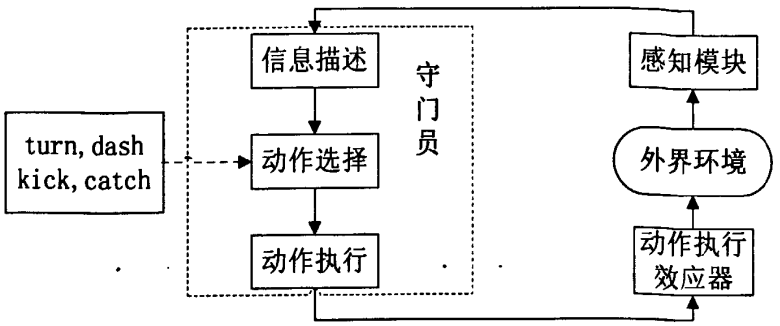


图 4.1 守门员决策基本模型

Fig. 4.1 Simple Model of goalie decision

基于目标的决策，就是在球员决策过程中，引入目标，并通过一定的推理进行动作选择，可以让球员的决策更理性。为达成这个目标，球员往往需要需要执行一个动作序列[25]。守门员策略中，最主要的目的就是防止对方进球，因此守门员每个决策都是按照防止对方进球作为指导。在执行中，通过对长远目标的分解，可以获得当前能够达成的目标，从而逐渐地实现长期的目标。例如：守门员防守过程中，目标是防止对方进球，假如对方球员正在实行“单刀”的进攻方式，守门员应该尽量向前进行阻止，对长远目标分解就是将守门员向前阻止对方进球分解为更小目标：首先是转身到对方球员的方向；然后加速跑位某个位置；接着根据与球和对方球员的位置执行踢球或扑球的动作。在基于目标的决策中，最重要是如何将长远的目标分解为小目标，并利用一定规划的方式实现这些目标。规划的算法是多种多样的，但是针对大规模的问题，规划的效率会比较低，实时性也差，因此这种方法很难在实时性要求高的系统中有效地解决问题。

4.2 守门员决策存在问题与分析

守门员决策是机器人足球比赛中的重要部分，守门员的意义是保证球队尽量少被进球，从而保证球队能够有机会获得胜利。在 4.1 中介绍了几种守门员决策的方法，基于规则的方法执行效率高，但是规则的确定很难，同时其表现很大程度依赖于规则库的规模；基于目标的规划方法理论上有较好的效果，但是建模和规划的过程复杂，实现效果也不佳。

在机器人足球比赛中，守门员可以执行的动作有 kick、turn、dash、catch 等，当己方球员进攻时，守门员不需要做太多的决策，而只是在固定的位置“观望”场

面形势，如果是对方球员进攻，守门员就必须实时合理地选择相应的动作以避免对方进球。当对方进攻时，守门员的决策必须快和准，否则很容易错失良机被对方进球，这也对球队开发者提出了更高的要求。因此，守门员决策中存在的问题主要有[19,25]：

- 环境复杂：对方进攻时，往往有数个球员共同参与进攻；同时己方防守球员也混入其中，导致决策环境非常复杂。如果对某一重要外界信息的忽略，可能会导致对方抓住机会进球。
- 决策时间短：当对方已经将球带入禁区附近，也被称为是最危险的区域时，守门员必须迅速做出判断，因为对方进攻球员往往只需要几个仿真周期就能完成进攻。这就要求守门员必须在尽可能短的时间内做出决策。
- 决策要求高：守门员的决策要求要比普通球员要求高，因为一次的疏忽可能导致整个场面形势的逆转或者失控，而守门员的决策与普通球员决策的区别还在于：普通球员决策失败还有挽回的余地，但是守门员决策失败就会导致被进球。

通过上面的分析，我们希望能够找到一种合理并有效的守门员决策，能够在较短的时间内帮助守门员做出最准确合理的决策。

机器人足球比赛中，守门员并不会在所有周期都进行决策，一般是在对方发起进攻时，守门员开始根据场面形势进行决策。守门员决策一定程度上依赖于球的位置和对方球员的位置，当球与对方球员进入己方半场，守门员才开始进入决策状态。同时，根据球和对方球员的位置可将己方半场进行划分，并标注区域的危险程度。

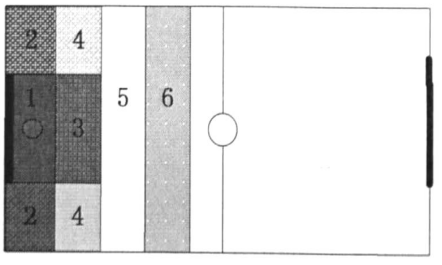


图 4.2 守门员危险区域划分

Fig. 4.2 Partition of dangerous areas by Goalie

如图 4.2 所示，蓝色为守门员，己方半场进行区域划分，按照颜色进行编号，其中红色 1 号区域为最危险的区域，浅红色 2 号区域为次危险区域，接下来是 3、4、5、6 等区域，危险程度依次递减。对多场比赛分析发现，最危险的 1、2、3 区域，

是对方球员进攻最容易进球的区域，因此守门员决策必须对这些区域内的情况进行综合处理。

球和对方球员所处的位置是影响守门员决策的重要因素，在上面已经对这些区域进行了分析，当球与对方球员刚好处于危险区域，守门员就必须适时地做出准确的决策，从而防止对方球员的进攻，而此时一个重要的问题是多名球员聚集在危险区域，守门员如何应对才是对自己球队最有利的。如图 4.3 所示，黄色 1 号球员为红方的守门员，蓝方 7 号球员持球；此时守门员必须迅速根据场面形势决定是路线 1 所示防止蓝方 7 号球员射门还是按照路线 2 对蓝方 8 号球员进行防守，以防蓝方 7 号球员将球传给 8 号球员造成一个有威胁的射门。

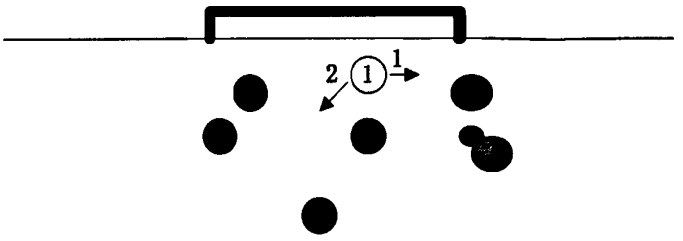


图 4.3 守门员决策

Fig. 4.3 Decision of Goalie

因此，在守门员决策中，尤其是当球处于危险区域，守门员决策必须要充分考虑场面上可能构成威胁的情况，通过综合权衡，判断出最有利的动作并执行。而在机器人足球 2D 仿真比赛服务器 server 中，对球员的视觉距离和范围都有一定的限制，同时为保证和真实比赛环境相似，服务器 server 还对每种动作加上随机误差，这些原因也给守门员决策带来了极大的挑战。尤其是在危险区域的守门员决策，稍有闪失就能给对方带来极好的射门机会从而失去对比赛的控制。在守门员决策短短的几个仿真周期内，守门员不能单一对某种形势进行分析判断，如图 4.3 中描述的，假如守门员只注意到蓝方 7 号球员持球，而没发现蓝方 8 号球员的位置，从而单一判断需要对 7 号球员进行布防，此时 7 号球员改变进攻策略直接传球给 8 号球员，那么 8 号球员就获得了非常好的射门机会；这种情况下守门员就很难对 8 号球员进行防守，也就错失良机。

为对方球员的运动进行建模的方法在一定程度可以解决上面描述的守门员决策的问题，这种方法需要在危险区域为每个对方的球员进行建模，预测其可能的运动并做出相应判断。利用这种方式，守门员可以及早地为对方球员建模并预测其可能的位置和动作，在危险的情况下做出最合理的判断，尤其是受到服务器限制，当前

周期只能获得部分外界环境信息的情况下，为对方球员的运动进行建模的方法就能尽可能地为守门员提供全面的信息，如图 4.4 所示。为对方球员建模的方式理论上可以解决守门员决策中综合分析的问题，但是也存在很多不足。首先建模过程的复杂性，在建模过程中，要适时地分析对方球员可能的动作和情况并进行保存；其次是环境的不确定性，在为对方球员建模的过程中，环境的不确定性可能导致建模的误差，这样可能为守门员决策提供带有太大误差的信息，将使守门员决策失误；再次是维护信息庞大，在为对方球员建模的过程中，可能需要维护多名对方球员的信息，而且也可能要维护多个周期内对方球员的可能的动作，因此建模中可能要维护比较庞大的对方球员信息。另外一个不足就是，当维护信息过于庞大，守门员决策中会出现大量无用信息，在处理这些信息时候会耗掉守门员大量决策时间，对实时性要求比较高的守门员决策是不利的。综上，为对方球员建模的方法理论上能改善守门员决策，但是实施起来比较复杂，而且会在每个阶段累计放大随机误差导致守门员的误决策。

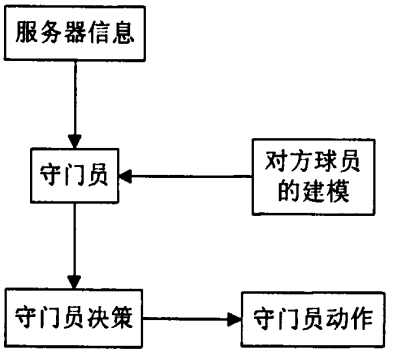


图 4.4 守门员为对方球员建模的决策

Fig. 4.4 Goalie Decision by re-model of opponents

综上所述，在机器人足球仿真比赛中，由于特殊的角色和动作，守门员决策体系与普通球员的决策体系稍有区别，而守门员决策即防守的过程中，环境复杂、影响因素多以及不可能获得全局信息等原因也给守门员决策带来了更高的挑战。在本文第二章介绍了 POMDP，即部分可观马尔科夫决策过程的模型和求解算法，POMDP 是非确定环境下具有局部外界环境信息的智能体决策的较好模型，能够在不完整的外界信息下为智能体的决策提供新的求解方式。因此本章将利用 POMDP 模型为机器人足球仿真比赛中守门员决策建模并进行求解。

4.3 基于 POMDP 模型的守门员决策研究

基于 POMDP 模型的守门员决策研究，就是将 POMDP 模型引入到守门员决策模块，利用 POMDP 为守门员决策进行建模并求解，求解结果为守门员动作序列，此时守门员只需要按照这个动作序列执行。决策和对决策的评价过程都被设置在守门员决策模块。因此引入 POMDP 模型的决策结构如图 4.5 所示：

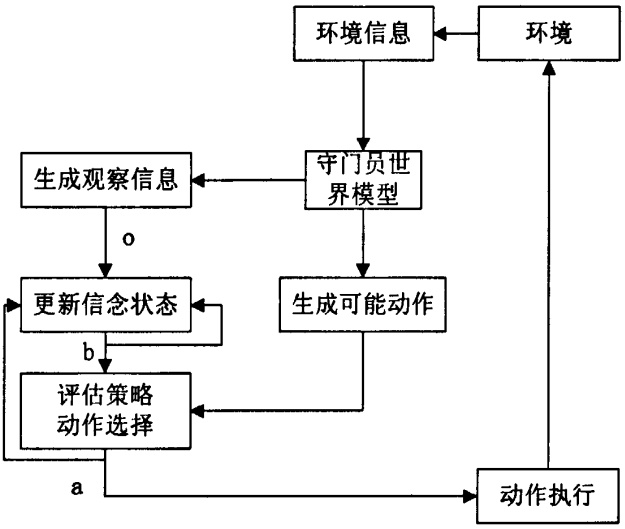


图 4.5 基于 POMDP 的守门员决策体系

Fig. 4.5 Goalie Decision system based on POMDP

在第二章分析中我们已经知道在 MDP 的求解中，得到的是最佳策略，表示为状态到动作的映射，智能体按照这个策略执行动作，就能得到最大期望回报累计值；而在 POMDP 模型中，由于外界环境状态是不完全客观的，并无法完全准确地对状态进行描述，因此引入一个信念的概念，所谓信念就是智能体根据历史动作和状态信息推测出目前所应该处于的状态，是一个概率分布。在 POMDP 求解过程中，同样得到一个最佳策略，用信念到动作的映射来描述，智能体同样按照策略执行相应动作时，能够得到最大期望回报累计值。利用 POMDP 为守门决策进行研究，就要先建模，然后确定求解方法，下面将对建模和求解过程进行解释。

4.3.1 状态空间表示

在机器人足球仿真比赛环境中，场地共有 22 名球员和 1 个足球，而每个球员和足球都有位置和速度信息，如果用因子化方式描述状态信息，则每个周期所更新的状态空间就非常庞大，此时所维护的状态空间中信息并不是都能为球员决策起作用，

同时因为服务器的设置，每个球员并不可能完全准确地获得服务器所维护的场面信息，因此对状态空间的表示要根据实际的情况进行选择。

影响守门员决策的主要因素是球的位置，对方球员的位置以及对方球员是否持有球等，然而距离守门员太远的信息一方面由于服务器设置无法准确发送到守门员，另一方面距离守门员太远的信息对守门员决策基本没有影响，因此这种信息研究价值不大。在机器人足球 2D 仿真平台中，球员在某个时刻以最大的速度踢足球，然后足球在不受到任何干扰的情况下可以运行大概 30 米，因此距离守门员超过 30 米距离的足球运行对守门员并不会造成威胁[19,37]。在守门员决策研究中，距离守门员距离超过 30 米的信息则可以被守门员忽略，那么在本问题的模型中，首先所构造的状态空间就是守门员的位置与距离守门员距离 30 米以内足球与对方球员的位置信息，表示如下：

$$s = < x_g, y_g, x_b, y_b, x_1, y_1, \cdots x_n, y_n, true > \tag{4.1}$$

其中$< x_g, y_g >$表示守门员的位置,$< x_b, y_b >$表示球的位置，$< x_n, y_n >$表示对方第 n 个球员的位置，*true* 表示对方球员持球。

在真实比赛中，由于服务器的设置，守门员并不能随时准确地得到所有球员的位置信息，因此在本问题求解中仍将球和守门员的位置作为最重要的状态空间变量，同时只考虑在距离守门员 30 米以内的球的位置信息，此时状态空间就描述为：

$$s = < x_g, y_g, x_b, y_b > \tag{4.2}$$

4.3.2 动作空间表示

在守门员决策中，可以根据守门员动作的只是和具体的状态生成有限的可选动作集合，守门员动作如下：

MovetoPoint：跑位，守门员要根据实际情况跑到某个位置；

CatchBall：扑球，守门员执行 catch 动作以获得球的控制权；

KickBall：踢球，守门员在开球或者不能 catch 球时候要执行的防守动作。

守门员的基本动作就如上面所介绍，一般情况下是如果守门员能扑球则执行 CatchBall 动作；如不能则判断是够能够踢球，若能踢球则执行 KickBall 动作；如不能则根据跑位策略跑到相应的点。

4.3.3 状态转移函数

守门员决策的研究中，状态描述的是守门员的位置和球的位置，守门员的位置仍然可以根据服务器各项随机误差进行建模；而球的位置就要根据对方球员的动作而变化，然而对方球员的动作并无法完全预估，因此对球的状态变化建模就很难[19]。

在图 4.6 中演示了智能机器人格子寻宝的例子[39]，黑色为智能机器人，红色位置为宝藏的位置，机器人的目标就是通过自己的判断，每一步运动一个格子，从而在最短的时间内到达红色宝藏的位置。在图中所示的机器人的位置，下一步可以到达的位置为标号为 1，2，3，4，5 的格子，以机器人到达的格子与目标的距离来判断：当机器人下一周期到达 1 号格子，则到达目标位置最少还需要移动 3 步；当机器人下一周期到达 2 号格子，则到达目标位置最少还需要移动 4 步；同理，到达 3 号格子还需要最少移动 5 步；到达 4 号格子还需要最少一定 6 步；到达 5 号格子还需要最少 4 步。因此可以相应地为机器人状态转移函数进行建模，假设机器人下一周期到达 1 号格子的概率为 40%；到达 2 号格子和 5 号格子的概率均为 20%；到达 3 号格子的概率为 15%；到达 4 号格子的概率为 5%。

在足球比赛中，持球的队员在进攻时，会尽量考虑向对方的球门处运动，根据上面分析，我们可以用估测的方式为球的位置转移函数进行类似建模。对状态空间的守门员位置转移函数和球的位置转移函数分别建模后就可以统一得到此问题求解中的状态转移函数。

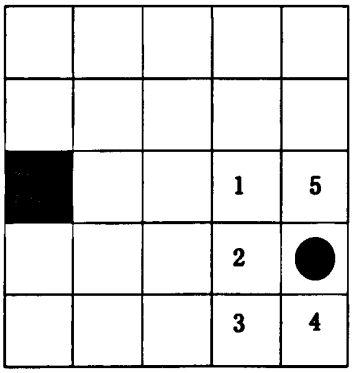


图 4.6 状态转移模型

Fig. 4.6 State Transfer function model

4.3.4 观察集合

比赛中，对方球员的位置也会对守门员决策产生影响，同时对方球员的位置并不能随时准确地发送给守门员，需要守门员自行进行观察，因此观察值即对方球员的位置，根据上面的介绍，同样是距离守门员 30 米以内的对方球员的位置会对守门员决策产生极大影响。观察值就描述为在距离守门员 30 米范围内的对方球员的位置信息。

同理，观察集合根据定义，就表示为守门员所能得到的观察的有限集合。

4.3.5 观察函数

观察函数是对智能体在给定动作和结果状态下可能得到的观察的概率分布，在本问题中，守门员可以做出任意一个动作，根据选定的动作，状态可以根据状态转移函数转换到下一个状态，此时守门员可以得到的观察值的概率就被称为本问题中的观察函数。建模方法与状态转移函数相同。

4.3.6 信念空间

信念是描述智能体当前处于状态的概率，用$b(s)$描述当前智能体处于 s 状态的概率，这个概念是基于非确定环境下决策所提出的，是根据智能体的观察和历史信息计算得到了目前可能处于状态的概率分布。在守门员决策问题中，由于存在大量不确定因素，因此引入信念状态的概念，通过式(2.12)和(2.13)计算可以求得。

4.3.7 回报函数

与 MDP 问题一样，模型中会先设定回报函数，对守门员每一次决策进行评价，并返回相应的回报值，求解的目标就是求出守门员决策的一系列动作，通过执行这一系列的动作能够得到最大期望累计回报值。但是守门员决策中，环境情况比较复杂，同时需要根据球的状态和守门员的动作联合确定回报函数，因此回报函数描述为下：

- 守门员执行 MovetoPoint 动作时，则守门员与球的距离越近，回报值越大，记为：$R = \frac{30}{L_{bg}}$，其中L_{bg}表示守门员与球的距离。

- 守门员执行 CatchBall 动作时，若守门员成功获得球，回报值为 30，否则回报值为-30。
- 守门员执行 KickBall 动作时，若成功将球踢出禁区，回报值为 20，否则回报值为-20。
- 除了以上情况外，无论守门员执行任何动作，如果球的位置在己方球门位置，即对方进球，回报值为-100。

4.3.8 策略求解

在 MDP 问题求解中，策略通常用状态到动作的映射来表示；POMDP 模型中，智能体的决策通常用策略树表示，智能体选择一个动作后，通过观察，得到相应的信念值，然后再选择动作，并继续下去。因此 POMDP 模型求解中的策略通常被描述为信念到动作的映射。智能体每执行一次动作，就可以根据回报函数获得一个回报值，假若智能体按照某策略执行，则可以获得期望累计回报值，也叫期望回报的值函数，求解过程也就是寻找最佳的策略，智能体沿着这个最佳策略执行动作时候能够获得最大期望回报的值函数。

POMDP 求解的算法在第二章已经进行了介绍，其求解方法主要有值迭代方式和策略迭代方式。值迭代的方式是初始化值函数，然后通过动作的一步步迭代，最大化值函数直至值函数稳定，然后返回相应的策略，即动作序列。值迭代的方式是寻求最佳策略比较优秀的方式[12]，因此在本章介绍的基于 POMDP 模型的守门员决策也采用值迭代的方式进行求解。首先初始化值函数，然后通过对动作的一步步迭代，最大化值函数，直到值函数趋于稳定，此时所选择的动作序列就为最佳策略。值迭代的方式是比较简单和有效的方式，但也存在一定的不足：问题求解的时间是本模型求解过程中最大的困难。

4.4 基于临界状态的 POMDP 求解方法

在 4.3 节介绍了利用 POMDP 为守门员决策建模的方法，但是在求解上，常规的值迭代方法可能因为运算时间过长让求解无法顺利完成。实时的 POMDP 求解方法要求在求解的过程中能够随时开始和结束，并且能够在尽可能短的时间完成。由于 POMDP 的策略是以策略树的形式生成，策略树的规模将直接影响求解的时间；

同时在策略树上搜寻最佳策略的时候，往往是游历所有的节点，然后根据一定的算法和规则得到相应的最佳策略，搜寻过程中存在的一个情况是，智能体可能在还未对策略树进行完整搜寻的时候已经得到了一个最佳策略，此时完整的搜寻对整个决策并无实质意义，同时进入了冗余的决策时间。

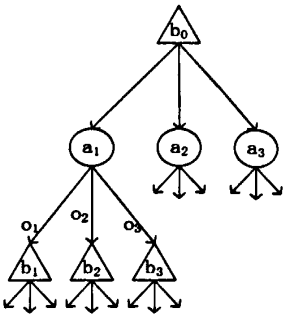


图 4.7 POMDP 上策略树

Fig. 4.7 Decision Tree of POMDP

为解决智能体实时决策过程中的冗余决策时间问题，我们提出一种基于临界状态的 POMDP 模型，利用临界状态的思想，降低模型求解过程中搜寻时间。所谓临界状态，就是智能体在决策过程中能够停止求解的状态。比如守门员决策中，守门员在已经构建好的策略树上搜寻最佳策略，假若已经在某个策略上得到守门员能够扑得球，那么这个策略可以被作为最佳策略，此时守门员可以直接停止搜寻过程从而直接按照这个最佳策略执行相应的动作。利用临界状态的方式对基于 POMDP 模型的实时智能体决策进行求解，在一定程度可以降低决策时间，尤其当构建的策略树规模比较大的情况下，这种方式能够有效保证智能体及时迅速地对当前的状态进行决策。求解的过程如图所示：

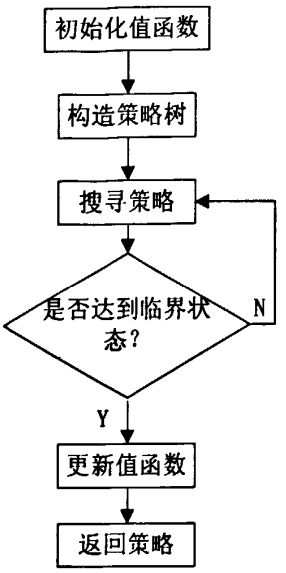


图 4.8 基于临界状态的 POMDP 求解方法

Fig. 4.8 POMDP solving method based on the critical state

基于临界状态的求解方法是求解 POMDP 问题中大规模策略树的有效方法，当策略树规模不是很大的时候，算法能够在有限的时间内求出相应的最佳策略；基于临界状态的方法主要思想是在大规模策略树中搜寻最佳策略树时避免产生太多无关的策略从而影响智能体决策，这种方法一定程度依赖于策略树的规模，如果策略树规模较小，可能在策略树中并没有生成相关的临界状态，因此仅用简单的算法就可以完成最佳策略搜寻。

4.5 实验及分析

在本章前面几节分别分析了机器人足球 2D 仿真比赛中守门员决策存在的问题，然后分析利用 POMDP 模型为守门员决策进行建模和求解的可行性，针对建模和求解过程又提出基于临界状态的求解方法，这种方法在理论上能够降低大规模策略树中搜寻最佳策略的时间。

为验证本章提出的方法，我们设置两组对比试验，第一组为 agent2d 球队与 2011 年世界冠军球队中科大 WE 队；第二组为 agent2d-G 与 2011 年世界冠军球队 WE 队。WE 球队连续多年蝉联机器人足球 2D 仿真比赛冠军和亚军，属于进攻非常犀利的球队，因此选用此球队作为标准测试球队，主要测试守门员在真实比赛中的表现，并用平均失球数作为测试标准。为排除比赛中的可能存在的随机误差，分别对每组实验进行 50 场比赛，并统计每个球队的平均进球数，然后以 WE 球队的进球数分别作

为 agent2d 和 agent2d-G 球队的平均失球数。

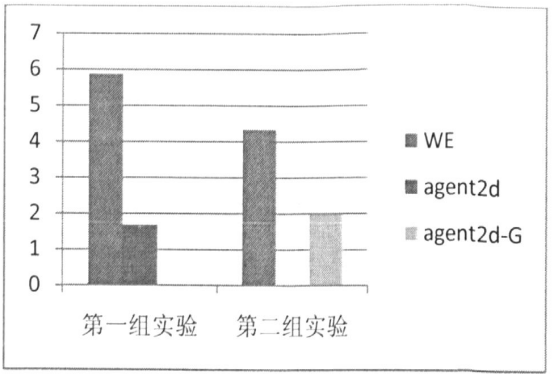


图 4.9 平均失球数统计

Fig. 4.9 Statistic of average losing ball

在实验中，分别进行了 50 场比赛并统计每个球队的平均进球数，通过对比发现 agent2d 的平均失球数为 5.88，而改进后的 agent2d-G 的平均失球数为 4.34，表明了本章提到的守门员决策方法一定程度上降低了球队的失球率；同时发现 agent2d-G 球队的平均进球率也比 agent2d 球队得到提高，从 1.68 上升到 1.97，这也说明良好的守门员决策不仅可以降低球队的失球率，一定程度也增加了进攻的机会。

4.6 小结

本章的主要内容是用 POMDP 模型为机器人足球 2D 仿真球队的守门员决策进行改进。首先分析当前守门员决策方法存在的问题，并分析利用 POMDP 模型为守门员决策进行建模的可行性，然后实时地根据守门员决策的问题进行建模。由于守门员经常要在一些紧急情况下进行决策，求解过程必须降低守门员的决策时间，因此文中提出基于临界状态的求解方式，并应用于守门员决策的过程。

为检验本章提出的模型和求解方法的有效性，文中设置了两组比赛进行对比，通过比赛数据的统计验证了利用 POMDP 建模的守门员决策及基于临界状态的求解方式能够有效地提高守门员成功扑球的概率。

第五章 多智能体系统决策问题研究

前面两章主要介绍了在机器人足球 2D 仿真球队智能体决策问题的研究，都是对单个智能体的决策进行建模和求解。在现实问题中，决策问题往往存在于多个智能体之间，这也就是多智能体系统决策的问题。本章关注的重点是多智能体系统决策问题的研究。

5.1 多智能体系统决策

多智能体系统(Multi-agent System)通常是指多个智能体联合起来的决策系统，系统中的每一个智能体都是一个独立的决策个体，通过从环境中获得的信息自主地进行决策[12]。每个智能体独立地对环境实施一个动作，而整个系统的状态则受到所有智能体联合动作的影响。因此，每个智能体在决策的时候都必须考虑其他智能体可能采取的动作，以及这个可能采取的动作对自己决策的影响。例如著名的囚徒问题(Prisoner’s Dilemma)[12]中，可以将每个参与决策的囚徒作为智能体，每个囚徒可以选择的动作是认罪或者沉默。对于甲囚徒来说，如果他与乙囚徒都认罪的话，那么两个人都服刑两年；如果他与乙囚徒都沉默的话，那么两个人同时服刑半年；如果他认罪而乙囚徒沉默的话，那么他被无罪释放而乙囚徒服刑十年；反之如果他沉默而乙囚徒认罪的话，那么他就要服刑十年而乙囚徒被无罪被释放。这个问题就可以被认为是典型的多智能体决策问题，每个参与的智能体都想获得最大化的收益，然而在其决策过程中又必须要考虑对方的可能做出的决策。

囚徒问题是一个简单的多智能体直接对抗的问题，每个智能体的策略规模不大，而且智能体执行任何动作都会直接到达下一个状态，不涉及到环境的不确定性，因此求解过程并不复杂。但在真实生活中，多智能体系统决策经常要在非确定性环境下进行，因此就对建模和求解过程提出新的挑战。

移动机器人决策是一种典型的智能体结构，机器人通过摄像头从外界获得信息，然后通过自身的决策模块进行策略规划，并通过执行模块执行规划好的动作；然而机器人摄像头捕捉的外界信息可能因为传输过程延误出现信息不完全的情况，同时机器人动作执行部件也可能会出现动作执行不完整的情况，这些情况为移动机器人

决策引入了极大的不确定性[11,12]。月球漫游车[39]就是一种典型的移动机器人实例，当有多个月球漫游车在月球上共同执行岩石采集等任务的时候，每个漫游车都是一个移动机器人，在单独决策的时候具有极大的不确定性；如果需要多个漫游车共同合作完成任务，则规划的过程就更复杂，因为不仅要对环境的不确定性进行建模，在规划过程中还需要考虑基于对方动作的决策。在前面两章我们介绍了针对单智能体非确定环境下决策的模型，MDP 专注于外界环境完全可以获取的情况下的智能体决策；而 POMDP 则以非完全可观的环境信息为智能体决策提供新的模型。

多智能体系统有合作决策和对抗决策两种，在合作决策中，整个系统可以被认为是一个团体，即所有的智能体都享有同样的回报函数，抱着一个共同的目标去规划和决策；而对抗决策中，不同智能体有不同的目标，其回报函数等也会根据实际情况有所差别，本文关注的是多智能体合作策略的研究，也就是享有共同目标的多智能体策略。多智能体的马氏决策过程(MMDP)[40]为多智能体系统合作策略研究提供了较好的模型，MMDP 与 MDP 在模型定义上的唯一区别就是用多智能体的联合动作代替了在 MDP 模型中的动作，同时系统中的状态信息是完全可以被所有智能体共享地获得[12]。因此在问题求解上，MMDP 与 MDP 比较类似，只是其状态数和动作数等随着智能体个数的增加呈指数级增加。 但是 MMDP 要求在决策的每个周期内，所有的智能体都能完全得到环境的状态信息，这个前提在现实问题中并无法得到满足。多智能体系统的一个研究意义是分布式的信息获取和决策，也就是智能体只需要根据自己所获得的部分信息进行决策，而决策必须要考虑整个团队。分布式部分可观马尔科夫决策过程(DEC-POMDP)[12]正是基于现实问题，利用部分可观察环境状态信息求解多智能体系统决策的模型，也是对马氏决策理论中 POMDP 模型在多智能体协作问题上的扩展。因此 DEC-POMDP 模型及相关求解方法是本章关注的重点。

5.2 DEC-POMDP 模型与求解

多智能体老虎问题(multi-agent tiger problem)[41]是 DEC-POMDP 模型研究中最常见的问题，这个问题中，有一个密室，密室有两道门，一道门后面是珠宝，另外一道门后面是老虎；密室里面有两个机器人，各自能够通过自己的机械手打开门，同时机器人带有专门的设备用于监听是否有老虎的声音；两个机器人之间是不允许交

流通讯的，即相互之间不允许交换信息。他们的行为被认为是分布合作的：如果其中一个机器人独自打开有老虎的一道门则整个团体将会受到很大的惩罚；如果两个机器人同时打开有老虎的一道门则获得的惩罚相对较小；如果机器人打开有珠宝的一道门则会获得一定的收益，回报函数的设置一定程度上鼓励两个机器人之间展开合作，但是机器人之间获得的信息并不一致，同时两者不能交换信息，因此合作还是很困难的。这个问题可以用 DEC-POMDP 描述为：1. 系统的状态是指老虎在左边门和老虎在右边门，即表示为：S={tiger-left，tiger-right}；2.机器人的动作，分别为打开左边门，打开右边门和监听老虎声音，即表示为：A={open-left，open-right，listen}；3.机器人的观察值，分别为听到老虎声音在左边门，听到老虎声音在右边门和没有老虎声音，即表示为 O={roar-left, roar-right, none}。问题开始时候，老虎和珠宝被随机地放在两道门后面，机器人不是从系统中直接获得状态，而是获得观察值，这个观察值一定程度上反映了当前系统所处的状态，但是本身是具有不确定性的，因此用观察函数对这个不确定因素进行建模。初始状态对机器人来说，老虎处于两道门后面的可能性是等概率的，每个机器人都需要执行监听动作，从而判断老虎在某道门后面的可能性最大，从而打开这道门，以最大的概率去获得珠宝[12]。

5.2.1 DEC-POMDP 求解思路

在第二章的马氏决策理论中我们已经简单介绍了 DEC-POMDP 模型的构成，与 POMDP 最大的区别就是：在 DEC-POMDP 中分别用联合动作、联合状态、联合信念和联合观察取代在 POMDP 中的动作、状态、信念和观察等。在任意决策周期，每个智能体根据自己获取的部分信息进行决策并执行一个动作，系统会根据所有智能体的联合动作转移到下一个状态，同时系统进入下一个决策周期。根据系统中回报函数的定义，每个决策周期，系统会根据联合动作返回一个相应的回报值。整个过程的最优策略就是找出一个能够最大化系统累计回报值的策略。

策略树仍然是 DEC-POMDP 模型中常用的策略表示方式，每个智能体的策略可以表示为一个独立的策略树，而联合策略就是所有智能体的策略树的集合。在策略树中，树的节点表示为将要执行的动作，树的每一条边则表示为智能体执行动作后可能获得观察值，树的深度与总共决策的时间 T 是一致的，图 5.1 为两个智能体的策略树示意图。DEC-POMDP 问题中，策略树是智能体决策的体现，智能体执行过

程中就是按照策略树一步一步地执行相应的动作，求解的过程就是计算出策略树的过程，当策略树被计算出来，智能体的执行是非常简单的，只是一些简单的分支转移[12]。因此求解过程中最重要的就是如何构建出最合理的策略树。

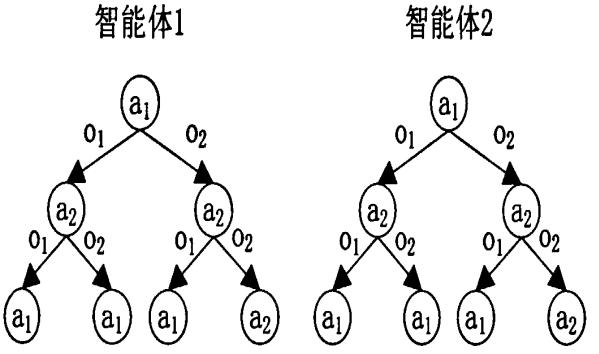


图 5.1 两个智能体的策略树示意图

Fig. 5.1 Decision-tree of two agents

DEC-POMDP 模型是解决多智能体系统决策的有效模型，通常是对多个智能体的决策进行规划，如图 5.1，对多个智能体的决策分别构建策略树，并通过回报函数衡量相应的联合策略，最后为整个系统找到最佳策略。构建策略树的方式有两种，一种是离线的方式构建策略树，在离线的情况下对整个状态进行搜寻和构建，这种方法的好处是不用考虑时间的消耗，可以用一天甚至一个月求解一个问题；劣势就是可能需要维护比较庞大的策略树，而且假如在构造过程中有没有考虑到的情况，则智能体在执行过程中将无法进行。也就是说离线的方式虽然能够完全考虑到所有的情况，以牺牲时间为代价寻求完整的策略，但是不会根据实际遇到的情况进行转变，同时规模庞大的策略树也容易让智能体在执行过程中消耗一定的决策时间。在线的求解方式，就是智能体在决策周期一边进行规划一边执行动作，其优势在于无需考虑所有可能的情况，只对当前遇到的情况进行求解和规划，因此就极大程度降低了策略空间。相比离线求解方式，在线的求解方式最大的劣势就是求解时间过短，尤其是在实时决策系统中，每个智能体可以用来决策的时间是非常短的，比如机器人足球 2D 仿真比赛中，每个决策周期只有 100 毫秒，要在 100 毫秒对进行精细的规划往往是不现实的。考虑到模型求解的时间等原因，本章将首先对几种离线规划的求解方法进行分析和介绍。

5.2.2 DEC-POMDP 离线规划的求解算法

蛮力法(Brute Force Search)[12,41]是构建策略树的最简单方法，其实也就是用完全枚举的方式将所有智能体可能的策略进行列举，分别为每个智能体维护相应的策略树，然后通过建模过程中设置好的回报函数为系统中每个智能体执行相应的动作进行衡量，找出能让整个系统获得最大回报的联合策略。蛮力法是最形象也最简单的求解方式，但是完全枚举的方式，将消耗大量的时间和空间，如果智能体数目较多或者决策时间较长，整个计算空间将迅速增加，经常导致普通计算机无法完成运算的可能。因此蛮力法只适合小规模的 DEC-POMDP 模型在有限阶段的决策规划。

MAA*[42]算法，也叫做多智能体的 A 星算法，是比蛮力法相对好的策略树枚举算法。MAA*算法从一个完整的联合策略开始，根据联合策略得到的启发值来不断地展开这个联合策略直到得到了最佳策略。与蛮力法不同的是，MAA*算法中策略树是从根节点展开生成的不完全策略树，也就是说展开生成的策略树是从根节点到第 *t-1* 步的节点，但是并不生成从 *t* 到第 *T* 步的节点。因此对这个策略树的评价值也分为两个部分：首先是利用贝尔曼式对根节点到 *t-1* 步的节点进行计算；而 *t* 到 *T* 步的节点由于并没有生成完全的策略树，所以采用一个启发值进行计算。这个启发值的计算一般是忽略 DEC-POMDP 模型中的观察值，将问题退化为 MMDP 后，计算得到的最优函数值，此时的值也被认为是实际值的一个上界值。MAA*算法通过将策略树进行不完全扩展的方式，一定程度上能够降低问题的规模，在应用上比蛮力法有更好的效果，但是依然是理论上能够求得最佳的联合策略。同样地，MAA*仍然只能应用于求解小规模的 DEC-POMDP 问题[12]。

上面两种离线求解方式都被称为精确离线求解方法，因为算法构造过程中都是依据于理想状态进行求解，不可避免地，这些算法都只能对小规模的 DEC-POMDP 模型求解，一旦问题规模逐渐增大，精确离线方式仍然不能对问题进行求解，因此近似离线的方法就应运而生。基于联合均衡的策略搜索算法，简称 JESP[41]，是第一个被提出的 DEC-POMDP 近似离线求解算法。

在多智能体系统决策中，每个智能体不仅要考虑系统状态，还需要对其他智能体可能采取的决策进行分析和利用，然后对自己的行为进行规划。当不知道其他智能体决策的情况下，直接进行推理是非常困难的。因此如果其他智能体可能采取的策略集已经确定，或者能够提前得到，智能体的决策就简单多了。比如在机器人足

球比赛中，不同的智能体担任着不同的职能，守门员的职能就是保证对方不进球，前锋的职能就是尽可能地创造机会进球，然后当前锋独自带球闯入禁区的时候，主要目标就是进球，此时对方守门员的主要任务就是防止自己进球，因此前锋在推测对方守门员策略时候，很明显需要知道对方守门员的目标是防止自己进球，而不可能会退回到自己球门后方或者左右边。这个例子就说明，在多智能体系统中，单个智能体决策必须同时考虑系统状态和其他智能体可能的决策，不知道其他智能体决策的时候，此智能体的决策就必须要考虑多种可能性，然而这些可能性或许并不是决策中需要考虑的，是可以被忽略的，如果一起纳入决策过程中，则会增加决策的难度和问题规模。

JESP 算法的思路就是：当单个智能体 i 决策的时候，给定其他智能体的策略集 Q_{-i}，那么智能体 i 的信念状态就可以定义为系统状态与其他智能体策略集的概率分布，即用$\Delta(S*Q_{-i})$表示智能体 i 的信念状态，在这个信念状态的前提下，为智能体 i 生成一系列的候选策略，然后在这些策略中选出对智能体 i 最有利的策略，随后继续这个过程，保证所有的智能体都按照同等的策略进行改进，直到所有智能体的策略无法再被改进为止[12,41]。算法 6 描述了 JESP 算法的步骤：

算法 6：JESP 算法
1. 产生随机联合策略； 2. 对任意智能体 i， 　为系统中除了 i 以外的所有智能体策略进行填充： 　　2.1 在 t=T 到 1 的时候， 　　计算所有的可能的信念状态B_i； 　　2.2 在 t=1 到 T 的时候， 　　对任意属于B_i的信念计算出其相应的值函数V_i； 　　2.3 对于所有可能产生的观察序列进行迭代： 　　在 t=T 到 1 的时候， 　　更新信念状态并根据值函数选择最佳动作。 3. 重复步骤 2 直到所有联合策略无法改进为止； 4. 返回当前最佳联合策略。

JESP 算法会按照等同概率对每个智能体 i 进行策略改进，直到所有智能体的联合策略无法被改进，即得到的值函数无法改进的时候就停止搜索并返回最佳策略。整个求解过程比较简单，由于已经从其他智能体可能的策略集中降低了问题规模，因此这种方法可以算近似求解中比较有利的方法，但是 JESP 最主要的问题就是容易陷入局部最优而无法对整个策略进行改进，同时求解过程中信念状态空间还是很大的，求解的时间复杂度依旧很高[12]。

DEC-POMDP 有很多离线求解算法，大部分都是在权衡时间和空间复杂度方面进行研究[12]，本节只是介绍几种最常见和经典的离线求解算法，这些算法最早被提出时候是解决 DEC-POMDP 问题最优秀的算法，后期随着 DEC-POMDP 常用模型的扩充，模型中状态空间、信念空间不断增大，其求解难度也得到了指数级增大，常见的离线规划已经开始不能满足需求，因此研究者就展开了降低大规模 DEC-POMDP 模型计算空间的研究。

5.3 分组有限空间离线规划算法

在上一节我们分析了几种 DEC-POMDP 离线规划求解的算法，离线规划的思想是通过对智能体策略空间进行搜索并寻得最佳联合策略，整个求解过程是离线状态下完成，严格意义上说是没有时间和空间限制。但是如果问题规模得到极大提高，而普通计算机的资源又有限的情况下，面对庞大的联合策略，普通计算机可能用长时间的运算也无法完成搜索和求解。因此如何利用有限的时间和空间资源对问题进行求解就成了 DEC-POMDP 离线规划求解中最主要的研究问题。

JESP[41]是最早被提出最有效的离线规划求解算法，其主要思想还是用枚举的方式对 DEC-POMDP 中联合策略进行搜索，不同之处是求解过程中以单个智能体作为研究对象，同时先获得了其他智能体可能执行的策略，因此在一定程度上能够降低联合策略空间和问题求解的规模，但是依然不能保证对大规模问题进行合理求解。完全枚举的方式是离线规划求解很好的理论思想，但是完全进行枚举不可避免出现了很多没用的策略，也就是说在成千上万条策略中，仅有几个策略是最佳的，如果要维护所有可能生成的策略并一一求解出最佳策略，消耗的时间和空间是非常大的。基于这个思想，我们提出一种叫做分组有限空间的离线规划求解算法，在 JESP 算法的基础上利用分组有限空间对所生成的策略进行维护，抛弃所产生的没用策略，以提高策略求解的效率。

分组有限空间离线规划求解算法是对 JESP 算法的扩充，在较短阶段的决策中，生成的策略树比较小，很容易得到多个智能体的策略树并找到最佳联合策略，如果决策周期比较长，将生成较大的策略树，在这组较大规模策略数种寻求最佳策略难免是费时的。事实上在每一步决策过程中，单个智能体生成的策略树深度与决策周期有关，因此每个决策周期，每个智能体的策略树都会在之前基础上得到一次扩展，

扩展的过程是对整个系统的状态和其他智能体的策略进行评估，分析自己应该做出的决策，在 JESP 算法中，这个步骤就是对信念状态的计算。事实上，策略树扩展后也会根据值函数进行运算，分别为每个新生成节点进行评估，并维护所有产生的新的子策略，然后再进入下一个过程，如此反复。在分组有限空间的离线规划算法中，我们将每次扩展生成的策略树分支数按照值函数进行分组，相应地对每个智能体在这个决策周期的策略树分支数按照值函数进行分组，然后对每一组中最优值函数的策略进行下一步规划，并生成新的子策略，最终再根据系统最初设置的系统回报函数进行评价，并得到系统的最佳策略。这种算法的优势是用分组有限空间去维护每个决策时刻需要生成的子策略，从而避免对所有策略都进行重复计算，在有限的空间中有效地降低求解时间。算法步骤如下描述：

算法 7：分组有限空间离线规划算法
1. 产生随机联合策略，并初始化分组数目 N 2. 对任意智能体 i， 为系统中除了 i 以外的所有智能体策略进行填充： 2.1 在 t=T 到 1 的时候， 计算所有的可能的信念状态B_i； 2.2 在 t=1 到 T 的时候， 对任意属于B_i的信念计算出其相应的值函数V_i； 将V_i随机分为 N 组，分别计算出 N 组中最优值函数。 2.3 对于所有可能产生的观察序列进行迭代： 在 t=T 到 1 的时候， 更新信念状态并根据值函数选择最佳动作。 3. 重复步骤 2 直到所有联合策略无法改进为止； 4. 返回当前最佳联合策略。

分组有限空间离线规划算法是基于 JESP 算法的简单重构，利用分组的方式对 DEC-POMDP 问题进行求解，目标是在有限空间中降低大规模 DEC-POMDP 问题求解时间。根据上面的算法描述可以清晰地发现分组的数目是最重要的影响因素，当分组数目为 1 的时候，算法基本上很难找到最佳策略，因为每一步规划都在找当前最优的策略，而整个问题的规划过程中并不是每一步都是最优的，问题求解是为了找到整体最优的规划；而当 N 的数目无限增大并趋于信念状态空间的时候，算法必然能够在不考虑时间和空间的情况下得到最佳策略，但是也忽略了我们研究的出发点：在有限空间的情况下有效地降低问题求解时间。另外在分组过程中用随机分组的方法而不用先排序再分组的方法是因为如果排序的过程已经将问题求解时间无限地增加。

5.4 实验及结论

本章简单介绍了 DEC-POMDP 在多智能体系统决策中的应用，并介绍和分析了几种的离线规划的求解方法。DEC-POMDP 问题求解目前仍是一个待解决的问题，因为求解过程比较复杂，而且对时间和空间的要求比较高。为了在有限的空间内降低求解 DEC-POMDP 求解时间问题，我们大胆地为策略树扩展部分提出了随机分组的方法，能够有效地降低问题规模，并提高求解速率。本节将通过一系列实验来验证分组有限空间离线规划算法在 DEC-POMDP 中的有效性。

5.4.1 标准测试问题集

本节将介绍 DEC-POMDP 模型中标准测试问题(benchmark problem)集，由于目前对 DEC-POMDP 的研究仍然处于理论分析阶段，问题求解的空间和时间是限制其应用于实际多智能体系统的主要原因，由于考虑多个智能体的决策，其建模的过程也非常复杂。用标准测试问题集作为检测 DEC-POMDP 算法是目前在多智能体系统决策研究中常用的方式。

常见的标准测试问题包括本章前面提到的多智能体老虎问题，这个问题是比较简单的 DEC-POMDP 标准测试问题[12]。另外还有广播信道问题(Broadcast Channel)[43]，火星漫游车问题(Mars Rover)[44]，格子相遇问题(Grid Meeting)[43]等，这些问题都是目前广泛应用于评价 DEC-POMDP 模型的求解算法。广播信道问题是一个简化的两个智能体的网络通讯问题。在任意一个决策周期中，每个智能体都要决定是否要在公共的信道中发送一条信息。如果两个智能体同时在一个周期向公共信道发送问题，则会发生冲突，则两条信息都无法发送。在这个问题中总共包含有 4 个状态，每个智能体包含有 2 个动作和 5 个观察值。格子相遇问题中则是简单描述智能机器人移动问题，两个机器人在一个没有障碍的格子世界中移动，其任务是在尽可能短的时间内相遇。实验一般采用的是 3*3 的格子，同时根据实际情况为每个机器人的移动等引入噪声，这个问题中机器人可以在 9 个格子中任意移动，因此系统状态共有 81 个状态，每个机器人就有 5 个动作和 7 个游泳的观察信息来识别周围的物体。火星漫游车(Mars Rover)问题要比前面几个问题都复杂，在这个问题中共有两个智能体，共 256 个状态，每个智能体有 5 个动作和 8 个有效的观察值。这个问题的求解难度

要远远高于前面两个问题。

5.4.2 MADP 工具箱

本实验对所有标准测试问题进行测试是采用 MADP[45]工具箱，MADP 工具箱是由 MIT 博后研究员 Frans A Olieho 和 DTU 研究员 Matthijs T.J. Spaan 于荷兰阿姆斯特丹大学攻读博士学位期间完成，目的是为智能体决策研究者提供一个开源和标准的平台进行马氏决策理论研究。

MADP 工具箱是目前公认多智能体规划决策研究很好的工具，主要针对 DEC-POMDP 模型求解。MADP[46]包含四个库，分别是底层库、解析库、支撑库和规划库。所有 DEC-POMDP 模型均用.dpomdp 文件格式描述，并通过解析库和规划库纳入工具箱进行规划和求解。一般研究者所作的算法研究都是在规划库中实现相关算法并运行得到相应的结果。MADP 中已经对所有 DEC-POMDP 的模型进行解释和建模，是一个有效简单地研究算法的工具。

5.4.3 多智能体系统规划实验

分组有限空间的离线规划求解方法是将 JESP 算法中计算所得的信念状态空间进行随机分组，然后分别通过值函数的计算来挑选每组的最优策略并进入下一阶段的规划过程，所分组的数目 N 的取值范围为1~$|B_i|$。本节是通过实验验证我们提出的分组有限空间的离线规划求解算法的有效性，因此设置一系列的 N 值来验证这种方法为 DEC-POMDP 模型求解的意义。由于每个实例里面得到的信念状态空间大小不一，因此实验中我们分别取$N \in \{1, \frac{1}{4}|B_i|, \frac{1}{2}|B_i|, \frac{3}{4}|B_i|\}$，并通过 MADP 工具箱分别为多智能体老虎问题、广播信道问题、火星漫游车问题和格子相遇问题进行求解，为做比较，同时在 MADP 工具箱中利用蛮力法和 JESP 算法求解如上问题，并统计每种情况下最优策略下的累计回报值和运行时间。为避免不同决策周期引起的累计回报值和运行时间的不一致，我们还需要设置不同决策周期下的求解最优策略得到的累积回报值和算法运行时间。根据实验测试我们发现蛮力法这种完全枚举的方式只能求解简单的问题，比如求解多智能体老虎问题，当决策周期为 3 的时候，蛮力法共用了 83.65 秒的时间进行求解；决策周期更长，则蛮力法求解过程中所需要的时间更长，甚至无法直接求解。为保证实验顺利进行，我们首先选定决策周期为 3

进行实验，并在表 5.1 中进行数据统计，运行时间超过 300 秒还未能求解的问题，则记为--。

表 5.1 H=3 时数据统计(时间单位为 s)

Table. 5.1 Statistic of solving time when H=3

		多智能体老虎		广播信道		火星漫游车		格子相遇	
		回报	时间	回报	时间	回报	时间	回报	时间
蛮力法		**5.191**	83.65	**2.99**	1.24	--	--	--	--
JESP		-19	0.49	2.8	0.34	--	--	--	--
分组有限空间离线规划	1	-65.175	0.41	1.2	0.38	--	--	--	--
	$\frac{1}{4}\|B_i\|$	-9.5	0.43	2.19	0.31	--	--	--	--
	$\frac{1}{2}\|B_i\|$	**5.19081**	**0.37**	**2.99**	**0.29**	--	--	--	--
	$\frac{3}{4}\|B_i\|$	-35.0625	0.45	2.8	0.31	--	--	--	--

表 5.2 H=5 时数据统计(时间单位为 s)

Table. 5.2 Statistic of Solving time when H=5

		多智能体老虎		广播信道	
		回报	时间	回报	时间
蛮力法		--	--	—	--
JESP		-24.8092	4.97	3.199	1.98
分组有限空间离线规划	1	-49	3.87	3.37	1.84
	$\frac{1}{4}\|B_i\|$	-36	3.23	4.6	1.93
	$\frac{1}{2}\|B_i\|$	**-11.4458**	**2.86**	**4.79**	**1.68**
	$\frac{3}{4}\|B_i\|$	-84.4902	3.56	3.289	1.91

通过表 5.1 可以发现目前几种离线规划求解方法均不能对规模较大的火星漫游车和格子相遇问题进行求解。而在多智能体老虎和广播信道问题中，明显发现分组有限空间离线规划方法的求解时间要比蛮力法和 JESP 算法的求解时间更短，同时当 $N=\frac{1}{2}|B_i|$ 的时候，对此两个问题所获得的回报值最大。这也说明我们提出的算法在一定程度能够降低问题求解的时间并取得相对较好的效果。为保证实验的完整性，我

们再次以决策周期为 5 进行上述实验，由于在决策周期为 3 的时候，火星漫游车和格子相遇问题均已经无法求解成功，我们在决策周期为 5 的时候只对多智能体老虎和广播信道进行实验，并统计出相应的数据，如表 5.2 所示。

在表 5.2 中可以发现当决策周期增加到 5 的时候，蛮力法已经无法对多智能体老虎和广播信道问题进行求解，而此时分组有限空间离线规划的方法仍然能够和 JESP 算法一样对这两个问题进行求解，而且当 $N=\frac{1}{2}|B_i|$ 具有比 JESP 算法更高的回报值以及最低的运行时间。这也说明我们提出的分组有限空间离线规划的方法在一定程度上能够有效地促进解决多智能体系统决策规划问题。

5.4.4 实验及分析

在上面进行的几组实验中，我们简单地将分组有限空间离线规划的方法应用于几种常规的 DEC-POMDP 模型求解中，通过对系统累计回报函数和运行时间的对比，我们发现分组有限空间离线规划的方法能够一定程度提高问题求解的速率，同时和 JESP 算法一样，能够解决蛮力法不能求解的问题。

由于在分组有限空间离线规划算法中我们只是简单地选择了几个分组的数目 N，两次实验均发现当 $N=\frac{1}{2}|B_i|$ 时候能够取得最好的效果，但是我们并没有全面对其他实验中 N 的数目进行分析，因此不能简单做出当 $N=\frac{1}{2}|B_i|$ 的时候，分组有限空间离线规划算法能够在 DEC-POMDP 模型求解中有最好的效果。但是从整体的效果看，这种方法确实能够有效地解决一些常见的问题，同时其算法运行时间也得到了一定的降低。另外在第一组实验中我们也发现，目前我们提出的这种离线的规划方法仍然不能对大规模的 DEC-POMDP 问题进行求解，只是在小规模问题中能够有着相对明显的优势。

5.5 小结

本章介绍了 DEC-POMDP 模型，这是目前在多智能体系统决策问题研究中常用的模型。DEC-POMDP 模型求解要比 MDP 和 POMDP 更难，因为规划过程中要考虑多个智能体之间的关系。

DEC-POMDP 求解一般有离线规划和在线规划两种：离线规划的方法对时间和

空间的要求比较高，而在线规划又不能保证有足够的时间进行规划。因此目前的研究主要是在有限空间降低离线规划的求解时间。本章提出分组有限空间离线规划求解的方法，以降低策略树规模的方式来降低求解时间。为验证本方法的意义，文中在 MADP 工具箱中利用几个常用的检测 DEC-POMDP 模型的标准测试问题(benchmark problem)对几种离线规划算法与分组有限空间离线规划算法进行对比实验，通过数据的整理和分析验证了本章提出的方法能够在有限空间降低 DEC-POMDP 模型的求解时间。

结论与展望

机器人足球是促进人工智能、控制决策理论和智能机器人领域发展的一项研究，本文的工作是以机器人足球 2D 仿真比赛中球员高层策略的研究开展。在机器人足球 2D 仿真中，将每个球员看做是一个智能体，整个系统就可以看做是典型的分布式多智能体系统。马氏决策理论是智能体决策研究较好的理论和平台，尤其对非确定环境下智能体决策提供了较好的模型和算法。本文首先介绍了几种马氏决策理论的基本模型：马尔科夫决策过程(MDP)，部分可观马尔科夫决策过程(POMDP)和分布式部分可观马尔科夫决策过程(DEC-POMDP)。基于 2 这几种基本决策模型，本文的工作总结如下：

1. 利用 MDP 模型为机器人足球 2D 仿真球队中持球球员进攻决策进行建模，并提出了值函数分解的迭代求解方法，通过实验验证了此方法的有效性。
2. 利用 POMDP 模型为机器人足球 2D 仿真球队中守门员决策进行建模，并提出了基于临界状态的求解方法，通过实验验证了经过 POMDP 模型建模的守门员在扑球成功率上要高于经过普通基于规则建模的守门员。
3. 对 DEC-POMDP 决策过程进行研究，提出了分组有限空间离线规划的方法，并在常用 DEC-POMDP 标准测试问题中验证这种方法能够有效降低问题求解的时间。

通过本文的研究我们发现马氏决策理论在智能体决策研究中有着重要的意义。本文的工作已经表明 MDP 和 POMDP 在机器人足球 2D 仿真球队中的应用有着较好的效果，同时文中也开展了一些 DEC-POMDP 模型相关的研究，由于 DEC-POMDP 模型求解时间较长，我们暂时未能将 DEC-POMDP 模型应用于机器人足球 2D 仿真球队中。未来希望能继续的工作有以下几点：

1. 将其他成熟的马氏决策理论模型应用于机器人足球 2D 仿真球队中，验证其相关的求解算法并进行改进。
2. 研究大规模 MDP 和 POMDP 问题的求解方法，提出可行有效的算法改进这两种模型在大规模非确定环境下决策的表现，为这些模型应用于更多的决策环境奠定基础。

3. 继续开展 DEC-POMDP 模型的研究，提出更好的算法降低求解时间，并根据实际应用改进模型，为本模型及相关模型应用于机器人足球 2D 仿真等实时性要求高的系统打下基础。

参考文献

[1] Shoham Y. Agent-oriented programming[J]. Artificial Intelligence, 1993, 60(1):51-92.

[2] Lane D M, Mcfadzean A G. Distributed problem solving and real-time mechanisms in robot architectures [J]. Engineering Applications of Artificial Intelligence Journal. 1994, 7(2):105-117.

[3] Franklin S, Graesser A. Is it an agent, or just a program? A taxonomy for autonomous agent[C]. Proceesing of the 3rd International Workshop on Agent Theories, Architectures, and Language. Budapest, Hungary. Springer-Verlag, 1996, 21-35.

[4] Wooldridge M, Fisher M. Agent-based software engineering[C]. In IEEE Proceedings on Software Engineering, 1997, 144(1):26--37.

[5] Stuart Russel, Peter Norving. Artificial Intelligence: A modern Approach(Second Edition)[M]. Pearson Education, 2003.

[6] Puterman M.L. Markov Decision Processes[M]. New York, John Wilkey and Sons, 1994.

[7] Eugene A. Geinberg, Adam Shwartz. Handbook of Markov Decision Processes(Methods and Applications)[M]. Kluwe, 2002.

[8] R. Bellman. A Markovian Decision Process[J]. Journal of Mathematics and Mechanics 6, 1957.

[9] Geffner, H. Modelling Intelligent Behaviour: The Markov Decision Process Approach[C]. In Proceedings of the 6th Ibero-American Conference on AI: Progress in Artificial Intelligence, 1998, 1484: 465-540.

[10]石柯. 基于马尔科夫决策过程理论的 Agent 决策问题研究[D]. 中国科学技术大学硕士学位论文. 2010 年.

[11]范长杰. 基于马尔科夫决策理论的规划问题的研究[D]. 中国科学技术大学博士学位论文.2008 年.

[12]吴锋. 基于决策理论的多智能体系统规划问题研究[D].中国科学技术大学博士学位论文.2011 年.

[13] S.P. Singh, T. Jaakkola. Model-free reinforcement learning for non-Markovian decision problems[C]. In Proceedings of the 11th International Conference on Machine Learning, Morgan Kaufmann, San Francisco, 1994:284-292.

[14] Craig Boutilier, Richard Dearden. Using Abstractions for Decision-theoretic Planning with Time Constraints[C]. In Proceeding of the 12th National Conference on Artificial Intelligence, Seattle , 1994:1016-1022,.

[15] Craig Boutilier. Exp Loiting Structure in Policy Construction[C]. In Proceeding of the 14th International Joint Conference on Artificial Intelligence. Montreal, CA ,1995:1104-1111.

[16] 张友红，谷文祥. 非确定性规划及带有时间和资源的规划的研究[J]. 计算机应用研究，2005 年第 3 期，37-42.

[17] Kitano, Asada. RoboCup: A Challenge Problem for AI[J]. AI Mahazine, 1997, 18:73-85.

[18]王健. 机器人导航 POMDP 算法研究[D]. 哈尔滨工程大学硕士学位论文. 2008 年.

[19] Mao Chen, Klaus Dorer. RoboCuo Soccer Server Manual 7.07[Z]. RoboCup Federation, 2003.

[20] Daniel W. An Introduction to Markov Processes[M]. New York , Springer Berlin Heidelberg, 2000.

[21] Howard R A. Dynamic Programming and Markov Processes[M]. Cambridge Mass , MIT press , 1960.

[22] Hansen, Zilberstein. LAO*: A heuristic search algorithms that finds solutions with loops[J]. Artificial Intelligence Magazine, 2001, 129:35-62.

[23]Barto A G, Bradtke S J. Learning to act using real-time dynamic programming[J]. Artificial Intelligence Magazine, 1995,72:81-138

[24]Barto A G. Labeled RTDP: Improving the Convergence of Real-Time Dynamic Programming[C]. In Proceeding of 13th International Conference on Automated Planning and Scheduling (ICAPS). Trento, Italy. AAAI Press. 2003, pp:12-21.

[25] 王芬. 一种实时 POMDP 求解算法及其应用[D]. 南开大学硕士学位论文.2007 年.

[26] HB Mcmahan. Bounded real-time dynamic programming: RTDP with monotone upper bounds and performance guarantees[C]. In Proceeding of 22nd International Conference on Machine Learning, Bonn, Germany, 2005, 569-576.

[27] Trey Smith. Focused real-time dynamic programming for MDPs: squeezing more out of a heuristic[C]. In Proceeding of AAAI'06 proceeding of the 21st national conference on Artificial Intelligence. 2006, 1227-1232.

[28] Leslie Pack Laelbling. Planning and acting in partially observable stochastic domains[J]. Artificial Intelligence magazine, 1998, 101:99-134.

[29] G.E.Monahan. A survey of partially observable Markov decision processes: theory, models, and algorithms[J]. Management Science, 1982, 28(1):1-16.

[30] K.J.Astrom. Optimal control of Markov decision processes with incomplete state estimation[J]. Journal of Math. Anal. Application,1977,10:174-205.

[31] Smallwood, Sondik. The Optimal Control of Partially Observable Markov Processes over a Finite Horizon[J]. Operations Research, 1973, 21(5):1071-1088.

[32] Cheng, H. Algorithms for Partially Observable Markov Decision Processes[D]. PhD Thesis of University of British Columbia, 1988.

[33] Zhang.N.L, Liu,W. Planning in stochastic domain: Problem Characteristic and Approximation[R].Technical Report of HKUST-CS96-31, Department of Computer Science, Hongkong University of Science and Technology,1996.

[34] Cassandra, A. , Littman, M.L. Incremental Pruning: A simple, Fast, Exact Method for Partially Observable Markov Decision Processes[C]. In Proceeding of the 13th Conference on Uncertainty in Artificial Intelligence, 1997, 54-61.

[35] Hansen, E. A. Solving POMDPs by Searching in Policy Space[C]. In Proceeding the 14th Conference on Uncertainty in Artificial Intelligence, 1998, 211-219.

[36] Braziunas, D. Stochastic local search for POMDP controolers[C]. In Proceeding the 19th National Conference on Artificial Intelligence(AAAI-04).2004, 690-696.

[37]Hidehisa Akiyam. HELISO2011 Team Description[R]. In Proceeding 2011 RoboCup Soccer Simulation 2D league, Turkey, 2011.

[38]Aijun Bai. WrightEagle 2D Soccer Simulation Team Description 2011[R]. In Proceeding 2011 RoboCup Soccer Simulation 2D league, Turkey,2011.

[39]王学宁. 策略梯度增强学习的理论、算法及应用研究[D].国防科学技术大学博士学位论文，2006 年.

[40] Craig Boutilier. Planning, learning and coordination in multiagent decision processes[C]. In Proceeding of the 6th Conference on Theoretical Aspects of Rationality and konwledage, 1996, 195-210.

[41] Ranjit Nair. Taming decentralized POMDPs: Towards efficient policy computation

for multiagent settings[C]. In Proceeding of the 18th International Joint Conference on Artificial Intelligence, 2003, 705-711.

[42]Daniel Szer. MAA*:A heuristic search algorithms for solving decentralized POMDPs[C]. In Proceeding of the 21st Conference on Uncertainty in Artificial Intelligence, 2005, 576-590.

[43] Deniel Szer. Bounded policy iteration for decentralized POMDPs[C]. In Proceeding of the 19th International Joint Conference on Artificial Intelligence, 2005, 1287-1292.

[44] Chistopher Amato. Incremental policy generation for finite-horizon DEC-POMDPs[C]. In Proceeding of the 19th International Conference on Automated Planning and Scheduling, 2009, 2-9.

[45] Matthijs T. J. Spaan. Approximate planning under uncertainty in partially observable environments[D]. Ph.D. Thesis, Universiteit van Amsterdam, 2006.

[46] Matthijs T. J. Spaan. The MultiAgent Decision Process toolbox: software for decision-theoretic planning in multiagent systems[Z]. 2009.

[47] Aijun Bai, Feng Wu. Online Planning for Large MDPs with MAXQ Decomposition[C]. In Proceeding of the 11th International Conference on Autonomous Agents and Multiagent Systems(AAMAS2012), Valencia, Spain, June 2012.

[48] Geoffrey Hinton. Unsupervised Learning and Map Formation: Foundations of Neural Computation [M], Cambridge Matt, MIT Press, 1999.

攻读学位期间发表的论文

一．攻读学位期间发表的(含已录用)学术论文情况：

1. Chen Wei, Guo Jing. Hybrid Q-learning Algorithm about Cooperation in MAS. 21st Chinese Control and Decision Conference (CCDC 2009), Guilin, China. EI: 2009- 4712479751.

2. Guo Jing, Chen Wei. The Application of MD-MIML Frame Based on MIML. 3rd International Conference on Computer and Automation Engineering (ICCAE 2011), Oral presentation, Chongqing, China.

二．与学位论文内容相关的其他成果：

1. 2010年RoboCup 中国公开赛机器人足球2D仿真组 全国三等奖；

2. 2011年RoboCup 中国公开赛机器人足球2D仿真组 全国一等奖；

3. 2012年入围RoboCup机器人足球2D仿真世界杯决赛。

致谢

硕士研究生的3年生活即将结束，每每回首这忙碌又充实的3年，万千感慨，无以言表。在此，谨以最简单的语言表达我最真实的感想。

首先要感谢我的导师陈玮教授，在过去的几年中，陈老师一直非常关心我的生活学习和研究，无论是做人还是做事，陈老师都时时刻刻为我做榜样，随时提醒着我要谦虚谨慎。在我最消沉的时候，陈老师会放开自己的事情开导和关心我；在我获得荣誉的时候，陈老师会在庆祝之余告诫我继续努力。最难以忘怀的是陈老师这几年一直以来都支持我的学习和研究，给我提供了最好的条件外出比赛学习和交流。在此，无以言表，谨以短短百字表达我最真挚的感激。

再次要深深感谢徐维超教授，在徐教授来到广东工业大学一年多的时间内，以其行动给我们诠释了什么叫学习和研究，并热心地与我们交流论文写作方法和研究的经验与心得。尤其在本文完成后，徐教授亲自阅读本文，提出许多宝贵的意见并亲自为本文修改中英文摘要，在此深深感谢徐维超教授。

其次要深深感谢实验室全体，包括已经毕业在国内外发展的与尚未毕业继续努力的每一位实验室成员，谢谢你们一直的支持和鼓励，激发着我一直努力。

另外，感谢我的父母姐姐和姐夫，谢谢你们一直对我学习的支持，让我没有后顾之忧地奋斗在学业上。

最后，深深感谢Dollfie女士，谢谢你这几年对我的理解和关怀，在我不能陪在你身边的时候，充分地体谅和支持我。