浙江工业大学硕士学位论文

基于 Markov 决策理论的足球机器人协同机制研究

作者姓名：贾玉博

指导教师：杨马英 教授

浙江工业大学信息工程学院

2013 年 03 月

Dissertation Submitted to Zhejiang University of Technology
for the Degree of Master

Research on Soccer Robots Cooperation Mechanism Based on Markov Decision Theory

Candidate: Jia Yubo

Advisor: Prof. Yang Maying

College of Information Engineering
Zhejiang University of Technology
March 2013

浙江工业大学

学位论文原创性声明

本人郑重声明：所提交的学位论文是本人在导师的指导下，独立进行研究工作所取得的研究成果。除文中已经加以标注引用的内容外，本论文不包含其他个人或集体已经发表或撰写过的研究成果，也不含为获得浙江工业大学或其它教育机构的学位证书而使用过的材料。对本文的研究作出重要贡献的个人和集体，均已在文中以明确方式标明。本人承担本声明的法律责任。

作者签名：贾玉博　　日期：2013年05月25日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权浙江工业大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本学位论文属于

1、保密□，在_____年解密后适用本授权书。

2、不保密☑。

（请在以上相应方框内打“√”）

作者签名：贾玉博　　日期：2013年05月25日

导师签名：[illegible]　　日期：2013年　5月　28日

基于 Markov 决策理论的足球机器人协同机制研究

摘　要

多智能体系统的协调和协作机制，是目前人工智能研究的重点领域之一。多智能体系统的广泛应用决定了研究其协调协作机制有很大的现实意义。

本文以机器人足球比赛为背景，研究基于 Markov 决策过程（MDP）理论的多智能体协调和协作机制，完成的主要研究成果如下：

首先在一类通信条件良好的集中式控制方式下，基于任务层次分解的决策框架，结合博弈论的有关概念和方法，提出了一种基于效用函数预测的在线策略规划算法。在 FIRA 2D 仿真组比赛平台上的实验结果表明，该算法能够进行合理的行为选择，实现良好的团队合作效果。

其次，针对一类感知和通信受限的分布式大规模决策问题，应用基于 MAXQ 值函数分解的任务层次分解方法，提出了一种在线策略求解算法，MAXQ-RTP 算法。该算法设计了一种充分利用问题域受限的感知和通信资源的多智能体决策系统框架，基于与或图表示可行策略，在线实时地求解当前状态下的最优策略，可用于解决连续状态空间和动作空间的决策规划问题。

论文的主体实验工作在分布式控制的 RoboCup 2D 仿真组比赛平台上进行。通过对智能体有限感知和通信、决策系统规模等特点的分析，采用基于 MAXQ 值函数分解的 MDP 任务层次分解方法对球员智能体的决策问题进行建模，通过 MAXQ-RTP 算法在线求解智能体的最优策略。实验结果表明，该方法有较高的计算效率，通过协同决策使球队取胜的效果良好。

由于使用 MDP 模型进行球员智能体建模中对队友和对手策略的简化处理，上述 MAXQ-RTP 算法有可能丢失一些最优对策。下一步的研究工作主要是将模型扩展到对策论框架中，结合队友和对手的可能策略，求解最优对策。

关键词：多智能体系统，机器人足球，马尔可夫决策过程，MAXQ 值函数分解，在线规划

Research on Soccer Robots Cooperation Mechanism Based on Markov Decision Theory

ABSTRACT

The coordination and cooperation mechanism of multi-agent systems (MAS) is one of the most popular research areas of AI, which is of much practical significance due to the widely usage of MAS.

In this thesis, the coordination and cooperation mechanism of agents is studied based on Markov decision process (MDP). The main results of this work are as follows:

First, we proposed an action selection algorithm for well-communicated central control situations, which is based on MDP hierarchical decomposition method, by combining the concepts and methods of game theory and predicting the reword value of player's strategies. The algorithm was tested in the central controlled FIRA 2D robot soccer platform, which gave rather good performance in the games.

Next, for the problems of continuous state space and large scales of distributed controlled MAS with bounded perception and communication, a real-time policy planning algorithm called MAXQ-RTP was proposed, which incorporates the MDP task decomposition method based on the MAXQ value function hierarchical decomposition. The algorithm adopts a framework of fully using the perception and communication information, employing AND-OR graph to describe the possible policies of each agent, and performing real time optimal planning.

The main experimental work was performed on the distributed controlled RoboCup 2D platform. We used MAXQ hierarchical decomposition to model the soccer agents, applied the MAXQ-RTP algorithm to plan the optimal strategy in real time. Experimental results showed good performance of the MAXQ-RTP algorithm.

Due to the simplification of other soccer agents' reactions in the MDP model, some good strategies may be lost. In the following research we will extend the MAXQ-RTP algorithm in the game theory framework, taking other agents' reactions into consideration to make better decisions.

Key Words: Multi-agent System, Robot Soccer, Markov Decision Process, MAXQ value function decomposition, Online Planning

目　录

第1章 绪 论

多智能体系统（Multi-agent Systems，MASs）是目前人工智能领域一个相对较新的研究方向。多智能体系统旨在研究由多个智能主体相互协作，共同解决现实世界中的大型复杂问题。这些问题一般具有极高的复杂度，在通过提高单个智能体的能力不足以解决问题，或者花费成本很高的情况下，用相对简单的多个智能体协同工作，共同解决问题。多智能体系统目前已经是人工智能领域扩大解决实际问题能力的主要研究热点。其发展前景已在众多的应用领域得到了验证，如多机器人系统、交通系统、计算机网络、电子商务、分布式控制、协同制造等。

多智能体系统研究的内容很多，但根据智能体本身的特点，最重要的内容仍然是智能体的决策问题，因为决策是实现协调合作特性的基础。这也是本文研究的主要内容。本章作为介绍性的部分，将从人工智能开始，到多智能体系统，再到机器人足球比赛；从多智能体协作的研究，到目前比较有代表性的决策方法的研究，由广入微，由浅入深地介绍该领域当前的主要研究工作，并分析本文研究的方法的现状。

1.1 多智能体系统简介

多智能体系统是分布式人工智能（Distributed Artificial Intelligence，DAI）的重要分支。本节首先介绍人工智能与 MAS 的发展，然后对比传统人工智能的特点，介绍 MAS 的主要优点。

1.1.1 人工智能与多智能体

人工智能一词，最初是在 1956 年的 Dartmouth 学会上被提出来的，它是计算机科学的一个分支。人工智能最早正式的定义是：能制造智能机器，特别是智能计算程序的科学和技术，类似于计算机理解人类智能，但是人工智能使用的方法并不要求和生物上的方法一样[1]。此后，人工智能获得了巨大的发展，也引起了不同学科和不同领域的越来越多的研究人员的重视，成为一个涉及知识广泛的交叉学科。主要研究通常需要人类才能完成的复杂工作。

人工智能旨在构建实用的并能够自行解决现实问题的系统，通常还要求这个系统具有

学习的能力。近二三十年人工智能领域提出的新概念—智能体（Agent）。一般认为智能体是一个能够通过传感器感知所处的环境，并能通过执行器作用于外部环境的主体。但是，学术界对智能体的定义一直以来并不统一，所以一般更倾向于用 Agent 具有的特点来描述智能体。[2, 3]认为一个智能体应具有自主性、反应性、社会性和预动能力等基本特点，根据情况还可以具有移动性、自适应性、理性等特性。Rusell 等认为 Agent 要能够通过感知了解环境，并做出动作改变环境。Jennings 对智能体的描述，认为 Agent 应具有情景性、自制性和适应性[4]。不管是采用何种定义，都可以看到Agent应该具有如下一些基本能力：认知能力、决策能力、行动能力。由这些特点可以将智能体认为是集中了人工智能的一个主体。

1997年，IBM的“深蓝”计算机击败了人类国际象棋高手Kasparov，这是人工智能发展史上的一个里程碑。它是单个智能体在静态不可预测环境下问题求解的巨大成功。这类问题也是传统人工智能的重要假设，即静态的、环境上下文（context）已知，并且主体有足够的时间进行思考决策。

现代人工智能所研究的对象一般更加复杂。动态环境、不确定性、决策时间限制、大规模状态空间、多主体等是这类问题主要的特点。例如交通管理系统、商业决策系统、多机器人作业等应用问题。在这些领域中，传统的人工智能方法在系统构造、求解上都遇到了极大的困难，甚至很多情况下不可能实现[5]。

随着互联网技术、并行计算技术的进步，分布式人工智能应运而生，并且在过去的二三十年中获得了快速的发展。分布式人工智能的目标主要是将大规模复杂系统拆分成相对较小的、相互之间可以协调的、易于管理的系统，并行地解决问题[6]。多智能体系统（MAS）作为分布式人工智能研究的一个重要分支，主要研究在动态、实时、不确定的环境下，一组具有自主能力的智能体，如何相互协调地与环境进行交互，高效地完成一定的任务。

在多智能体系统中，多个智能体拥有共同的目标。每个智能体都尽可能的推动总体目标的实现。任何智能体的行动都会导致环境的变化，同时每个智能体也都要努力地适应环境，以达到主体与客体的交互[7]。所谓的适应环境就是说智能体要理解环境的状态，明确自己采取的行动对环境的影响，以选择有利于总体目标的动作。一般的智能体设计方法都会根据领域知识，明确智能体行动的效用，根据智能体行动可能的收益来决定是否执行动作。采用这种机制设计智能体，也更加突出了智能体的理性、社会性和反应性。

1.1.2 多智能体系统特点

多智能体系统与单智能体相比，有如下特点：每个 Agent 拥有不完全的感知和问题求解能力，由此，智能体之间必须通过相互合作，协同作业，才能有效地完成总体任务。很多情况下可能不存在全局控制，这就意味着没有一个统一的协调者来进行任务调度和行为协调，为智能体在独立的决策和动作时，实现事实上的相互协作提出了很高的要求。数据是分散的或者分布式的，智能体必须能够利用局部的，很可能是不完整的数据进行计算。计算是异步、并发或并行的。这不仅仅是指计算的并行，还包含感知、动作的并行性，比起只是并行的计算更难于解决。不同智能体可能是异构的，这就要求对于不同结构的智能体分别进行设计，必须有效处理异构的信息和行为。

多智能体系统具有自主性、分布性、协调性，并具有自组织能力、学习能力和推理能力。采用多智能体解决实际问题，具有较强的鲁棒性和可靠性，并具有较高的问题求解效率。多智能体系统用于解决实际问题时有很多的优势特点。基于这些特点，多智能体系统对于复杂系统具有更强的表示能力，可以为很多实际复杂系统提供良好的模型框架。

1.2 机器人足球比赛简介

1.2.1 概述

机器人足球比赛的想法最初是在 1992 年由加拿大学者 Alan Mackworth 教授提出来的。随后在 1993 年，由日本学者 Minoru Asada、Hiroaki Kitano 和 Yasuo Kuniyoshi 创办了机器人足球世界杯（Robot World Cup Soccer Games，RoboCup）[8]。并于 1996 年在日本举行了第一届 RoboCup 比赛。随后，很多研究人员开始将机器人足球作为研究课题，并在 1997 年第 15 届国际人工智能大会上，正式将机器人足球比赛列为人工智能的一项挑战。这项挑战的最终目标是到 2050 年，全部由机器人组成的足球队伍，在人类世界杯足球比赛的规则下，击败当时人类世界杯冠军球队。至此，机器人足球比赛成为多智能体系统的一个标准问题。

机器人足球比赛作为一个多学科研究和测试的标准平台，涉及自动控制、机器人学、人工智能、传感与感知融合、通信、机械设计等众多学科。特别是多智能体系统的研究，包括动态不确定环境下的智能体协作、实时决策、多主体机器学习等当前人工智能研究的热点问题。本文主要以机器人足球比赛为背景，研究以实现多智能体协作为目的的智能体决策规划算法，因此，在下一小节将介绍机器人足球比赛的一些主要的特点。

1.2.2 机器人足球比赛特点

“深蓝”计算机系统是传统单智能体人工智能问题的一个典型代表。它是在一个静态的不确定性环境下，单个智能体通过知识推理来解决一个较大规模的对策问题。

机器人足球比赛是现代多智能体系统的一个典型研究平台。相比“深蓝”对象，主要有如下特点：

- 问题规模相当大

场上 22 名球员和一个足球都有连续的位置、速度和方向变量，球员的动作同样也是连续的变量。这构成了连续的状态空间和动作空间。从状态空间到动作空间的策略表示问题，是决策的一个重要的难点。

- 不确定性

球员观察到的信息是不确定的，带有一定的误差。动作的效果也有较大不确定性。加上对手策略的无法预测，使得决策问题具有很大的不确定性，解决决策规划问题的难度极大。

- 并行性

每个球员有独立并行的决策和动作。一般没有统一的智能体进行任务的协调分配，决策系统必须有良好的任务分配和冲突消解机制。

- 实时系统

一般情况下都有一个相对较短的决策周期，这对于决策系统算法的实时性要求非常高。再加上连续的状态空间，使得完全的在线规划很难在这有限的时间内找到最优策略。

- 合作与对抗

机器人足球比赛中，同队的球员之间是完全的合作关系，拥有共同的利益。对手间是完全的对抗关系，实际上一队获得多少收益，对方就损失多少。所以，如何使球员之间进行协商、规划以及在对抗中使己方获得最大利益，是足球机器人决策系统研究的主要问题。

- 通信受限

一般的比赛项目允许球员之间进行通信，但是对于通信的频率和内容量都有较严格的限制。而且可靠性没有保证，通信的内容不一定能够完全送达。这对设计者如何有效的利用有限的通信来提高决策的效率是个不小的挑战。

由于这些特点，使得机器人足球比赛成为一个比较标准的多智能体系统的研究和理论验证平台。其大部分特性与现实世界的多智能体系统相似甚至更复杂，在该平台得到的很多研究成果稍加改造即可用于现实系统中去。

1.2.3 足球机器人决策

机器人足球比赛举办以来，到现在已经有十几年的历史。在各个方面，都已经取得了很大的进步。本文主要研究的是多智能体的协同机制，这主要体现在足球机器人的决策系统中。

决策是指智能体在得到环境的状态后，经过一定的计算和权衡，决策出可以有效的利用多个球员智能体的行为能力，相互协作地完成任务的行动选择过程。决策系统是足球机器人众多研究领域中关于人工智能的方向。研究方法可以主要地分为决策协作框架的研究，基于决策理论的研究和基于机器学习方法的研究等。各方法的研究现状将在下节介绍。

1.3 研究现状

本节从回顾多智能体系统的主要研究内容出发，接着介绍足球机器人智能体决策方法的研究现状。通过各种方法的比较，引出本文的研究方法。

1.3.1 多智能体系统研究内容

对多智能体系统的研究，主要有智能体认知模型、体系结构、智能体协调协作机制、分布式控制和通信机制等方面。

- 智能体的认知模型

智能体的认知模型主要研究怎样形式化地描述智能体的特性，以及智能体如何根据得到的环境信息进行推理和决策的过程。认知模型认为智能体是一个有意识的系统。Bratman 最早提出用 BDI（Belief-Desire-Intention）模型来表示智能体。认为信念是智能体对当前世界状态的认识以及为达到某种效果可能采取的行为的估计；愿望是智能体对未来世界状态以及可能采取的行动的偏好；意图是得到了行动承诺的那部分目标。Rao 和 Georgeff 在 Bratman 工作的基础上给出了 BDI 模型的形式化描述，并引入了决策论的思想。认知模型更加侧重于对智能体主观状态的刻画[9]。

- MAS 的体系结构

单个智能体的体系结构主要研究如何将感知、决策、规划、动作等模块有机的结合在一起，形成具有特定功能的智能系统[10]。智能体的体系结构主要分为慎思型、反应型和混合型[11]。

而多智能体系统的体系结构主要研究如何将多个相互独立的甚至是异构的智能体组织在一起，使他们相互之间进行有效的协同合作，来解决复杂的现实问题。按照不同的组

织方式，可以分为集中式、完全分布式和混合式；从构成 MAS 的单个 Agent 出发，可以分为同构、异构和同异构混合；从构成系统的 Agent 之间的关系划分，又可以分为完全型网络结构、层次型网络结构和联盟型网络结构等[12] 。

- 通信机制

多智能体系统中的通信模式的划分有多种方式，较为常用的区分方法将其分为五种：无通信、方案传递、消息/对话模式、消息传送方式和黑板系统。篇幅限制，此处不多做介绍，可参考[13]。

- 多智能体协同机制

MAS 在复杂动态的环境下，如何有效地利用有限的资源，在有限的时间内解决资源分配、任务调度、行为协调、冲突消解以及协同合作问题，是 MAS 协调协作机制要完成的工作。而协调协作机制也是多智能体系统需要解决的最重要的问题之一。纵观近年来的一些研究成果，主要有以下几个方面。

对多智能体协作模型的研究，提出了很多智能体的模型。如，基于理性智能体的 BDI 模型，协商模型，协作规划模型，自协调模型等[14]。

较早时候对于多智能体系统协作机制的研究还有一方面是合作协议的设计。这些协议一般基于 BDI 模型，用信念、愿望和意图来表示和推理智能体的行为。智能体之间可以通过协商模式进行合作。也有人提出联合意图的概念，表示智能体之间为实现共同的目标而缔结的相互之间的承诺。这些合作机制一般基于逻辑协议，它们最大的劣势在于难于对智能体的行动定量的进行评估。特别是在环境存在不确定性时，对合作协议的评估需要用到很多不同的规则。另外，针对不同的问题域，合作协议一般需要具体设计。这样“自上而下”的方法缺乏通用性，而且有时对于本质上相同的问题域也可能需要不同的设计。也就很难对整个多智能体系统地协作问题进行分门别类的研究。

基于决策规划理论的研究。这方面最显著的成果莫过于基于Markov决策理论和博弈论思想的智能体决策规划研究。这在下一小节将有较详细的介绍。

基于机器学习的方法也是近年来研究最热门的方法之一。普遍认为智能体应当具有学习能力，能根据不断变化的环境，提高自己的知识和能力。通过加强学习方法求解智能体决策问题也是智能体决策理论的一大研究方向。

本文研究的目的是设计有效的多智能体协调协作机制。这将通过智能体的决策来实现。下面介绍一下智能体决策方面的主要研究现状。

1.3.2 足球智能体决策研究现状

决策系统是足球机器人智能体的大脑，控制着智能体的行动。这一节介绍机器人足球比赛发展的十多年里，决策系统主要的几个研究重点。其中较详细地介绍基于决策理论的策略规划方法的研究现状。

- 领域知识和技术

机器人足球比赛从举办至今，尤其是举办的前几年，很多研究者根据足球比赛的领域知识，将其应用于足球机器人智能体的设计中。例如，基于形势的决策站位 BPSP，基于球员角色的策略系统、以及固定战术 Setplay 等概念和方法[15]。

- 协调机制和协议

借鉴多智能体协调协作机制的很多成熟方法，成功地应用于机器人足球比赛中。例如，市场机制法、黑板模型法，基于 Petri 网的足球智能体决策系统设计等[16]。

- 基于决策理论的规划方法

近年来，有很多研究将决策论和对策论的方法引入足球机器人决策系统中。尤其是马尔可夫决策过程（Markov Decision Process，MDP）和博弈论的研究和应用。传统决策规划理论一般不考虑状态转移的不确定性[17]，例如，象棋比赛中，下子的结果和效果是确定的。而机器人足球比赛中，一个动作，比如踢球，球运行的方向和距离都有可能产生偏差，是不确定的。解决这个不确定性问题的方法一般就用马尔可夫决策理论。

Markov 决策理论是上世纪五六十年代开始研究的理论，有一套严格的数学模型，可以得到决策问题的最优策略。一般 MDP 问题的求解方法可以分为离线算法和在线算法。离线算法基本的有后向迭代算法和前向搜索算法。这些算法的基本原理是通过遍历问题的状态空间，搜索出累积回报最高的策略来执行。对于状态空间很大的问题，很难做到，一般都会加入各种启发式搜索方法来加速求解，如搜索算法 A*等。规模较大的问题，有基于层次分解的求解算法。在线算法在状态空间较大的情况下更难实现。

基本的 MDP 认为环境状态是完全可以得到的，而在环境只能部分观察到的情况下，对 MDP 进行扩展，得到部分可观察 MDP（POMDP），通过搜索信念空间，得到最优策略的概率分布。而这两个模型都假设决策者只有一个，其他所有的因素都归于环境。在有多个决策者存在的条件下，如果其他决策者的策略已知或者可以精确得到时，仍可以用 MDP 或 POMDP 模型进行策略求解。但是当其他决策者的策略不可知时，需要使用分布式 Markov 决策过程模型 Dec-MDP 和 Dec-POMDP 来处理它们的决策问题，使智能体系统实现协调合作。在这些模型中，智能体都有相同的收益函数，共同合作来实现集体目标。

而当系统中既有合作者又有对抗者的情况下，可以用 Markov 博弈和部分可观察 Markov 博弈来建模，也有学者称其为部分可观察随机博弈，即 POSG。

但是随着问题的不确定性以及状态空间的增加，策略求解将变得非常困难。在很多实际问题中，以上模型很难表示和求解。对这类大规模问题，目前应用最多的是基于任务层次分解方法（Hierarchical Approaches）的策略规划算法，来减少问题的复杂度[18-20]。其方法的核心是状态抽象（States Abstraction）的应用，可以有效地降低状态空间的大小，简化策略求解。

- 强化学习

强化学习（Reinforcement Learning）是目前人工智能领域研究最热的方法之一，同时也在很多领域得到成功应用。强化学习不同于其他学习方法的核心特点是智能体在没有例子参考的情况下，通过与环境的交互，得到智能体动作对环境的影响，取得一个动作评价的回报，然后逐步调整行动的选择，最终学到最优的策略。强化学习以 MDP 作为模型基础，其学习的结果得到的策略也是 MDP 的解，从这个意义上讲，也可以认为强化学习是 MDP 问题的求解方法[21]。目前最常用的强化学习方法是 Q-学习和 SARSA 学习方法[7]。如，德国的 BrainStorm2D 仿真球队十年来一直将强化学习作为研究方向，构建足球智能体，其队伍在近几年的 RoboCup 2D 比赛中一直名列前茅。

本文的主要工作是基于 MDP 理论展开的，通过一定的简化将足球机器人智能体进行 MDP 建模。由于问题的大规模特点采用基于 MAXQ 值函数分解的任务层次分解方法[19]，在各任务层通过状态抽象降低决策过程的复杂度。通过设计的实时策略规划算法，计算最优分层策略。

1.4 文章主要内容和安排

在机器人足球比赛环境下，观察信息的不确定性以及局部性，动作和状态空间是连续的，动作后的状态转移是不确定的，再加上有限的决策时间限制，求解最优策略难度极大。

本文对基于 MAXQ 值函数分解的 MDP 决策规划算法进行研究。以该方法为基础，设计了一种有效利用受限的感知能力和通信资源的多智能体决策框架结构。针对机器人足球比赛连续的状态空间问题，提出了一种基于与或图表示策略路径的在线策略规划算法。并从理论和实验两方面讨论了算法的效果。

本文的内容安排如下：

第一章从人工智能到机器人足球比赛，从多智能体到足球机器人智能体决策，介绍了

本文的研究背景和目前的研究情况。

第二章主要是基本背景知识的介绍，包括 MDP 概念知识的介绍，经典求解方法，及本文所研究的面对大规模问题的挑战和常用方法。

第三章是在智能体集中式控制方式下的分层决策规划方法的一次尝试。使用基于 MDP 任务层次分解的决策框架，结合博弈论的相关概念，提出了一种基于效用预测的球员智能体动作选择方法。在 FIRA 2D 标准仿真比赛平台上的实验验证了算法的有效性。

第四章介绍在大规模 MDP 问题中，基于 MAXQ 值函数分解设计智能体的决策模型的方法。在此基础上，设计了一种有效利用受限的感知和通信资源的多智能体决策框架。并提出了一种基于与或图表示策略路径，利用状态可达性条件，采用前向搜索方法的实时策略规划算法，称为 MAXQ-RTP。

第五章通过对 RoboCup 2D 仿真比赛平台问题域的分析，将基于 MAXQ 值函数分解的方法应用于球员智能体决策系统中。根据 MAXQ-RTP 决策规划算法，设计球员智能体的决策系统。通过实验验证了方法的有效性，也讨论了算法的不足之处。

第六章对文章工作进行总结，并展望下一步的研究方向。

第 2 章　背景知识

MDP 是对具有马尔可夫性的问题决策的基本模型，具有广泛的应用，也是本文的理论基础。本章介绍 MDP 的基本理论。首先介绍 MDP 的基本概念和模型，然后介绍几种经典的 MDP 问题求解方法，最后指出在大规模决策问题中，经典 MDP 可能面对的挑战，以及较为研究者接受的解决方法。

2.1　Markov 决策过程

人工智能领域最经典的行动规划模型是确定性模型[22, 23]。确定性模型包含有限状态集合 S，有限动作集合 A 和状态转移函数 f，表示一个状态怎样在一个动作的作用下转移到另一个状态。经典规划问题可以理解为具有以下特点的确定性状态模型：

S1. 离散的、有限的状态空间 S；

S2. 具有初始状态 $s_0 \in S$；

S3. 非空的目标状态 $G \subseteq S$；

S4. 每个状态 $s \in S$ 下可执行的动作 $A(s) \subseteq A$；

S5. 对于状态 $s \in S$ 和动作 $a \in A(s)$ 的确定性的状态转移函数 $f(s,a)$；

S6. 正的行动花费 $c(a,s) > 0$。

这个模型的解是一个动作序列 $a_0, a_1, ..., a_n$，它将产生状态的一个轨迹 $s_0, s_1 = f(s_0, a_0), ..., s_{n+1} = f(s_i, a_i)$。其中 a_i 是在状态 s_i 时执行的动作，s_{n+1} 是一个目标状态。当总的花费 $\sum_{i=0}^{n} c(s_i, a_i)$ 最小时，解就是最优的（Optimal）。

而具有不确定性的规划问题可以将上面确定性模型的确定性转移函数 $f(s,a)$ 改为状态转移的概率分布 $P_a(s' \mid s)$。不确定性模型具有如下特性：

M1. 离散的、有限的状态空间 S；

M2. 具有初始状态 $s_0 \in S$；

M3. 非空的目标状态 $G \subseteq S$；

M4. 每个状态 $s \in S$ 下可执行的动作 $A(s) \subseteq A$；

M5. 状态转移概率 $P_a(s'|s)$，表示在状态 s 执行动作 a，状态转移到 s' 的概率；

M6. 正的行动花费 $c(a,s)>0$；

在这个模型中，状态 s_t 下执行动作 a_t，转移到的状态 s_{t+1} 是不可预测的。实际上，模型要求状态是完全可观察的，观察到的状态 s_{t+1} 就相当于为选择动作 a_{t+1} 的反馈信息：

M7. 状态是完全可观察的[24]。

具有 M1-M7 特性的模型就是我们熟知的马尔可夫决策过程（MDP），或者更具体地说，是随机最短路径问题[25]。下面，我们给出 MDP 的具体定义。

2.1.1 马尔科夫决策过程模型介绍

马尔可夫过程(Markov Process)是俄罗斯数学家 Markov 在 1907 年提出的,它是一类具有普遍共性的过程：某一阶段的状态一旦确定，则此后过程的演变（状态转移）就不再受此前各状态的影响。也就是说，当前的状态是此前历史状态的一个完整的总结，此前的历史状态通过产生当前的状态去影响过程的未来状态。在现实世界中，有很多这类过程，如液体中微粒的布朗运动、青蛙在荷叶上跳跃的轨迹、蜂蜜采花的路径等[26]。

将决策者加入具有 Markov 性的系统中，通过决策者的动作和当前系统的状态就可以完全地确定系统未来的状态，而与系统此前的状态无关，这就是 Markov 决策过程（MDP）。经典的规划方法一般是基于确定性的环境进行的，如启发式搜索方法等。这些方法在现实应用中有很大的局限性。实际中的决策问题，智能体在某状态下执行动作的结果往往具有不确定性，正如上面 M1-M7 描述的模型。这类问题的决策规划一般用 MDP 模型来处理。

上世纪 50 年代，Shapley 研究随机对策和 Bellman 研究动态规划时，马尔可夫决策过程的基本思想就已经出现了。Blackwell 和 Howard 等人的研究工作奠定了 MDP 的理论基础[27]。

MDP 用来描述智能体与环境之间的相互作用。智能体通过观察环境状态作为输入，决策出动作作为输出，通过动作的执行来影响环境状态的改变。基本模型如图 2-1 所示。

正如不确定模型特性的 M7 所要求的，因为智能体动作对环境状态可能产生不确定的影响，要求智能体拥有完全的环境感知能力[28]。

Markov 决策过程可以定义为一个四元组 $\langle S, A, P, R\rangle$，其中：

- 状态集合 S：是有限的状态集合；
- 动作（行动）集合 A：是有限的动作集合；
- 状态转移函数 P：是状态转移函数，$P(s'|s,a)$ 表示在状态 s 下执行动作 a，环境状态

转移到s'的概率，即$P(s'|s,a)=\Pr\{s_{t+1}=s'|s_t=s,a_t=a\}$；

- 报酬函数R：是立即收益函数，$R(s,a)$表示智能体在状态s执行动作a可以获得的立即收益。实际上它即是状态s'的报酬函数，一般表示成与当前状态和当前状态下采取的动作相关[24]。

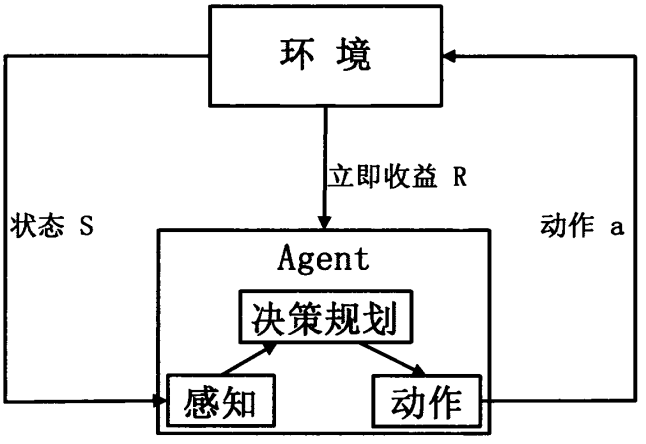


图2-1　MDP 基本模型

1. 状态

状态是在某个时间点，智能体对世界（系统）的理解和描述。在不同的系统中，对状态的具体定义并不一样。但是，一般在 MDP 建模时，状态必须包括所有系统中智能体能够利用，对智能体的决策会产生影响的信息。在建模过程中，某些因素要不要加入到问题的状态表示中，或者需不需要对一些状态量进行某种拆分或合并也是至关重要的。处理得不好，会引入大量的冗余信息，增加问题的复杂度，极大地增加决策规划的求解的难度。

状态的表示，最一般的是平铺式的表示方法。对所有可能的世界状态给以标号，以$s_1,s_2,s_3,\cdots$的方式表示。这种表示方法，标号的数目就是状态空间的大小。对于状态空间极大甚至是连续状态空间问题，这种表示方法不够实用。而一种更加自然的表示方法是因子式的表示。用固定维度的多元组表示影响决策的状态因素，每一个状态就是由这些因子组成的多元组[9]。本文的主要工作是基于因子化的表示方法的。

图 2-2 表示一个离散的、随机的动态系统。对于图中每个随机变量S^t，有条件概率$\Pr(S^t|S^0,S^1,...,S^{t-1})=\Pr(S^t|S^{t-1})$。就是说$S^t$只是概率依赖于$S^{t-1}$，而与$S^{t-1}$之前的状态都无关，这也体现了马尔可夫性质。

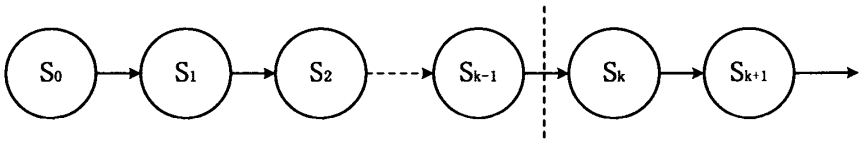


图2-2　马尔可夫链示例

这里我们引入吸收状态（absorb state）的概念：如果对于某个状态s，不论执行任何动作，过程都以概率 1 转移到s本身，则称该状态s为吸收状态。

2. 行动

智能体的行动作用于世界，引起世界状态的改变。MDP 中一个关键的部分是提供一个智能体用于决策的行动集合。当集合中的某个行动被执行时，世界状态根据一个状态转移的概率分布随机地转换为另一个状态，这个概率分布和所执行的动作有关。

在 MDP 中，某一状态下可执行的动作可能很多，但是有的行动可能对某状态没有影响或者是有负面的影响，这和经典人工智能规划问题不同。在经典人工智能规划中，我们可以根据模型的先决条件来决定哪些行动对该状态有益，这样，决策时可能有很多动作不需要考虑，计算时的后继状态空间也相对减小。在 MDP 中，我们也可以借鉴这个特点，在决策规划时尽可能缩小一个特定状态的行动空间，加快决策规划速度。

本文在不加说明的情况下，讨论的都是时齐 Markov 决策过程，即所有行动的执行时间都是相同的，状态转移的时间间隔也是一致的。这种行动有时也被称为原子动作。原子动作不能再分解成更小的动作。比如，在 RoboCup 2D 环境下，球员智能体在固定的 100ms 决策和动作周期里，可以选择以一定的角度转向，或者以一定的加速度前进，以一定的力量踢球，这些构成了该系统下智能体的原子动作。

3. 状态转移函数

状态转移函数描述了系统的动态特性。在确定环境下和不确定环境（随机环境）下的行动的特点，可以作如下比较：

- 确定性环境下的行动：$P:S\times A\to S$

在某个世界状态s下执行动作a可以得到一个确定的状态；

- 随机环境下的行动：$P:S\times A\to \Pr(S)$

在某个状态s下执行某一动作a，得到的是状态的一个概率分布$P(s'\mid s,a)$，也可记为$P^a(s,s')$。

下图 2-3 给出了对某一个给定行动，状态之间概率转移的情况。在简单的 MDP 问题中，状态转移可以用表的形式表示。

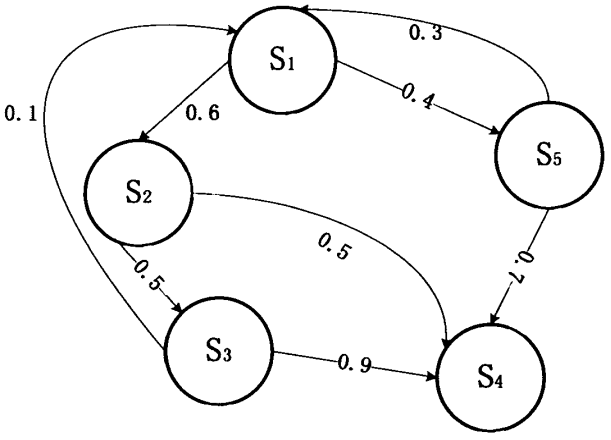


图2-3　给定行动的状态转移图

4. 报酬函数

报酬函数，也称收益函数、回报函数，表示在一个状态下执行一个动作，状态转移到另一个状态所得到的的立即收益。我们希望智能体在决策时能够按照某个标准来选择行动使得可以获得的长期收益最大化[11, 29]。例如有现阶段最优准则，使智能体最大化有限阶段的总收益，即 $\max E[\sum_{t=0}^{k-1} R_t]$，其中 R_t 是智能体在第 t 步得到的报酬。这个准则要求清楚地知道决策次数 k，要得到更理想的结果，应该考虑决策周期为无限阶段。考虑整个过程智能体的总收益，一般要引入一个折扣因子 $\gamma, (0<\gamma<1)$，以保证最大化的总收益 $\max E[\sum_{t=0}^{\infty} \gamma^t R_t]$ 收敛。

2.1.2 值函数和策略

上节介绍了 MDP 的基本定义，和各要素的概念。这里介绍一些 MDP 决策求解过程中需要用到的概念。

策略（Policy）就是决策问题的解。它是从 MDP 模型的状态集合到动作集合的一个映射，即 $\pi: S \to A$。按照某一策略 π 解决问题的步骤是：首先智能体观察到当前世界的状态 s，根据策略相应的动作 $\pi(s)$ 执行，进入下一状态……如此重复直到问题解决。显然，这个过程要求智能体可以完全地观察到状态，这也是标准 MDP 的一个基本假设。我们将动作的选择只跟当前的状态有关，而与决策的时间无关的策略称为平稳策略。而将选择动作时不仅考虑当前状态，还要考虑决策时间的策略，称作非平稳策略。例如，在机器人足

球比赛中，将近终场时如果我方落后，则采用比之前相同态势下更激进的策略，就是非平稳策略。

对于一个策略，我们用执行这个策略所能获得的长期期望报酬来评价策略的优劣。定义值函数（Value Function）来表示在某个初始状态下，执行某个策略获得的期望回报。在状态s执行策略π的值函数$V^{\pi}:S\to R$：

$$V^{\pi}(s)=E[\sum_{t=0}^{\infty}\gamma^{t}R(s^{t},\pi(s^{t}))] \tag{2-1}$$

其中，s^{t}为时刻t环境所处的状态。用递归的形式表示的值函数就是：

$$V^{\pi}(s)=R(s,\pi(s))+\gamma\sum_{s'\in S}P^{\pi(s)}(s,s')V^{\pi}(s') \tag{2-2}$$

对于一个策略π，对应的值函数V^{π}是一系列线性方程的唯一公共解（每个状态s对应一个方程）。通过值函数的定义，给定一个策略，可以计算出它对应的值函数。但是同时，我们也需要知道怎样从给定的值来计算相应的策略。为了这个目的，我们首先定义一个求解过程中经常用到的中间变量，行动值函数（Action Value Function）$Q^{\pi}:S\times A\to R$来表示在状态s执行行动a，而其他状态执行策略π的期望回报：

$$Q^{\pi}(s,a)=R(s,a)+\gamma\sum_{s'\in S}P^{a}(s,s')V^{\pi}(s') \tag{2-3}$$

当只有值函数V，而没有策略的显式记录时，记行动值函数为Q。则策略可以通过下式计算得到：

$$\pi(s)=\arg\max_{a\in A}Q(s,a) \tag{2-4}$$

即：

$$\pi(s)=\arg\max_{a\in A}\{R(s,a)+\gamma\sum_{s'\in S}P^{a}(s,s')V^{\pi}(s')\} \tag{2-5}$$

同时，有：

$$V^{\pi}(s)=\max_{a\in A}\{R(s,a)+\gamma\sum_{s'\in S}P^{a}(s,s')V^{\pi}(s')\} \tag{2-6}$$

式（2-5）事实上用的是一步前瞻的贪婪策略，这样获得的策略我们称之为贪婪策略。

我们定义一致性条件（Monotonic Condition）如下，对于所有状态s，有

$$V(s)\le\max_{a\in A}[R(s,a)+\sum_{s'\in S}P^{a}(s,s')V^{\pi}(s')]$$

如果值函数不满足一致性，即在某一个系统状态s处，有$V(s)>\max_{a\in A}[R(s,a)+\sum_{s'\in S}P^{a}(s,s')V^{\pi}(s')]$，就无法从式（2-5）确定出一个满足$V^{\pi}(s)\ge V(s)$的策略$\pi(s)$。相反，当值函数满足一致性条件，$\pi$即是对当前值函数对应的策略的改进，

有$V^{\pi}(s) \geq V(s)$。一般，一个满足一致性条件的值函数，只按照式（2-6）的 Bellman 公式进行迭代更新的话，一致性条件始终能够成立。

将最优策略记为π^{*}，它对应的值函数记为V^{*}，称为最优值函数。通常，对于所有状态s，一个策略π满足$V^{*}(s)-V^{\pi}(s) \leq \varepsilon$时，称策略$\pi$为状态$s$处的$\varepsilon$最优策略。当$\pi$对所有状态都满足上述条件，即称其为问题的$\varepsilon$最优策略。

2.1.3 半马尔可夫决策过程

一个离散时间的 Semi-MDP 是一个 MDP 的一般化形式，在 Semi-MDP 中，动作的完成时间可以是一个变化的量，而不一定是一个固定的周期[30]。特别地，用一个随机变量N表示动作a在状态s执行时所需要的时间步数。可以将状态转移概率函数扩展为在状态s执行动作a，经过N步到达结果状态s'的联合概率分布$P(s', N \mid s, a)$，与此相似，期望回报可以改为$R(s', N \mid s, a)$[19]。

一个固定策略π的值函数可以用一个修改的 Bellman 公式来定义：

$$V^{\pi}(s)=\sum_{s', N} P(s', N \mid s, \pi(s))\left[R(s', N \mid s, \pi(s))+\gamma^{N} V^{\pi}(s')\right] \tag{2-7}$$

该公式与标准 MDP 的值函数的唯一区别是等式右边同时考虑了s'和N，在折扣因子γ上加了N次方来表示报酬折扣。

注意到期望算子是个线性算子，可以将 Bellman 公式写成执行动作a的期望报酬和结果状态s'的期望值函数的和。上式可以重写成：

$$V^{\pi}(s)=\bar{R}(s, \pi(s))+\sum_{s', N} P(s', N \mid s, \pi(s))\, \gamma^{N} V^{\pi}(s') \tag{2-8}$$

其中，$\bar{R}(s, \pi(s))$是在状态s执行动作$\pi(s)$的期望报酬，且期望值是相对于s'和N得到的。

2.2 Markov 决策过程经典求解方法

MDP模型用来表示客观世界决策规划问题的数学模型，其相关的求解算法按是否求解状态空间的全部进行划分，可以分为后向迭代算法，如早期的值迭代和策略迭代；以及前向搜索算法，如AO*，LAO*[31]。后向迭代算法采用动态规划，在模型的所有状态中搜索最优策略。而前向搜索算法一般只求解从给定初始状态开始的最优策略，同时利用状态可达性信息，通常可以避免大量不必要的计算，有更高的求解效率。其他还有Heuristic Search/DP (HDP)[32]，Envelope Propagation(ZP)[33]以及Focused Dynamic Programming(FP)[34]等算法。

从另一个角度，按照算法的实时性分，可以分为离线算法和在线算法。实际上，对于大多数实际应用中的大规模问题，不论有没有利用状态可达性，都不可能用离线方式一次性求出问题的解[29]。这种情况使用在线算法更适合，也称实时算法。实时算法在交互过程中交替的进行决策计算和动作执行，且通常随着时间的推移，解的质量不断提升。实时动态规划（RTDP）算法是最早的基于动态规划的实时算法[35]。该算法通过不断地模拟尝试（trail）来改进策略，每次尝试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，最终能够收敛于最优解。然而 RTDP 不处理停止问题，即不判断当前解是否已经满足要求而可以停止对该状态的计算。近年来有很多改进的算法，如 Labeled RTDP[24]，BRTDP[36]及 FRTDP[37]等。下面，介绍几种典型的 MDP 求解方法。

2.2.1 后向迭代算法

值迭代和策略迭代算法均是基于动态规划（DP）的算法[24]。

1. 值迭代

值迭代算法中，策略没有显式的表示，过程按照动态规划的 Bellman 公式不断进行迭代更新，来改进值函数。

$$V(s)=\max_{a\in A}\{R(s,a)+\gamma\sum_{s'\in S}P^{a}(s,s')V^{\pi}(s')\} \tag{2-9}$$

值迭代的算法步骤如下表 Alg. 2-1 所示：

表2-1　值迭代算法

	Alg. 2-1：值迭代（**Value Iteration**）
1	**Start**
2	V：initialization；ε：initialization；
3	**Repeat**
4	Update Value Function based on the following equation： $V(s)=\max\{R(s,a)+\gamma\sum_{s'\in S}\Pr(s,s')V(s')\}$；
5	Convergence test： **if**（$V'-V\le\varepsilon$） **goto** line 7； **else** **goto** line 4；
6	**End**（**Repeat**）
7	**Return** Optimal Policy；
8	**End**（**Value Iteration**）

当值函数接近最优时，策略可以通过式 $\pi'(s)=\arg\max_{a\in A}Q^{\pi}(s,a)$ 反求得到。值函数是否达到最优一般是通过一个有界误差来衡量的。误差界限通过定义 Bellman 误差或称残差

（residual）：$r: r = \max_{s \in S} |V(s) - V'(s)|$。对于所有状态$s$，Bellman 误差足够小时迭代停止。而在折扣 MDP 中，误差的界限有折扣因子γ和最大 Bellman 误差确定。

2. 策略迭代

策略迭代中，策略显式的表示，可以得到策略对应的值函数V^{π}，然后使用下面公式改进策略：

$$Q^{\pi}(s,a) = R(s,a) + \gamma \sum_{s' \in S} P^{a}(s,s') V^{\pi}(s') \tag{2-10}$$

$$\pi'(s) = \arg\max_{a \in A} Q^{\pi}(s,a) \tag{2-11}$$

迭代过程总是在不断地改进当前的策略，而策略的总数目是有限的，这样，算法在经过有限步的迭代之后，总能收敛于最优策略。策略迭代的算法如下表 Alg. 2-2 所示：

表2-2　策略迭代算法

	Alg. 2-2：策略迭代（**Policy Iteration**）
1	**Start**
2	*Policy* π：initialization；
3	**Repeat**
4	Value Evaluation，calculate V of policy π using the following equation： $V(s,\pi) = R(s,\pi) + \sum_{s' \in S} \gamma \Pr(s,\pi,s') R(s',\pi)$；
5	Update Policy using Bellman equation；
6	Convergence test： **if**（$\pi' = \pi$） **goto** line 8； **else** **goto** line 4；
7	**End**（**Repeat**）
8	**Return** Optimal Policy；
9	**End**（**Value Iteration**）

2.2.2 前向搜索算法

1. 基于与或图的启发式搜索算法

一般情况下，一个状态空间上的策略搜索问题与 MDP 类似，可以将其定义为一些列从初始状态到终止（目标）状态的集合，一些列行动以及报酬函数。问题的目标是找到一个从起始状态到终止状态的最大收益或最小花费的策略路径。搜索基于的树或图的数据结构来看，可以用与或图来表示。

经典人工智能中经常提到与或图的概念，用来对问题进行规约，能够方便的用一个类似树的结构把问题规约为几个后继问题的替换集合。比如，一个问题 A 可以单独地通过

解决问题 B 来解决，也可以将其分解为两个问题 C 和 D，进而分别解决，还可以分解为三个不同的子问题 E、F 和 H 来分别解决，如下图所示，图中的 X，Y，Z 三个节点被称为或节点，它们具有问题描述的作用；其他节点都是父节点规约的子问题集合中的元素，被称为与节点。通过与或图，把某个单一问题规约符具体应用于某个问题的描述，依次产生出一个中间或节点及其节点后裔。

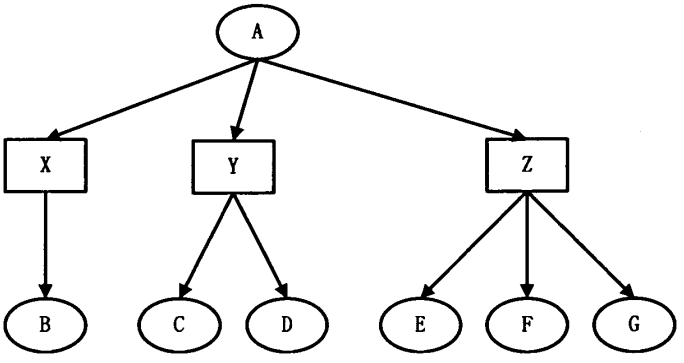


图2-4　与或图示例

实际上，上图还可以用来描述一个存在不确定性的 Agent 决策问题。根节点表示 agent 决策的初始状态，X，Y，Z 是或节点，表示可以选择的行动，其余节点表示在执行父节点对应的行动后 agent 可能达到的状态，对应一个概率。

2. 实时动态规划

实时动态规划（Real-time Dynamic Programming，RTDP）算法是 Barto et al.于 1995 年提出的，它是一种基于前向搜索的技术，避免了搜索状态空间的全部。

RTDP 是一种简单的 DP 算法，包括一系列尝试（trials）和运行（runs）。它们都从初始状态 s_0 开始，终止于目标状态。表 Alg. 2-3 是 RTDP 算法的总结。

每次 RTDP 尝试都是在用下式（2-12）对于所有访问的状态 s 更新至 V(s)时模拟贪婪策略 π_V 的结果：

$$V(s) := \max_{a \in A}[R(s,a) + \sum_{s' \in S} P^a(s,s')V^{\pi}(s')] \qquad (2\text{-}12)$$

因此，RTDP 是一种异步值迭代算法，每次迭代中只选择更新一个状态。状态的访问更新根据一个状态相应的转移概率选择后继状态。这样的运行方式很有效，它使得 RTDP 与一般的纯贪婪搜索和异步值迭代不同。RTDP 不需要评估整个状态空间，可以很快的得

表2-3　Alg. 2-3 RTDP 算法

```
    Alg. 2-3 RTDP（s: state）
1   Start
2     | Repeat  RTDPTrial（s）
3   End（RTDP）

    RTDPTrial（s: state）
4   Start
5     | While !s.GOAL() do
6     |   | //pick best action and update
7     |   | a = s. GreedyAction()
8     |   | s. Update()
9     |   | //stochastically simulate next state
10    |   | s = s. PickNextState（a）
11    | End（While）
12  End（RTDPTrial）
```

到较好的策略，然后不断改进。而且具有良好的 Anytime Behavior 特性，例如，它可以很快的得到一个较好的策略，然后随着时间的推移平稳地改进策略。

IDA*这样的经典启发式搜索算法可以解决有数以亿计甚至更多状态的最优化问题[38]，因为它用到了下界（可接受的启发式函数 admissible heuristic function），因而可以避开计算状态空间中的大部分状态。RTDP 是第一个能够在一般不确定环境中有相同性能的 DP 方法。当然，在两种方法中，启发式函数的质量对于算法性能是至关重要的，其实，很多基于经典启发式搜索的工作都旨在找到新的更有效的启发式函数[39]。LAO*[31]是一个有这个特点的较新的启发式搜索 DP 算法。

上面我们介绍了MDP的后向迭代和前向搜索两大类最基本的求解算法。后向迭代方法相对于状态空间，行动空间和求解精度通常具有多项式时间复杂度。前向搜索算法由于在求解具体问题时利用了状态可达性的信息，一般具有更高的效率[40]。

2.3　大规模不确定 Markov 决策过程问题

Markov 决策理论研究的其中一个主要热点是不确定性规划问题。大规模不确定性问题本身并没有很严格的定义，一个决策规划问题的规模受到其状态空间、行动空间、规划步数以及状态转移特性等因素的影响。在通常情况下，我们说的大规模规划问题一般是指那些接近现实世界的应用。

Littman 作为 Markov 决策领域的奠基人之一，在[41]有过这样一段描述。在不确定性

规划领域，近些年来学者们提出了大量的形式化的模型及相应的算法。这些模型在描述现实世界的问题上是很有用的，这些算法也显示了在解决此类问题中是很有前途的方法。但是这些研究的一个很重要的缺失是，没有真正地与现实世界应用的接触。很多算法都放在一些“证明概念”类的问题中进行测试，并没有太多信息能够说明算法如何向上扩展。现在是真正做出认真努力，将这些新生的技术应用到现实世界问题的一个关键时期；只有与实际应用结合才有希望将这一领域的研究集中在最富有成效的方向上。

现阶段，分层的设计是在大规模规划问题中应用最多的方法[19, 20]。广泛认为层次分解方法是解决超大规模规划问题的关键[42]。分层方法（hierarchical Approaches）的目标是降低问题的复杂度[18]。其核心是状态抽象（state abstraction）和时延行为（temporally -extended activities）的使用[29]。

分层的方法将一个给定的 MDP 分解成一系列按层次结构组织的子 MDPs，表示为 $\{M_0, M_1, M_2, ..., M_n\}$。通过递归地规划各层的最优策略，最终得到解决问题的在该层次结构下最优的分层策略。方法的详细介绍将在第四章给出。

RoboCup 2D 是目前研究不确定性规划问题的一个较好的平台，其规模很大，具有连续的状态和动作空间，队友和对手的影响，不亚于很多现实应用问题的复杂度。本文结合任务分层的思想设计足球机器人智能体的策略规划算法，实现使 Markov 决策理论能够更好地应用于现实世界问题也是本文的一个重要目标。

2.4 本章小结

本章首先介绍了MDP的基本模型，分析了几种经典的求解方法。这些经典求解方法一般只能解决较小规模的MDP问题，对于本文研究的大规模实时决策问题，并不适用。因此，本文要在经典MDP求解方法的基础上，寻找适合规模较大的决策问题的在线求解方法。这将在后面章节详细的介绍。

第3章 基于效用预测的球员策略选择

集中式的控制方式由于规划相对简单，在条件允许的问题中，用集中式的控制方式进行规划将得到良好的效果。FIRA 2D 仿真平台 SimuroSot 即是用的集中式控制方式。本章以此平台为对象，研究集中式控制下的智能体决策方法，以达到协调合作地完成任务的目的。通过基于任务层次分解的决策框架，提出了一种基于效用预测的决策方法。

本章采用基于 MDP 任务层次分解的决策框架，在基于角色的决策系统基础上，提出了一种基于行为效用预测的机器人足球比赛进攻情形下的智能体角色实现方法。首先介绍机动区域的概念，基于此对足球机器人采取角色动作后的效用进行预测，进而结合博弈论的有关概念和方法，选择机器人的角色动作，实现足球机器人之间的协调配合。

3.1 问题来源

目前对多智能体系统的研究热点主要在分布式控制领域，但是实际上有很多现实系统，特别是通信条件良好的系统可以使用集中式的控制方式。如教育及娱乐系统、清除危险区域、智能家居机器人等领域。这些系统中，各智能体有自己的决策能力，但是可以通过一个主要决策者来分配任务。相对于完全分布式的控制，以较高的通信花费来换取各智能体的协调性，降低误协调的发生。在有较好通信条件的系统中，集中式控制有很大的应用空间。

在对这方面已有很多研究成果，研究者们提出了不少的解决方法。其中研究较多且效果较好的是基于动态角色转换的机器人足球策略方法[16, 43-45]。角色的动态转换和任务分配是基于角色的策略系统研究的重点，许多研究者都提出了不少方法。Daniel Playne[46]提出基于知识的角色分配方法，根据机器人对于各角色的信心因数以及角色优先度信息来分配角色；黄波等[47]提出基于优度值的任务分配方法，等等。这些方法都是比较有效的智能体角色分配方法。陈建平等[48]以机器人到任务目标点的时间为效用，来分配角色任务。该方法在设计效用函数时仅考虑了球员到任务目标点的时间花费，忽略了较多的有用信息。章小兵等[49]将可传值和安全性相结合来设计效用函数，并依此选择传球策略。该算法比较有效，但是由于只考虑了当前比赛形势，因而无法有效把握行

为动作对比赛形势的影响。

对于机器人足球比赛这种动态环境，如果只根据某一时刻的静态场上信息，显然不足以准确地估计形势，确定策略。如果能对形势做出一定的预测，对策略的制订将很有帮助。在这方面，一些学者也做了一定的工作。[50]提出了结合 IMBBOP 方法和 MMTB 方法进行行为预测的多智能体协作模型。该方法对世界模型的精确性以及个体球员对队友的感知状态有较高要求。[51]提出用抛物线预测模型对被测物体的位置进行预测，进而对场上物体的状态进行短期预测，协助决策的方法。这是一种比较有效的行为预测方法，但是由于只是做较短期的预测，对提高球员动作成功率有较大帮助，而对于估计整体比赛形势的作用有限。

很多研究者将博弈论的方法应用于多智能体系统中，并取得了不错的成果。Adel Ghazikhani et al.[52]提出了一种用合作博弈论方法产生多智能体博弈联盟的方法。文章提出了一种新颖的博弈者模型，采用迭代算法剔除对联盟贡献小的博弈者，最终得到最优的智能体联盟。实验结果表明算法比较有效。尽管目前国际上将博弈论在足球机器人这个问题中应用的研究还很少，但是博弈论应用于多智能体系统中的效果已经得到了众多研究者的验证[53, 54]。

本章在对以上各种方法比较的基础上，在基于 MDP 任务层次分解方法的基础上设计球员决策框架。提出了一种基于行为效用预测的足球机器人角色实现方法。该方法在基于角色的决策系统基础上，重点对机器人足球比赛进攻情形进行了研究，提出了辐射区域的概念，并基于此预测足球机器人进行一步行为将带给球队的效用，进而应用博弈论的有关理论和方法，对足球机器人的行为动作进行合理的选择。

3 .2 决策系统基本框架

在讨论足球机器人角色实现问题之前，首先简要介绍一下决策系统的总体框架和基本处理流程，以及相关的博弈论基本概念，以备后续分析和讨论之用。

3.2.1 机器人足球决策系统

足球机器人决策系统与人类的许多群体活动的决策过程类似，如战争、体育比赛和商业活动等。都需要决策者根据形势确定攻防阵型，安排相应角色执行阵形中的某些任务，同时要求各角色有理性判断的能力，以较好的完成任务，实现协调，从而保证总目标的实现。

本章采用三层 Agent 体系结构设计足球机器人决策系统[55]，其体系结构框图如图 3-1 所示。将足球机器人决策系统分为三个层次，即：全局层、角色层和动作层。传感器得到的信息首先用来更新世界模型，全局层主要进行比赛态势分析和阵型决定；角色层的主要工作是球员角色的分配和实现，以及处理不同角色之间的通信任务；动作层主要是球员的可选动作库。决策系统在上层决定好球员行为后，调用动作库中的基本技术动作。

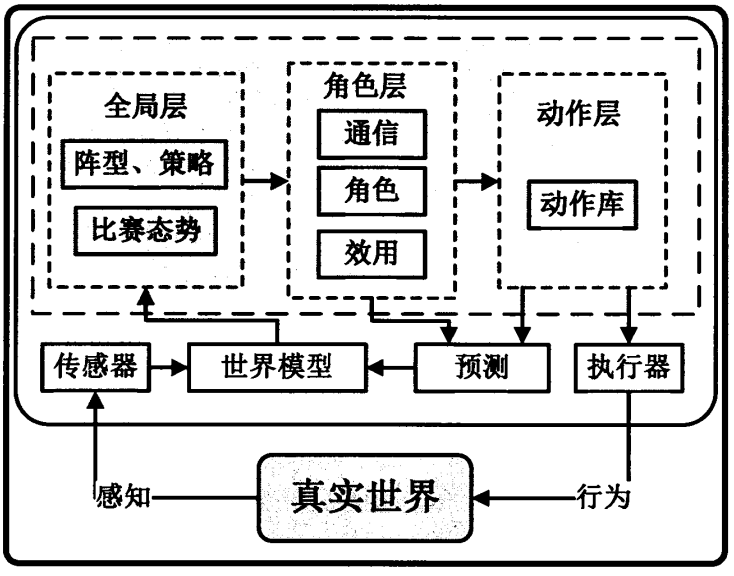


图3-1 决策系统体系结构

总的来说，足球机器人决策系统的处理流程如图 3-2 所示。由于我们研究的重点是决策系统，所以我们假设视觉系统已得到较精确的场上物体位置信息。

决策系统首先对获得的双方球员及球的位置信息进行处理，获得用户数据，包括控球方信息，球员相对位置和角度，球员相对于球的位置和角度，球员和球的速度等，并以此更新世界模型；然后根据定义的球场分区信息，结合球员及球的相对位置和速度信息评估比赛形势，确定我方的攻防策略，并选择合适的比赛阵形；根据球员距离阵形目标点的相对距离及角度信息分配球员角色；在这之后的任务就是角色的实现问题。角色实现包括两部分内容，即角色行为动作的选择和技术动作的执行，各角色球员根据一定的方法自主协调地选择合适的角色动作，以实现团队协同作战，然后进行路径规划，由执行机构执行底层动作。这里研究的主要内容是球员角色行为动作的选择机制。

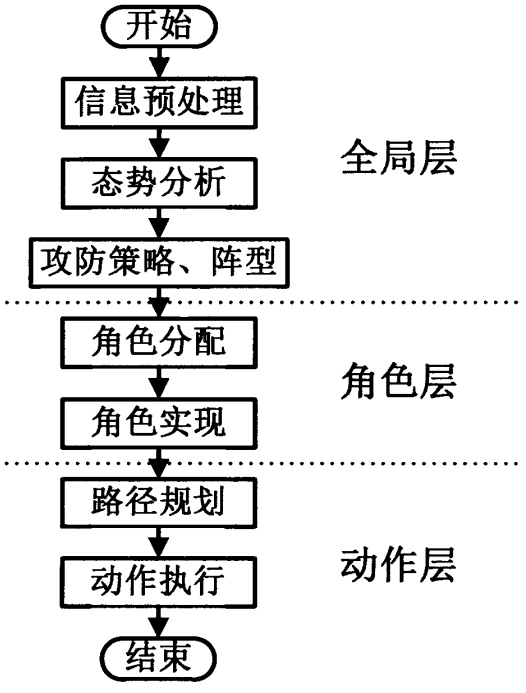


图3-2　决策系统处理流程

3.2.1　博弈论基本概念

在多决策主体之间存在利益冲突时，当事人所进行的行为选择，称为博弈。它是根据信息分析及能力判断，研究多决策主体的行为发生直接相互作用时的决策以及这种决策的均衡，以使收益或效用最大化的一种对策理论。

一个博弈由四个基本要素构成，即，博弈参与者、博弈策略、博弈结局和博弈效用。博弈各种解的概念是建立在两个重要假设的基础之上的：理性（Rationality）假设和共同认识（Common Knowledge）假设。

博弈问题的核心是博弈求解，纳什均衡是非合作博弈最基本的解，定义如下：

在一个 N 人博弈中，策略组合 $s=(s_1,s_2,\cdots,s_N)$ 构成一个纳什均衡，当且仅当：对于每一个博弈者 $i\,(i=1,2,\cdots,N)$，其策略 s_i 是对策略组合 s 中的其他所有博弈者策略 s_{-i} 的最优回应，即对任意 $s_i'\in S_i, u_i(s_i,s_{-i})\geq u_i(s_i',s_{-i})$.

可见，在基本非合作博弈的框架下，决策者追求自身效用的最大化，而在我们所面临的机器人足球比赛环境下，单个个体效用的最大化并不能够保证总体目标的完成，即胜利。因此，我们在用非合作博弈理论追求团队集体效用最大化的同时，也需要应用合作博弈的相关理论解决效用在团队中的分配问题，即各个球员具体行为动作的合理选择。

合作博弈论关注存在相互依存关系的各博弈者形成不同的集团（合作）会带来的结果.它抽象掉了非合作博弈中的策略，直接讨论博弈者效用的分配问题[56, 57]。限于篇幅，此处不再赘述。

3.3 基于一步行为效用预测的角色实现

角色实现即在角色分配好之后，各角色球员选择合适的角色动作和行为，相互协调地执行出技术动作，实现有效的配合。本章首先讨论辐射区域的概念，继而提出了一种通过预测角色动作执行后的效用，然后应用合作博弈方法决定各角色应选动作的足球机器人角色实现方法。

3.3.1 辐射区域

文献[58]提出了辐射区的概念，与本章要用到的辐射区域（影响范围，Influence Area）的概念有相似之处。但其文中并没有考虑辐射区与时间的关系，而且文中并没有有效地利用这个概念。本章提出的方法是在辐射区域概念的基础上实现效用预测的。

辐射区域，指的是在一段特定的时间内，某一个体能以较大概率到达的空间位置，即活动范围。如图 3-3 中智能体周围灰色椭圆框所示。辐射区域由如下函数表示：

$$IA = f(t, x, y, |\alpha|, v) \quad 0^\circ \le \alpha \le 180^\circ \tag{3-1}$$

其中，t 表示个体选择一动作，如传球从 A 点到 B 点，所需要的时间；x，y 为个体坐标位置，α 为空间某一点与智能体的连线与智能体正面法线方向之间的夹角；v 表示智能体移动的最大速度。需注意的是，机动区域的最远端并不是个体最大速度与动作时间的乘积，而是需要乘以一个系数，表示概率的大小。系数的选取视经验和个体运动特性而定，须防止由于计算出的辐射区域过大从而使我方球员无法选择合适动作路径的情况。

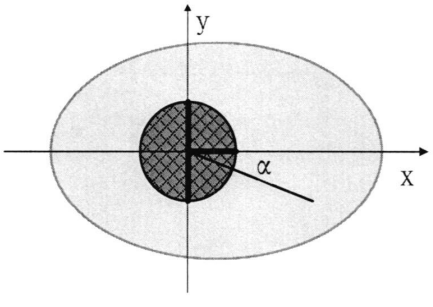


图3-3　辐射区域示意图

辐射区域的概念，克服了利用智能体在场上的绝对位置，很难对比赛形势进行准确估计，更难于进行有效预测的缺陷，在一个角色动作时间范围内将对方球员看成一片静止的障碍物，由此进行形势评估和效用预测将更加准确。从而最大限度地降低了由于无法精确预知对手行为而带来的决策失误。

3.3.2 效用函数的设计

效用（Utility）在经济学、博弈论、运筹学，以及多智能体系统中是一个统一的概念。它是基于智能体可以自主地估计行为的价值这一概念，通过比较行为集中各行为的效用大小来选择行为动作。

在博弈论中，主体的决策遵循效用最大化的原则，而在机器人足球比赛环境下，个体效用的最大化并没有决定性的意义。因此，在设计效用函数时不仅要设计单个智能体的效用函数，同时要把集体效用函数与个体效用函数联系起来设计。将足球机器人角色实现的问题转化为团队追求集体效用最大化,以及集体效用在各个体之间合理分配的问题。

在合作博弈论中，效用函数的设计需要满足以下性质：保证成功、集体（联盟）效用最大化、帕累托效率 、个体理性、稳定、简洁、分布式等。本文依据这些要求来设计效用函数。

基于以上分析，本章将进攻情形下智能体效用函数的设计分为三个层次，即，策略层（Tactic Layer），角色层（Role Layer）和动作层（Action Layer）。策略层效用表示协同动作集合对团队进攻的效用，角色层效用表示个体角色的位置效用以及个体各种动作的成功率和该动作对团队进攻的贡献，动作层效用表示具体角色动作的效用。

各智能体由于角色不同，可选的动作也不同，例如，无球队员就不能选择射门、带球等动作。所以，首先声明各角色的可选动作：

前锋：射门、带球、传球；

助攻（左前锋、又前锋）：接球、跑位、挡拆；

后卫：防守。

由于各角色可选动作之间有一定的关联，例如，带球的前锋将球传给左前锋，则左前锋必须选择接球，右前锋可选择跑位或者挡拆，而后卫主要负责防守。由此，可以将角色集体行为分为有限类别，每一个类别组合就是一个团体策略。例如，以图 3-4 所示进攻场景为例进行分析，定义团队策略类型：

Ⅰ. YA 传球给 YB，YB 接球，YC 跑位或挡拆，YD 防守；

Ⅱ. YA 传球给 YC，YC 接球，YB 跑位或挡拆，YD 防守；

Ⅲ. YA 射门，YB、YC 跑位或挡拆，YD 防守；

Ⅳ. YA 带球，YB、YC 跑位或挡拆，YD 防守；

etc.。

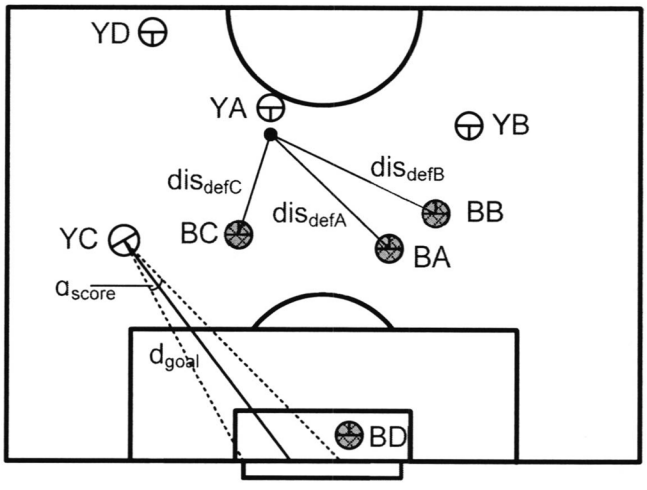


图3-4　黄队（Y）进攻场景示例

不同角色球员可选动作有限，这也在一定程度上降低了效用函数计算的花费。接下来，设计效用函数。

首先，设计球队的集体策略效用函数。策略效用表示球队在某一特定策略（团体动作）下的总效用，不仅包括该策略下各个体的效用，还包括该团体动作给球带来的效用，即促进得分的效果。定义策略效用函数如下：

$$U_{tactic} = \omega_p \sum_{i=1}^{n} \beta_i U_{role\,i} + \omega_b U_{ball} \tag{3-2}$$

其中，$U_{role\,i}$ 是标号为 i 的智能体的个体角色效用，U_{ball} 为球的效用，ω_p 为角色效用在集体效用中所占比例的权重，ω_b 为球的效用权重，β_i 是各角色效用的归一化系数。

个体角色的效用 U_{role} 由两部分构成，即个体站位的效用和个体动作的效用。体育比赛中个体的位置是至关重要的，好的位置对攻防有极大的促进作用。角色效用 U_{role} 定义为：

$$U_{role} = \lambda_1 U_{position} + \lambda_2 \sum_{j=1}^{k} U_{action} \tag{3-3}$$

其中，$U_{position}$是个体的位置效用，定义如下：$U_{position}=h_1(dis_{ToGoal},D_{open})+h_2(No._{def},dis_{Def})$，$h_1(\bullet)$表示个体与对方球门的距离和开阔角度，$h_2(\bullet)$表示个体所面临的防守状况，防守人个数和距离。

U_{action}表示智能体的动作效用，不同角色个体的可选动作不同，对不能选择的动作，效用值为零。动作效用的定义见后文。

U_{ball}是球的效用函数，表示进攻效果的好坏，由两方面因素决定，球相对于对方球门的位置和面对的防守状况。所以，能使球尽量向对方球门移动并使球面临的防守较弱的动作的U_{ball}值就较高。显然，当球进入对方球门，U_{ball}的值最高。

$$U_{ball}=\xi_1 g_1(d_{goal},a_{score})+\xi_2 g_2(No._{def},dis_{def}) \tag{3-4}$$

其中，$g_1(\bullet)$是球与对方球门的相对位置的函数，表示球是否接近对方球门的效用，由距对方球门距离和球门开阔度有关，如图 3-4 所示；$g_2(\bullet)$表示球所面对的防守情况，由球面对的防守人数量以及防守人与球的距离决定，如图 3-4 所示，对球有威胁的防守人数$No._{def}=3$，与球的相对距离分别为dis_{defA}、dis_{defB}和dis_{defC}。

最后，是动作层效用的设计。动作效用由两部分组成，动作安全性和该动作完成后对团队集体效用的贡献。安全性保证动作的成功率，对集体效用的贡献表示该动作能增加团队集体效用的性质，如跑位可以使自己获得更佳的进攻位置，帮助球队得分；挡拆可以拉扯防守队员，给前锋更大的射门空间。动作效用函数设计如下：

$$U_{action}=\eta_1 f_1(d,\theta)+\eta_2 f_2(aid_{team}) \tag{3-5}$$

其中，$f_1(d,\theta)$表示到动作目标点的距离和轨迹开阔度决定的动作安全性；$f_2(aid_{team})$表示动作对团队效用的增加，如对于射门动作，对团队效用的增加就是得分。

3.3.3 角色动作选择

本章提出的基于一步行为效用预测的角色动作选择方法，克服了[48]对世界模型和队友信息的较高要求以及[47]由于只是做较短期的预测，对于估计整体比赛形势的作用有限的缺点。本章提出的方法是基于智能体辐射区域进行预测的，不需要重点考虑动作安全性的问题，因此预测精度良好。其方法是，从带球的前锋的动作选择发起，如 3.2 节中提到，将每类团队动作类型执行后的预测效用进行比较，选择对球队最有利的策略类型。

如前文所述，该角色动作选择方法的执行分三步进行，具体步骤如下：

第一步，分别计算带球前锋执行一步可选动作的时间 t，并以此计算对方球员在时间 t 内

的防守区域；

第二步，分别计算前锋各种可选动作执行一步后我方各角色效用值、球的效用值，再计算团队策略效用值，以集体效用最大化的原则选择策略类型；

第三步，根据各角色球员角色效用值的大小按下面（3-6）式选择各角色动作。

$$U_{opt} = \max\{U_{role\,i}\} \tag{3-6}$$

其中U_{opt}是最终选择的角色动作下的角色效用。

注意，第二步中球的效用值是指前锋执行它的一步可选动作后球的实际效用，而其他的效用值均是此动作后的预测效用值。

下面结合图 3-5 所示进攻场景阐述角色动作选择的具体方法。首先计算动作类型 II 执行后的效用。假设 YA 传球到 YC 所需时间为 t，将 t 带入式（3-1）中，计算各防守队员的防守区域，如图中灰色框所示；分别计算各球员可选动作的动作效用，继而计算各角色效用和球的效用，求出集体策略效用。用同样方法求出动作类型 I 、III、IV的策略效用，选择策略效用最大的动作类型作为团队行为。

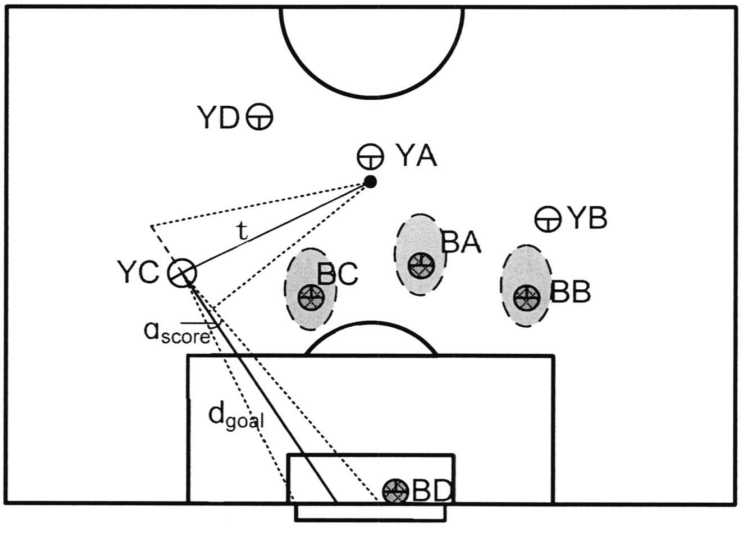


图3-5　进攻场景实例

为了避免无谓的传球和带球等动作，可设置一个阈值，在动作类型的集体效用相差不超过阈值时，以得分优先的原则选择动作类型。

动作类型选定后，各角色的可选动作也就确定了，例如，对动作类型 II 来说，YA、YC 的可选动作只有一个；YB 的可选动作有两个，应用（3-6）式选择动作效用更大的角

色动作来执行。

3.4 FIRA 2D 平台上的实验和结果

目前国际上的 2 个重要的足球机器人赛事 FIRA 和 Robocup 均设有仿真比赛项目。关于 FIRA 2D 足球机器人仿真比赛平台和 RoboCup 2D 足球机器人仿真平台的介绍，很多研究者都做得很详尽，如[59, 60]等。本章决策算法的验证将在集中式控制方式的 FIRA 2D 足球机器人仿真比赛平台上进行，以检验算法应用的典型性和效果。

3.4.1 FIRA 2D 仿真平台简介

FIRA 即国际机器人足球联盟，是韩国学者金钟焕与 1995 年发起的。FIRA 2D，即 SimuroSot 仿真组比赛是 FIRA 比赛的一个重要项目。比赛平台界面如下图所示：

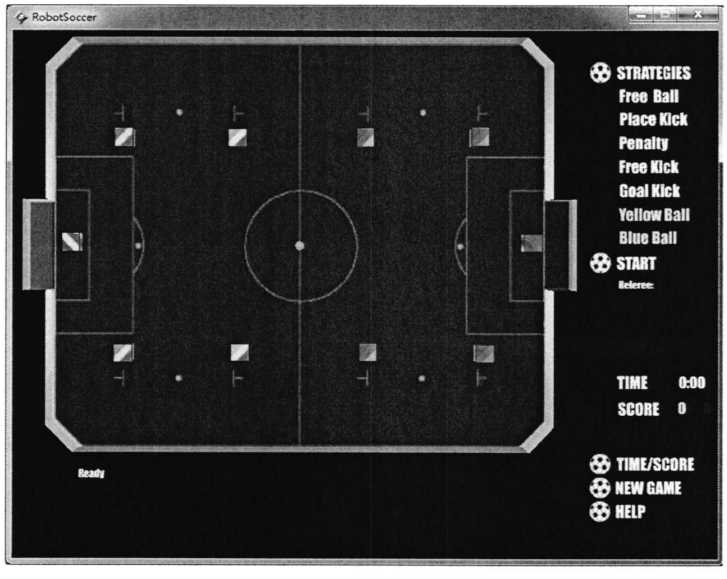


图3-6 SimuroSot 仿真比赛平台

该平台提供了一个精确地虚拟场地模型，其中机器人采用双轮小车模型，采用商业游戏引擎公司 Havok 的碰撞处理引擎，精确计算场上对象的动力学行为。开发者将决策程序编写为*.dll（动态链接库）文件，与平台交互信息并做决策。

该平台采用集中式的控制方式，决策系统取得场上所有对象的精确位置信息，统一决策，最后计算出我方球员的左右轮速，发送给仿真平台，有仿真平台统一执行。仿真平台避免了视觉，机械等系统误差对决策系统的影响，从而更加直接地测试决策系统的性能。

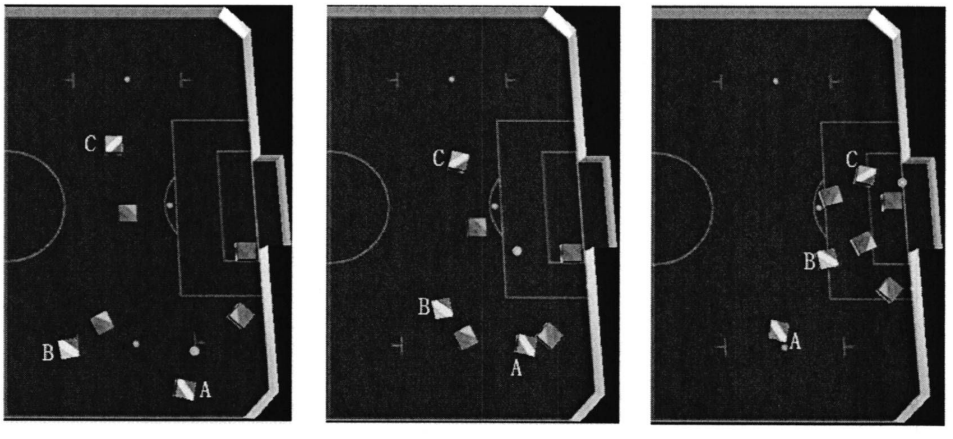
由于本文使用该比赛平台初步做一个基于效用的决策规划研究，关于平台的特点不再多做描述。

3.4.2 实验及结果

在机器人足球仿真比赛中，有几项比较重要的指标需要注意分析，它们不仅说明了队伍实力的高低，更体现了一种算法方法的效果。比分和控球率是最重要的两项指标，这两项体现了一个球队整体上的决策水平；传球、射门等动作的成功率说明了一个队伍动作设计的细腻程度；而传球的次数以及由传球带来的比赛局势的改变最能体现一支队伍中球员角色协调合作的效果。

按照 3.2 节中提出的策略结构设计策略程序为三个层次：全局层、角色层和动作层。其中角色层用上一小节提出的方法进行设计。设计一些特定进攻场景，调试策略程序，并调整效用函数各部分的权重，使算法最终达到稳定有效。

实验的第一部分是在给出特定攻防场景下，观察策略算法执行的效果。如图 3-7 a 所示比赛场景，球落在场地右下角，我方队员 A 控球，这种场景在比赛中经常见到。由于 A 在该场景下进攻较困难，要求队友 B、C 能够进行接应，A 将球传给位置更好的队友，射门得分。经过一定的参数调整，A 能够找到合适的路线准确地将球传给队友 C，如图 3-7 b，由 C 射门得分，如图 3-7 c 所示。



a　　b　　c

图3-7　设定场景下的策略执行

第二部分实验用本文设计的策略程序与平台自带的策略程序进行比赛，并主要针对前

文所述的几项指标统计记录比赛结果，图 3-8 为比赛截图。

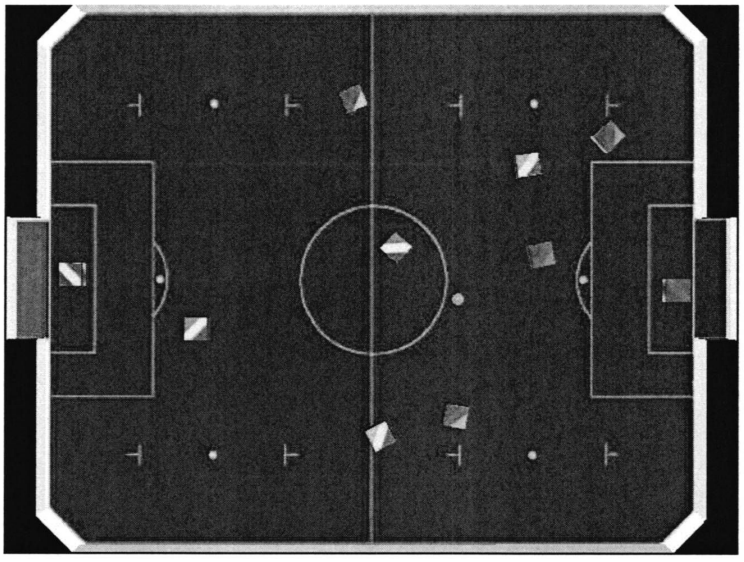


图3-8　Robot Soccer V1.5 比赛截图

仿真比赛试验一共进行 10 场比赛，每场比赛上下半场各 5 分钟，全场 10 分钟。比赛的统计结果如表 3-1 所示。

表3-1　本文方法 vs.平台策略的结果统计

策略	总控球时间比(%)	传球成功数/传球次数	得分/射门次数
平台自带	28.3	7/36	17/37
本文方法	71.7	61/79	51/63

从比赛数据统计表 3-1 可以看出，用本文方法实现的决策系统有较大的优势，从传球的次数以及成功率可以看出该方法能够选择合适的角色动作，实现良好的团队配合；从得分数以及控球时间来看，该策略算法对比赛形势有较强的控制能力，能很好的完成集体进攻目标。

辐射区域的使用对效用的预测及行为动作的成功率都有较大作用，所以实验的第三部分用本章的策略程序与不用辐射区域的策略程序进行比赛。比赛试验同样进行 10 场比赛，统计结果如表 3-2 所示。

表3-2　本文方法 vs.不用辐射区域策略的结果统计

策略	总控球时间比(%)	传球成功数/传球次数	得分/射门次数
不用辐射区域	43.2	37/71	27/57
本文方法	56.8	47/64	39/51

从统计表 3-2 可以看出，使用辐射区域概念设计的比赛策略有较高的稳定性。从传球和得分情况可以看出使用辐射区域概念的策略能更理性的根据比赛态势做出决策，虽然传球数和射门次数较少，但是有较高的成功率。从控球时间比来看，使用辐射区域能有效减少丢球，从而更好地主导比赛形势。

3.5　本章小结

多智能体之间行为动作选择的协调性是实现团队合作的基础，也是多智能体系统性能的重要方面。以机器人足球比赛为研究对象，基于 MDP 任务层次分解决策框架，对基于角色的策略系统进行分析，提出了一种新的足球机器人角色实现方法。在 FIRA 标准仿真比赛平台的运行结果表明，该方法能较好地实现智能体之间的协调配合，有效地完成系统总体目标。

由于实验平台采用的是集中式控制方式，而实际系统往往要求分布式控制。为了使该方法更易于在实际系统中应用，下一章将在智能体完全分布式控制，每个智能体只能得到局部信息，相互之间只能进行有限通信的环境下，研究其协调合作问题。并且优化效用函数各部分的比重关系，设计更加合理有效的协调机制。

第 4 章　基于 MAXQ 分解的决策规划

上一章介绍的是一种相对简单的集中式控制方式下的决策规划方法。而在现实世界中，更普遍的情况是分布式控制方式，各智能体可能有不完全的感知，智能体之间有一定的通信，而且状态空间和动作空间的规模一般较大。本章讨论在完全分布式控制下,智能体间可能存在部分通信，拥有局部感知能力，并且规模较大的多智能体的决策问题。

本章首先介绍基于 MAXQ 值函数分解的任务层次分解模型，以及在这种分层模型下，策略优劣的评判标准，即值函数的表示方法。基于任务分层模型，设计了一种有效利用有限的感知和通信资源的多智能体决策的框架结构。鉴于对分层模型策略的常用的基于学习的求解方法面对连续状态空间问题的局限性，采用基于与或图表示策略路径的方法，设计了一种在线实时求解分层策略的方法。该方法假设原始决策问题可以基于状态抽象和动作的执行条件生成有限的可行策略，但并不要求很精确地表示策略，在连续的状态空间条件下依然适用。方法成功在 RoboCup 2D 球员决策问题中实现，该实验的设计和结果将在下一章讨论。

4.1　任务层次分解方法

2.1 节已经介绍了基本 MDP 的一些概念和半 Markov 决策过程（Semi-MDP）的基本概念。本节首先介绍基于 MAXQ 值函数分解的任务层次分解方法，以及值函数的分层表示，然后讨论该分层方法解策略的收敛性问题和状态抽象方法的使用条件。

4.1.1　基于 MAXQ 的任务层次分解

对于大规模状态空间的决策规划问题，如果能用状态抽象（State Abstraction）的方法消除规划过程中大量不相关的状态变量、状态空间中大量不可达的状态以及将一些状态变量合并成维度较低的状态元组将会非常有用。近年来，基于层次分解的随机规划得到了很多的研究[61, 62]。基于 MAXQ 值函数分解的 MDP 任务分解方法是 Thomas G. Dietterich[63] 在 1998 年提出来的，在总结之前分层方法的同时，提出了一种有效表示分层结构值函数的方法。这里首先讨论分层方法的特点。

将大规模问题分解成子问题有很多优势：1）子任务的好的策略可以供很多更高层的

任务分享；2）子任务的值函数可以分享，这样有任务需要重用这个子任务时，就可以很快计算出值函数；3）如果可以使用状态抽象的方法，总的值函数就可以紧凑地（compactly）表示成状态变量的一些子集的相互独立的情况的和。

子任务策略分享可以使高层决策出需要相似的低层动作时，不需要重新规划低层行为，有效地提高决策规划效率。子任务值函数分享，使得规划出的策略在进行质量评估的时候，不需要重复计算低层的值函数。这在本章设计的策略求解方法中有重要作用。

状态抽象的方法对于连续状态空间问题有很多优点。可以经过去除不相关状态变量减小状态空间大小。可以通过状态抽象，将原本意义模糊的状态表示转换成意思明确，甚至离散的状态。例如，机器人足球比赛中，一种简单的攻防决策方法通过只计算出当前的控球方，来决定采取进攻策略还是防守策略。如果我方控球，则采取进攻规划；相反，对方控球，则规划防守策略。

这里我们讨论的是非折扣基于目标状态的（undiscounted goal-directed）MDPs，也被称为随机最短路径问题。[64]指出，任何 MDP 都可以转换成等价的非折扣基于目标的 MDP，这里非目标状态的报酬严格是负的[65]。所以非折扣的基于目标的 MDP 实际上是一种一般化的决策模型形式。

MAXQ 层次分解技术将一个给定的 MDP 分解成一系列按层结构组织的子 MDPs，表示为$\{M_0, M_1, M_2, ..., M_n\}$。如图 4.1 所示。

该图也称为任务图（Task Graph）。其中，每个相同层中的任务属于相同的任务层级。例如图中的M_1, M_2和M_3。每一个子任务都是一个独立的 MDP 问题。特别地，M_0是跟任务（Root Subtask），也就是说解决了M_0就相当于解决了原始的 MDP M 本身。我们定义其中的子任务M_i为一个三元组[19]：

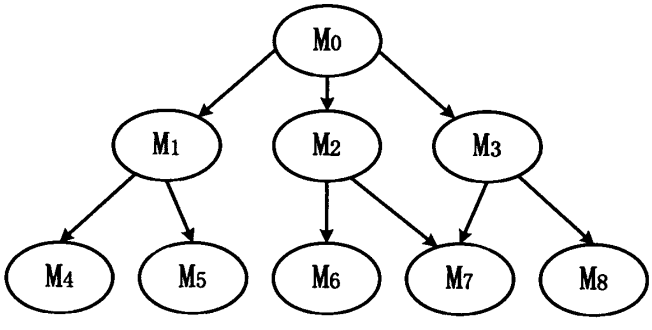


图4-1 一个 MAXQ 任务分解图

定义 1：一个不带参数的子任务M_i定义为一个三元组$\langle T_i, A_i, \tilde{R}_i \rangle$，其中：

- T_i：终止判定条件（Termination Predicate）。将子任务M_i的状态划分成一系列的活动状态S_i和终止状态G_i。子任务M_i对应的策略只有在它的活动状态S_i处执行。任何时候，执行M_i，使MDP到达终止状态，子任务M_i立即终止。
- A_i：可以被子任务M_i执行来完成该子任务的一系列动作。这些动作既可以是原MDP的原始动作（Primitive Action）A，也可以是其他子任务的可行动作。每个子任务都不能直接或者间接的递归调用它本身，而只能调用更底层的子任务。
- $\tilde{R}_i(s')$：伪报酬函数（Pseudo-reward Function）。它确定了每一个转移到终止状态$s' \in G_i$的（确定的）伪报酬。伪报酬代表了这个终止状态相对于一个子任务的满意度。通常，给一个目标状态0的伪报酬，而非目标状态的终止状态给以负的伪报酬[19]。

如果一个子任务带参数，那么不同参数都确定了一个独立的子任务。可以认为参数的值都是子任务名字中的一个部分。如果一个子任务的参数的取值范围很大，这相当于产生了大量的不同子任务。分层MDP的解是一个分层策略，定义如下：

定义 2：一个分层策略π是一个包含问题中每个子任务对应的一个相应策略的集合：$\pi = \{\pi_0, \pi_1, ..., \pi_n\}$。

每个子任务π_i的策略接受一个状态参数，返回一个要执行的原始动作或要调用的绑定特定参数的子任务。子任务s在状态终止的概率$\beta(s)$当$s \in S_i$是0，当$s \in G_i$是1。对于有参数的任务，策略也要参数化。策略π接收一个状态和一个绑定的形参，返回一个选定的动作和它的参数。

分层策略使用一个和一般编程语言类似的堆栈的形式执行。执行一个分层策略的算法可以参考[19]。分层策略优劣的评价标准，与经典MDP的值函数概念对应，通过分层MDP的分层值函数来评价。

定义 3：分层值函数，记为$V^{\pi}(\langle s, K \rangle)$，表示在状态$s$和分层策略执行堆栈内容$K$时执行分层策略$\pi$的期望累积回报[19]。

在Ron Parr [66]的HAMQ算法学习的就是这种分层值函数。但是基于MAXQ值函数分解的分层策略中，我们只集中于求解层次中每个子任务$M_0, M_1, M_2, ..., M_n$相应的投影值函数（Projected Value Function）。

定义 4：分层策略π在子任务M_i上的投影值函数，记为$V^{\pi}(i, s)$，表示从状态s开始，执行子策略π_i以及M_i的所有后代任务的子策略直到M_i终止的期望累积回报[19]。

根任务M_0的投影值函数也就是原始MDP的值函数可以记为$V^{\pi}(0, s)$。基于MAXQ值

函数分解方法的目的就是将$V^{\pi}(0,s)$分解成层结构中所有子任务的投影值函数$V^{\pi}(i,s)$的组合。

4.1.2 投影值函数分解

上一小节定义了分层策略和它的投影值函数的概念，本节介绍如何将投影值函数按层次分解。这样，通过将根任务的投影值函数分解成各子任务的投影值函数的组合，就可以得到原 MDP 的值函数。值函数的分解是基于如下定理展开的：

定理 1：给定一个子任务$M_0,M_1,M_2,...,M_n$的任务图，和一个分层策略π，每个子任务M_i定义了这样一个半马尔可夫决策过程，这个 SMDP 有状态S_i，动作A_i，概率转移函数$P_i^{\pi}(s',N\mid s,a)$和期望报酬函数$\overline{R}(s,a)=V^{\pi}(a,s)$。其中，$V^{\pi}(a,s)$是$M_i$的在状态$s$的子任务$M_a$的投影值函数。如果$a$是原始动作，$V^{\pi}(a,s)$定义为在状态$s$执行$a$的期望立即回报：$\sum_{s'}P(s'\mid s,a)R(s'\mid s,a)$[19]。

值函数公式可以写成 Bellman 公式的形式[19]：

$$V^{\pi}(i,s)=V^{\pi}(\pi_i(s),s)+\sum_{s',N}P_i^{\pi}(s',N\mid s,\pi_i(s))\gamma^N V^{\pi}(i,s') \tag{4-1}$$

这和上文中 SMDP 的 Bellman 公式有相同的形式，只不过那里右边第一项是期望回报$\overline{R}(s,a)$，而这里是子任务a的投影值函数。

为了得到投影值函数的层次分解，下面我们使用行动值函数（Q）表示法。定义上述 Bellman 公式的 Q 符号表示形式：

$$Q^{\pi}(i,s,a)=V^{\pi}(a,s)+\sum_{s',N}P_i^{\pi}(s',N\mid s,a)\gamma^N Q^{\pi}(i,s',\pi(s')) \tag{4-2}$$

公式最右端的部分是在状态s首先执行a，之后直到完成（completing）任务M_i的期望折扣回报。这一部分其实只取决于和i,s和a，因为求和消除了对s'和N的依赖。以下将公式的这一部分定义为记号$C^{\pi}(i,s,a)$：

定义 5：完成函数（completion function），$C^{\pi}(i,s,a)$，表示子任务M_i在状态s首先调用子任务M_a，之后直到完成任务M_i的期望折扣累积回报。该回报折扣到子任务a开始执行的时间点。

$$C^{\pi}(i,s,a)=\sum_{s',N}P_i^{\pi}(s',N\mid s,a)\gamma^N Q^{\pi}(i,s',\pi(s')) \tag{4-3}$$

根据这个定义，可以递归地表示 Q 函数：

$$Q^{\pi}(i,s,a)=V^{\pi}(a,s)+C^{\pi}(i,s,a) \tag{4-4}$$

最后，可以将投影值函数$V^{\pi}(i,s)$重新表示成：

$$V^{\pi}(i,s)=\begin{cases}Q^{\pi}(i,s,\pi_i(s)) & i\text{是组合任务}\\ \sum_{s'}P(s'\mid s,i)R(s'\mid s,i) & i\text{是原始任务}\end{cases} \qquad (4\text{-}5)$$

公式(4-3)(4-4)和(4-5)为MAXQ分层在一个固定策略下的分解公式(decomposition equations)[19]。分解公式递归的将根任务的投影值函数$V^{\pi}(0,s)$递归地分解成各子任务$M_0,M_1,M_2,...,M_n$的投影值函数和各自的完成函数$C^{\pi}(i,s,a)$。这样，要表示值函数的分解，或者说要得到一个策略的分层值函数，需要记录的基本量只是每个非原始子任务的完成函数C值和所有原始动作的值函数V值。

一般地，MAXQ值函数分解的递归展开形式是这样的：

$$V^{\pi}(0,s)=V^{\pi}(a_m,s)+C^{\pi}(a_{m-1},s,a_m)+\cdots+C^{\pi}(a_1,s,a_2)+C^{\pi}(0,s,a_1) \qquad (4\text{-}6)$$

其中$a_0,a_1,\cdots,a_m$是从根任务节点到原始动作节点的由分层策略选择的子任务和原始动作的一条路径。可以从下图较直观的看到这个层次关系：

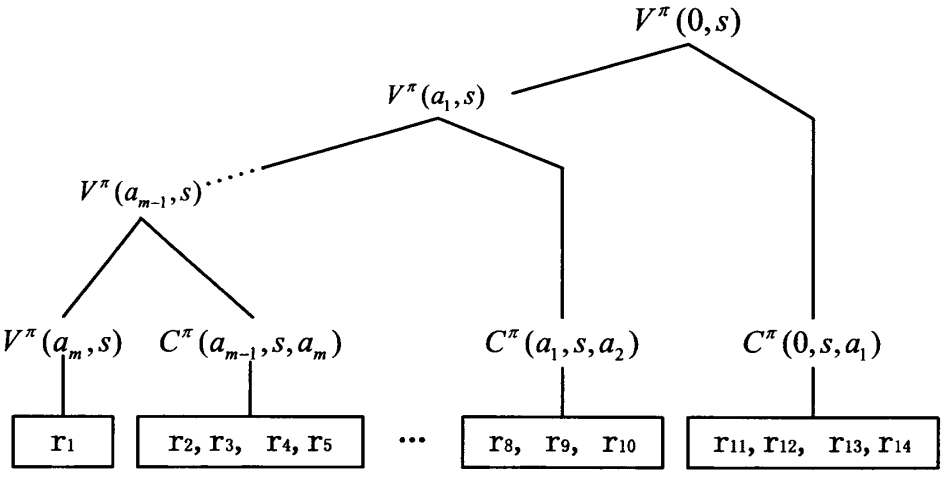


图4-2 MAXQ值函数分解图.

$r_1,r_2,...,r_{14}$表示在时间序列1,2,...,14执行的各原始动作得到的回报

（注：因为完成函数也是个行动执行直到子任务完成的期望报酬）

定理2：用$\pi=\{\pi_i;i=0,...,n\}$表示一个给定MAXQ任务分解$M_0,...,M_n$的一个分层策略，其中是$i=0$根任务。则存在非原始任务的$C^{\pi}(i,s,a)$值和原始任务的$V^{\pi}(i,s)$值，使得由式（4-6）得到的$V^{\pi}(0,s)$是在状态执行策略的期望折扣累积回报[19]。

4.1.3 MAXQ任务分解方法讨论

1. 解的收敛性讨论

一般基于层次分解的大规模MDP决策规划算法实际上是以获得问题的次最优解为代价来降低问题求解的复杂度[67]。因为任务的层次分解，限制了对一些可能策略的考虑。如果这种限制选择的不合适，最终得到的策略就有可能只是次最优的。

在MAXQ分层方法中，有两种形式的限制条件，使得解得的策略可能是次优的。1）在一个子任务中，只有一部分可能的原始动作是可选的。这样限制了一些可选动作的同时也就相应地限制了一些可能策略。2）考虑一个有孩子节点$\{M_{j1},...,M_{jk}\}$的子任务M_j，M_j的策略必须包括孩子节点的策略的执行。当执行一个孩子任务M_{ji}的策略时，必须一直运行到到达它的一个终止状态G_{ji}。这就限制了M_j的策略必须经过一些终止状态$\{G_{j1},...,G_{jk}\}$的子集。

如果得不到最优策略，那么对于我们次优的选择就是在给定分层结构的情况下，得到相对于这个结构最优的策略。一个MDP的分层最优策略（hierarchically optimal policy）就是在所有符合给定层次结构的策略中，使累积报酬达到最大的策略。

2. 状态抽象适用条件

使用状态抽象是引入分层策略规划方法的主要原因之一。如果仅用分层方法，而不进行状态抽象，有可能得到的分层状态空间比平铺式方式得到的状态空间更大；相反，使用状态抽象方法，一般可以极大地降低状态空间规模。

有三类可以引入状态抽象的情况。第一种情况通过消除任务图中一个子任务的无关变量。这种形式下，靠近叶节点的节点一般有很少的相关变量，高层节点有较多相关变量。因此，这类抽象对于任务图中低层节点更有用。

第二类是从“漏斗（funnel）”动作得来的。是一些动作将环境从大量的初始状态转移到少量的结果状态。这样的子任务的完成函数可以用与结果状态的数目成比例的数量的值来表示。漏斗动作经常在任务图的高层出现，所以这类抽象在靠近顶层时更有用。

第三类抽象来自于任务图本身。即由于很多子任务都有有限的终止状态，状态空间中的大部分状态事实上不可能到达[19]。

4.2 受限感知和通信的多智能体决策系统框架

4.2.1 基于贝叶斯估计的状态更新

受限感知和通信的多智能体决策规划问题本质上是一个多智能体部分可观察的模型，但是由于问题规模巨大，几乎无法用处理一般POMDP的方法进行求解。通过一种基于贝叶斯估计的方法来进行世界状态更新，消除信息的不确定性，将问题转化为MDP模型进

行求解。当忽略多智能体的因素时,整个过程实际上就是将一个 POMDP 通过最大似然法转化为一个 MDP 来进行求解[68]。

通过贝叶斯估计的状态更新实际上是用最大似然法得到当前状态的概率最大的一个分布。该方法有两步，首先，通过贝叶斯估计更新当前信念状态的概率分布，然后通过最大似然法得到最可能的当前状态作为当前决策周期的状态。

贝叶斯估计理论是当今科学推理的一个重要工具,与经典的统计估计理论相比有较大优势。贝叶斯估计利用系统的模型信息预测状态的先验概率，然后利用观察值进行修正，得到后验概率。它利用了所有的已知信息来构造系统状态变量的后验概率密度，不仅包括先验知识信息,还有观察信息,得到的估计误差相对较小。贝叶斯估计适合于处理非线性、非高斯系统的状态估计，因为它将未知参数看做随机变量，得到的估计值是该变量的一个特定的实现。

假设观测向量是 $y=[y_1,y_2,...,y_n]^T$，概率密度函数为 $p(y/x)$。其中，$x=[x_1,x_2,...,x_n]^T$ 是未知参数向量。假设在得到观察数据 y 之前，人们对先验概率 $p(x)$，即未知参数 x 已经有一定的认识。由公式：

$$p(y/x)p(x)=p(y,x)=p(y)p(x/y) \tag{4-7}$$

当得到观测向量以后，可以得到条件密度为：

$$\begin{aligned} p(x/y)&=\frac{p(y/x)p(x)}{p(y)} \\ &=\frac{p(y/x)p(x)}{\sum_x p(y/x)p(x)} \end{aligned} \tag{4-8}$$

注意到分母是与 x 无关的，也就是说后验概率分布与先验概率和观察向量的概率密度函数成比例。

先验分布 $p(x)$ 是我们对未知参数 x 的认识， $p(x/y)$ 是在得到观察样本 y 的分布后，对 x 的重新认识，即后验概率分布。这样，未知参数 x 的后验分布就综合了的先验信息和观察样本的信息。将 $p(y/x)$ 视为 x 的函数，称为似然函数。则该公式表明在得到观察样本时，未知参数 x 的后验分布与先验分布和似然函数的乘积成比例。

得到未知参数的后验概率，即信念状态的概率分布，可以直接使用最大似然法确定一个最可能的状态，作为当前环境的状态进行决策规划。该方法可以有效地将部分可观察特性去除，将 POMDP 转化为标准 MDP 模型进行策略求解。

4.2.2 受限通信的多智能体决策框架

在很多现实问题中，多个智能体间可能有一定的受限制的通信。利用这些资源对多智能体协调协作将起到很大的促进作用，并能有效地降低误协调。我们设计一种利用有限通信资源的多智能体决策框架结构。

通信可以在决策过程中共享各自的私有信息，从而可以提高决策效率。但在实际问题中，通信往往是不可靠的或者是代价昂贵的。但并不是说决策规划的时候就要放弃掉通讯的使用[69]。这里通过对受限通信条件的分析，设计了一种针对不可靠通信条件下的决策机制。在没有通信的条件下，智能体根据自己感知的环境信息进行决策；当得到友方智能体分享的信息时，优先依据友方的消息进行决策。由于大多数情况下是没有友方智能体消息分享的，所以要求智能体在根据独立信息决策的时候必须能够得到相互合作的策略（当然，这个时候得到的是一种隐式协作）。而在有消息分享时，可以使用这种消息保证智能体决策的协调一致性。从这个意义上说，通信对保证多智能体协作的决策规划不是根本要素，而是一种提高团队协作效果的辅助方式。

例如，在 RoboCup 2D 中，由于通信是不可靠的，而且在一定时间内对通信的次数有限制。所以，对通信时机的选择和通信内容的设计对该决策规划算法是至关重要的。通信时机方面，在一般情况下，根据 Server 对通信次数的限制，尽可能多的利用通信分享信息；而在一些至关重要的情况下，例如在对方球门前我方持球，要尽可能的利用通信交换决策信息，使我方球员的策略保持协调。而在通信内容方面，主要有以下几点考虑。

首先，所谓隐式协作，指的是智能体之间没有显式的协作规划机制，但是通过相互之间保持一种承诺，遵循一些相同的规则，在各自独立作出决策的时候，在总体上体现出一种协调合作的特点。显然，这种协作机制有一个潜在的问题，就是误协调[69]。例如，在机器人足球比赛中，球员 A 带球并决定将球传给球员 B，但球员 B 独立做出的决定认为自己应该移动到一个更好的位置，而执行移动动作，这样球员 A 传出的球很可能不能被 B 接到，这就是一种误协调。误协调有可能带来致命的结果，因此必须想办法消除。在本文研究的 RoboCup 2D 环境下，在重要决策时，使用通信条件来尽可能的提高协作的效果。

重要决策，一般指对决策问题的结果至关重要的策略，这些策略在执行时如果发生误协调，往往会产生严重后果。一般策略是指那些对决策问题结果作用不那么明显或直接的策略，例如在距球较远的位置的球员的跑位或挡拆策略，即使有误协调，一般也不会影响大局。对于重要决策，每次策略的改变必须第一时间通知队友，以使队友能够第一时间做出对策。

其次，如上节所介绍的。智能体的感知信息不精确，有时也不完整，所以通信的另一个重要任务是分享各自对世界状态的认识，使得各球员维持一个相同的精确的世界模型，这也将间接地提高决策的协调性。对于世界模型更新的方法将在下一节介绍，届时将会用到通信的这部分信息。

以上介绍了对通信内容和时间的要求，接下来分析一下“空间”要求。分别在球场左右边线附近的球员之间维持的世界状态可能不同，相互之间的决策信息也可能不会互相了解。鉴于这个问题，根据智能体分布的空间位置，设置距离较远的智能体之间，能够连接它们的智能体为信息分享节点。在比赛中，适当的条件下转发距离较远的智能体的状态信息和策略信息，使各智能体尽可能保持对环境的全面的、准确的认识。

通过以上分析，本文设计了一种利用受限通信条件下的智能体组织框架结构。多个智能体的决策规划过程如下图所示：

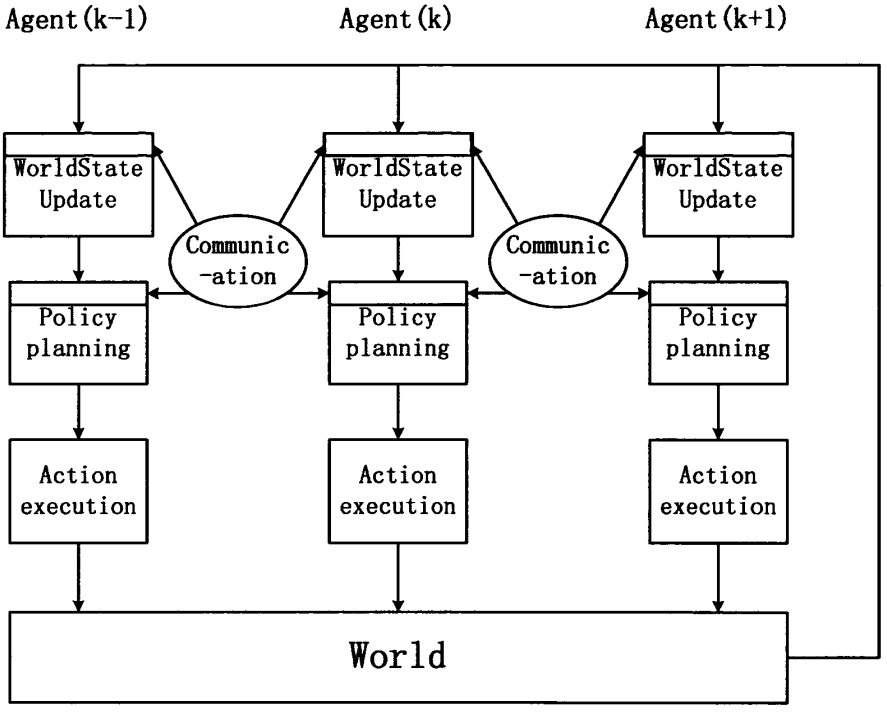


图4-3 利用受限通信资源的多智能体组织结构

从上图可以看出，通信作为智能体决策规划的一部分将不同智能体关联在一起，显式地协调智能体间的决策。而单个智能体的结构和决策规划方法用下文将要介绍的MAXQ-RTP方法进行设计。

4.3 实时分层策略求解

4.3.1 策略求解方法概述

近几年，基于 MAXQ 分解的策略求解方法很多[70]。[19]使用称为 MAXQ-Q 的强化学习算法学习分层策略。通过学习分层结构中各层的局部最优值函数，最终学习到原 MDP 的局部最优值函数。可见，MAXQ-Q 算法最终得到的并不是最优策略，而是收敛于一种比分层最优策略更弱一些的递归最优策略。这也是该方法的代价。另外，学习算法一般学到的是一个状态到行动的映射，而且有一个普遍的问题是很难给出足够的例子让智能体学习[71]。在连续状态空间的问题中很难适用。

[30]针对 RoboCup 2D 决策问题，基于 MAXQ 值函数分解方法对球员智能体决策系统建模，并设计了一种策略在线规划方法。通过边规划策略路径，边评估值函数，求解最优策略。该规划算法有两点值得改进的地方。第一，策略路径的值函数是递归表示的，对其计算有一定代价，对于一些明显不好的策略路径，例如路径中有一个子策略的成功率非常低时，根本没有必要对该策略路径进行评估，因此一边规划一边计算值函数有可能浪费一些计算时间。在实时策略规划问题中，应尽量避免规划时间的浪费。第二，对于很多实际问题，较低层的子任务很容易离线求解出对应的最优子策略。但是这些子任务可能经常被重复的使用，其值函数也经常需要重复的计算。如果离线的求解出这些低层子任务的最优策略和对应的值函数，在线规划时直接使用将可以有效地提高策略规划效率。

实时策略规划一般通过一些启发式方法，利用当前状态的可达性信息，即只考虑从当前状态可以到达的状态，采用前向搜索方法评估实时的更新可达状态和评估策略路径，选出其中最好的一个策略。在线算法由于不需要非常精确的策略表示，可以处理连续的状态空间和动作空间问题。另外，在线算法可以容易地处理不可预测的环境变化，是一个处理现实世界应用的良好选择。

本章提出一种用 MDP 建模的随机领域问题的实时策略规划算法，为叙述方便，将该算法称为 MAXQ-RTP（MAXQ Real-time Planning）。采用与或图表示可行策略路径，结合层次分解方法和实时在线规划方法的优点，解决大规模 MDP 问题。与现有大部分 MDP 算法不同，该算法在线运行，只寻找当前决策步骤下的最优策略。算法只搜索从当前状态开始的状态空间中的一小部分，对大规模问题，比离线算法有效的多。本算法的主要贡献是对 MDP 的分层结构模型进行实时在线规划，并能有效地处理连续状态空间但可以进行有效状态抽象的问题。

4.3.2 基于与或图的策略表示

与或图是实时规划算法常用的策略表示方法，如 2.2 节所介绍。与或图中圆圈表示状态（或节点），方框表示这个状态下的可选行动（与节点）。在与或图中，每条从根节点到叶子节点的路径即代表从当前状态到一个目标状态的策略路径，通过评估各条路径的分层值函数来选择最优分层策略。

首先我们介绍用与或图表示的策略路径。如下图示例。

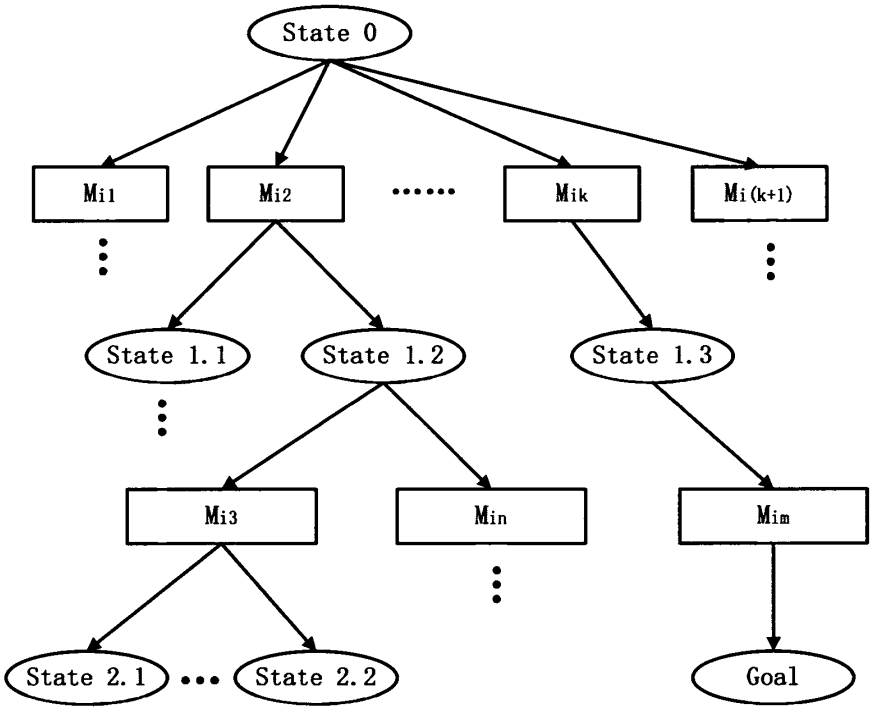


图4-4 与或图示例

图中每一条从根节点到叶节点的路径都是一个可行的策略执行过程。图中根节点对应的是当前的实际世界状态，其他与节点对应的状态均为预测状态。对于一般的决策问题，图中每个行动节点这里表示的是一个子任务。每层子任务都代表一个决策深度 N，即模拟执行子任务的步数。每个子任务执行后，对应一组可能的后继状态。在某一状态，可行的行动是单独执行的，但是执行后的结果一般不止一种可能，而是对应一组可能后继状态的概率分布。但在分层结构中，每个子任务的动作节点本身又是包含其孩子任务直到原始动作的与或图。如下图示例:

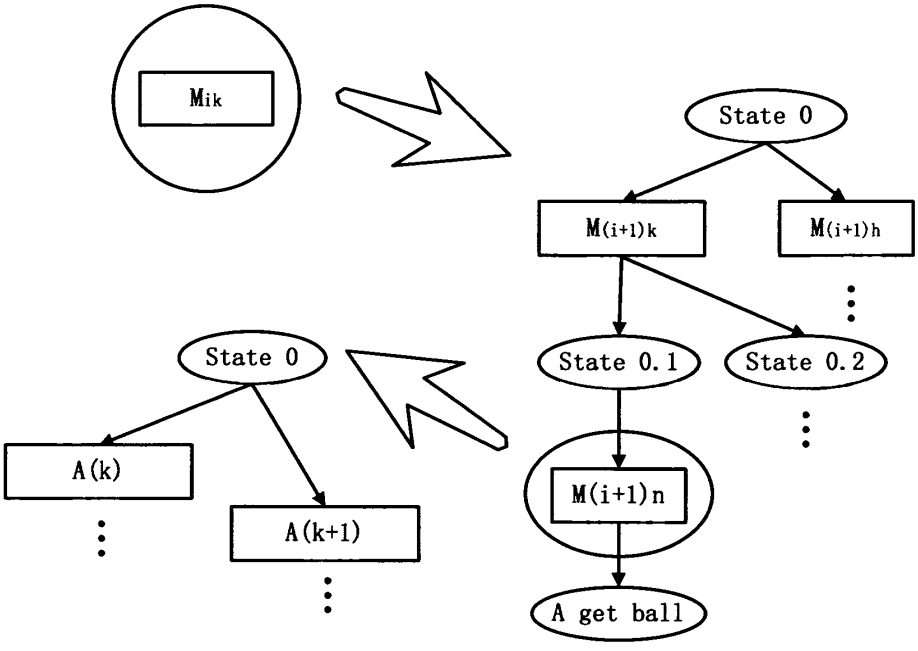


图4-5 子任务递归分解示意图

每一个连接第 i 层状态到第（$i+1$）层状态的行动节点都是它的子任务序列，以及子任务的子任务序列，直到最终的原始动作 A 的与或图。因此，经过策略评估器选择的策略实际上也包含了每个动作周期要执行的动作。根据 MAXQ 分解方法的解策略特点，求解到的策略也包含了所要执行的第一步可执行的基本动作信息。

4.3.3 分层策略求解算法

对于像机器人足球比赛这样的连续状态空间问题，基于与或图表示策略路径，设计 MAXQ-RTP 算法主要包括两个核心模块，即策略生成器（Policy Generator）和策略评估器（Policy Evaluator）。

在每个决策周期，通过策略生成器生成所有从当前状态到某个目标状态的与或图，也就表示了当前状态下所有可行的策略集合。由于生成的与或图可能会很大，所以首先对与或图进行裁剪。叶节点对应的是不同的终止状态，对于那些明显不理想的终止状态对应的路径，也直接删除。对于动作安全性较低的中间动作的节点及其后续节点也全部删除。对于剩余的与或图，使用策略评估模块根据每条策略路径的分层值函数进行评价，最终选择最优的分层策略。在评估每个可行策略值函数的时候，也需要递归地评估其子任务的值。根据分层值函数的大小选择最优的分层策略。显而易见，该规划算法的关键步骤是策略生

成器和策略评估器。策略生成器一般根据问题域的领域知识使用一定的启发式方法设计。

实时动态规划（RTDP）等方法可以根据标记已经计算过的状态是否已得到足够满意的策略[24],或通过一个判定准则来决定是否对当前状态不再计算[11]。但是在像机器人足球比赛这样大规模的连续状态空间问题中，要想记录一个状态是否已计算过而在下一次计算时利用这一信息几乎是不可能的。即使是通过一定的离散化手段，将相似的攻守状态进行记录也将花费巨大的存储空间。

基于以上分析，本文使用另一种方式利用计算过的决策信息。在一次决策完成后，执行选定的策略。由于一个执行步骤只是执行很小的一个原子动作，在很多情况下，相邻几个决策周期中得到的最优策略是差不多相同的，那么在下一次决策开始，通过首先判断是否可以继续执行上一步得到的策略，来决定是否利用计算好的决策信息。如果可以继续执行上一步的策略，则继续执行；否则，在跟据这个周期得到的状态信息，重新规划。

这里引出了两个问题，第一，如何判断上一个周期的策略是否可以继续执行，第二，如果继续执行上周期的策略，那么本周期的决策时间是不是要通过等待白白浪费掉。

对于第一个问题，我们首先分析与或图的结构发现，一个动作节点的动作的完成是要通过多步的原子动作的执行实现的。也就是说，在当前的状态下，执行上一步的策略，应该以近似的概率到达这个节点动作的后继状态。基于这一点，我们通过计算在当前步执行上一周期策略到达第一个后继状态的概率，与上周期计算得到的状态转移概率是否相近，来决定是否可以继续执行上一步得到的策略。

对于第二个问题，答案是否定的。由于我们的决策模型对原始问题做了较大的近似，所以有必要利用尽可能多的资源来弥补这种近似带来的策略优异度损失。我们设计一个待执行的动作的列表，在决策时，将决策得到的较好的策略放入列表中，在接近执行阶段时，再次对列表中的行动进行排序，最终执行最优的策略。在判断出可以继续执行上一步得到的策略后，将其放入列表中，然后仍然根据当前状态进行这一步的决策规划，如果找到一个更好地策略，就放弃上一步的策略，转而执行这个更好的策略。而在一个决策周期，有时可能没有来得及评估整个与或图中的策略就已经耗完了决策时间。对于这个问题，在决策时得到好的策略，即放入策略队列中，如果时间将近耗尽还没有找到最优策略时，则执行当前已经得到的策略中最好的一个。这也是一种 Anytime 行为机制，即策略的质量随着决策时间的增加而逐渐提高。

基于以上分析，将算法进行改进，加入策略是否继续的判定环节。改进后的算法流程如下表所示：

表4-1 MAXQ-RTP 算法

	Alg. 4-1：MAXQ-RTP
1	**Start**
2	**Repeat**
3	*PolicyList*：initialization；
4	*WorldModel*: update world model;
5	**If** *CommPolicy*, push to *PolicyList*;
6	**If** *LastPolicy* is continuable, push to *PolicyList*;
7	*PolicyGenerator*: generate all possible policies;
8	*PolicyPruner*: prune the policy AND-OR graph;
9	*PolicyEvaluator* : evaluate the hierarchical value functiong of each possible policy path;
10	**While** isGoodPolicy(), **do** push to *PolicyList*;
11	*Executor*: select the best policy and execute
12	**Return （Optimal Policy）**;
13	**End（Repeat）**
14	**End（MAXQ-RTP）**

由上面的算法可见，在线策略规划过程中，关键的步骤是策略路径的生成和评估。对于策略与或图的生成，一般对不同的问题，应用其领域知识，加上一定的启发式方法搜索得到。对于本文研究的实验对象 RoboCup 2D 环境，策略与或图的生成将在下一章具体介绍。策略评估器即分层值函数的计算，由 4.1 节的值函数分解可见，每个可行策略的值函数是这条策略路径上原始动作的值和一系列非原始任务的完成函数的和。原始动作的值一般是定义好的，所以关键是在完成函数的计算上面。

4.3.4 完成函数计算

要精确的计算出子任务 i 的完成函数 $C^{\pi^*}(i,s,a)$，根据定义，需要明确的知道最优策略 π^*，而这就相当于解决了整个决策问题。但是，一般情况下由于时间限制，在线找出最优策略很难做到。将 MAXQ 分解方法应用到大规模在线决策规划问题中时，一般需要使用近似方法来计算每个子任务的完成函数。

这里，我们首先考虑问题域的特点。对于大多数 MAXQ 分解中较低层的子任务，完成这些子任务一般较容易找到离线的最优策略。例如，RoboCup 2D 中，带球从当前位置到一个目标位置这个子任务可以通过多种离线方法得到最优的策略。在高层子任务规划时，就可以利用这点信息，而不需要重复规划带球行为。这也正是 MAXQ 分解方法的优点之一，即子任务分享。而离线计算到的较好策略的值函数也可以在在线计算策略值函数时供更高层的子任务分享使用。因此，本文将完成函数的计算分为两种：一种是较低层的子任

务的值的离线计算，一种是高层子任务值的在线近似计算。得到这些值即可计算出一个策略的值函数。

底层值函数计算，一般使用比较成熟的方法，如值迭代方法，强化学习方法等。例如，在 RoboCup 2D 决策问题中，[29]用值迭代求解最优多步踢球子任务，强化学习求解射门策略等。高层值函数在线近似计算，对于不同的实际问题，可能采用不同的近似计算方法。例如， [20]采用基于重要性采样的近似计算方法。

高层值函数的近似求解，我们采用[30]的基于重要性采样的近似计算方法。首先我们假设无折扣模型。给定一个最优策略，子任务将会在一个特定的目标状态以概率 1 终止。要计算完成函数，唯一需要考虑的是 $P(s',N\mid s,a)$，一个终止状态上的概率分布。给定一个子任务，一般总是可以直接近似找出这个分布，而不考虑执行的细节。基于这些考虑，我们假设每个子任务 M_i 将以一个优先的分布 D_i 终止于终止状态 G_i。理论上，D_i 可以是与每个子任务相关的任何概率分布。$\tilde{G}_a$ 表示用重要性抽样（Importance Sampling）[72]从优先分布 D_a 得到的采样状态集合，完成函数 $C^*(i,s,a)$ 可以由下式近似得到：

$$C^*(i,s,a)\approx\frac{1}{|\tilde{G}_a|}\sum_{s'\in\tilde{G}_a}V^*(i,s')\text{。}$$

算法过程如表4-2所示。

算法给出了一个递归估计完成函数的步骤。在实际中，作为计算完成函数的优先分布 D_a，可以考虑领域知识改进。以机器人足球为例，个体的状态 s 可能是在场地上的某个位置，假设 s' 是得分的状态。那么，智能体可能有较高的概率带球到球门前射门，或者传球给更近的队友来达到 s'。

4.4 本章小结

本章首先介绍了基于 MAXQ 值函数分解的大规模 MDP 任务层次分解方法。通过使用状态抽象方法，可以有效地解决连续状态空间问题。然后讨论了可以使用状态抽象并且保持收敛性的条件。基于任务分层模型，设计了一种有效利用有限的感知和通信资源的多智能体决策的框架结构。鉴于对分层模型策略的常用的基于学习的求解方法在面对连续状态空间问题时的局限性，采用基于与或图表示策略路径的方法，设计了一种在线实时求解分层策略的方法。在实际应用中，如 RoboCup 2D 问题中，使用这些状态抽象方法将极大地降低问题域策略求解的难度。

表4-2 基于重要性抽样的完成函数估计

	Alg. 4-2：完成函数评估（**EvaluationCompletion(*i*,*s*,*a*,*d*)**）
	Input：Subtask M_i，State s，Action M_a，Planning depth d
	Output：Estimated $C^*(i,s,a)$
1	**Start**
2	$\tilde{G}_a \leftarrow$ **ImportanceSampling**(*Ga, Da*)；
	$v \leftarrow 0$；
3	**While** $s' \in \tilde{G}_a$ **do**
4	$d' \leftarrow d$；
5	$d'[i] \leftarrow d'[i]+1$；
	$v \leftarrow v + \frac{1}{\lvert\tilde{G}_a\rvert} EvaluateState(i,s',d')$
6	**End**（**while**）
7	**Return** v；
8	**End**（**EvaluationCompletion**）

下一章将把该算法用于 RoboCup 2D 问题中，通过这个大规模的决策问题检验算法的性能。

第 5 章　分层策略求解方法在 RoboCup 2D 球队决策中的应用

本章将上一章介绍的基于 MAXQ 任务层次分解的策略规划方法用于 RoboCup 2D 决策问题建模中，基于 MAXQ-RTP 设计实时策略求解算法，并通过实验检验算法的效果。首先分析了 RoboCup 2D 平台的特点，然后对球员智能体进行 MDP 建模。应用第四章的任务分层方法进行任务层次分解。采用 MAXQ-RTP 算法计算最优分层策略进行实验。最后给出实验结果和讨论。

5.1　RoboCup 2D 仿真平台

更详细的平台介绍可以参考[7, 73]。这里我们主要介绍一下 RoboCup 2D 环境的主要特点，以及以此平台研究的典型性。

5.1.1　平台简介

RoboCup 2D 仿真比赛是机器人足球世界杯 RoboCup 联赛中最重要的比赛项目之一。由于该平台对硬件的要求较低，不需要很高的投入，开发人员可以将研究的重点放在决策问题中，因此吸引了国内外众多的研究人员。目前，该项目已经是 RoboCup 中参与队伍最多的比赛项目。

RoboCup 2D 平台允许开发者用不同的编程语言开发自主球员程序进行仿真足球比赛。仿真比赛采用服务器/客户端（Server/Client）模式。服务器端提供一个虚拟的足球比赛场地，通过计算动力学特性模拟场上球和球员在内的所有物体的移动；每个客户端程序相当于一个球员的大脑，控制场上该球员的动作选择。服务器和客户端之间通过 UDP/IP 协议进行信息交互，开发者可以使用任何支持 UDP/IP 协议的编程语言来设计球员程序。通过 UDP/IP 协议，客户端可以发送指令控制场上相应球员的动作，而服务器则按照规则给每个客户端发送相应可以获得的比赛信息。下图 5-1 是 RoboCup 2D 平台的结构简图。

每个客户端程序只能控制一名球员，比赛双方必须运行与比赛球员数目相同的客户端程序数量，包括 11 个球员 Player 程序和 1 个教练 Coach 程序。球员（客户端）程序之间

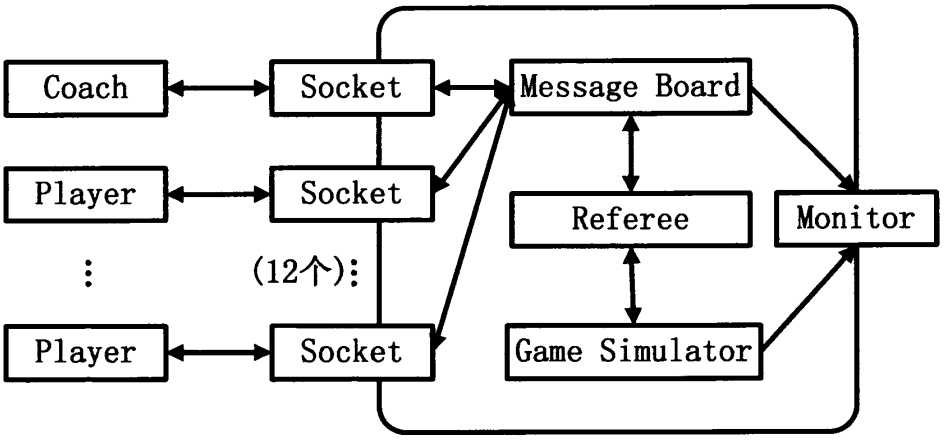


图5-1 RoboCup 2D 结构简图

的通讯必须经过服务器端程序转发，不经服务器端转发而自行进行的客户端之间的通讯都视为犯规，这一点充分体现了分布式控制的特点。一场比赛开始，双方各 12 个独立的客户端程序连接到服务器端进行比赛，每队的目标就是尽量多的将球踢入对方球门同时阻止球进入自己的球门。

服务器端程序主要由球场仿真模块、裁判模块和消息板模块三部分组成，如上图所示。

球场仿真模块计算场上对象的运动情况，检测它们之间是否发生碰撞等。球场上包括 22 名球员、球、球门、标杆和标记线等对象，教练不在场地中出现。球和球员具有大小、位置、速度和加速度等属性，球员另外还有身体朝向、头朝向和体力等属性。每个决策周期末，该模块都会根据动力学定律更新所有场上对象的属性，如果有重叠发生，则按照比赛规则进行碰撞处理。

裁判模块用来保证比赛公平地进行，根据与人类足球比赛相似的规则来控制比赛进程。仿真比赛环境具有动态、实时和不确定的特点，不可能按照事先设计按部就班的进行，因此需要一个有智能的裁判，处理比赛中的一些简单的情形，如进球、越位、界外球和犯规等。此外，也需要一个人工裁判的参与，来处理一些公平竞争的行为。

消息板模块负责服务器和客户端之间的通讯。服务器采用离散化的方式运行，所有程序的运行都以仿真周期作为单位，1 个仿真周期的时间是 100 毫秒。在每个仿真周期开始时，服务器向球员发送特定格式的信息，包括身体感知信息、听觉信息和视觉信息，由于每个球员在场上的状态不同，它们收到的比赛信息也不一样，这也体现了仿真比赛平台部分可观察的特点。在每个仿真周期结束前，球员发送动作指令给服务器，服务器统一执行所有球员发来的动作指令，更新场上状态，之后进入下一个周期。每个周期只能发送一个

基本动作执行，如果多于一个，服务器会执行发送的第一个指令。而如果球员在某个周期内没有发送动作指令，它将失去该周期的行动机会，这在实时对抗的环境中是很不利的。

仿真比赛的情况可以通过 Monitor 程序显示，Monitor 直接与服务器连接，可以实时显示比赛状态，也可以播放比赛录像，供开发者调试使用。

客户端程序通过 UDP 套接字连接到服务器，通过该接口，客户端可以发送指令控制场上球员的动作，也可以接收服务器发来的各种信息。可以将客户端程序当成一个球员的大脑，从服务器获取信息，并发送动作指令到服务器。

仿真平台对球员可获得的信息加入了很多限制和误差，以尽可能地模拟现实环境，球员智能体必须在这些限制下完成决策。每个球员都有一定的视野范围，每个周期只能获得可视范围内一定距离内的对象信息，这些信息也加入了一些随机误差；球员也有一个体力值属性，连续跑动一段距离后，球员可能因为体力过低而无法继续运动，如何合理地分配体力也是决策时需要考虑的；为了增加比赛的不确定性，平台还加入了风等各种噪声干扰，使比赛更趋于真实。

5.1.2 平台特点

RoboCup 2D 提供了一个研究大规模不确定环境下智能体协作和对抗的平台。虽然也涉及知识表示，机器学习等方法的研究，但平台本身的特点决定了其核心的研究问题是智能体的决策理论[29]。RoboCup 2D 平台所涉及的决策问题的特点主要有以下几个方面：

- 问题规模极大（large-scale）

单从状态空间的大小，可以从一个对比看出。“深蓝”计算机处理的国际象棋问题的规模大约是 10^{20}，围棋问题的规模约是 10^{200}。而 RoboCup 仿真平台是连续的状态和行动空间，状态空间即使做很粗略的离散化，如每个状态变量离散化为 1000 个离散值，其状态数量也要超过 10^{400}。

- 多个智能体（multiple agents）

由于队友的存在，涉及到智能体间的合作；而对手对抗的存在，则涉及到必须考虑对手策略的博弈。

- 不确定性（uncertainty）

2D 平台有很多不确定的因素，如行动结果的不确定，环境部分可观察而且存在噪声，队友之间信息不一致以及对手模型不确定等。例如，视觉信息的不确定性，其模型在下节介绍。

- 实时系统（real-time）

要求在 100ms 的仿真周期内，在考虑各种因素的情况下，做出决策，对决策系统的实时性有很高的要求。

5.2 智能体决策框架

本节首先介绍 RoboCup 2D 球员智能体的基本框架，包括信息解析、决策规划系统和动作执行三个模块。

多智能体的决策框架采用 4.2 节介绍的受限感知和通信的多智能体决策框架。感知方面，RoboCup 2D 中存在两个问题需要消除，即位置信息误差和球员身份的不确定性。位置误差的消除相对容易。理论上根据视野内两个标志物的相对距离就可以计算出对象的位置。一般在一个视野内会有多个标志物信息，根据相对于多个标志物信息的距离计算出的位置信息取均值即可基本消除位置误差。而身份不确定性的消除，我们首先对每个球员维护一个位置未更新周期 not_updated_cycle 表示在这么多步中，没有更新该球员的位置。根据这个时间信息，可以结合球员在一场比赛中的平均速度计算出一个活动区域。根据这个信息可以初步估计出当前周期会进入视野的球员列表。在看不到球员号码和队别的时候，通过未更新的球员在一个活动区域中出现的概率，通过贝叶斯估计出一个未知身份球员的概率分布，然后使用最大似然估计，确定未知球员信息。

通信方面，在 RoboCup 2D 中，由于存在不可靠的通信，而且在一定时间内对通信的次数有限制。所以，对通信时机的选择和通信内容的设计对该决策规划算法是至关重要的。这里，本队中多智能体的组织结构采用 4.2 节介绍的受限通信多智能体组织结构进行设计。由通信模块在每个决策周期开始时分享世界状态和重要决策等信息，以充分的利用问题域的通信资源。

鉴于 RoboCup2D 平台的连接特性，即决策周期只有 100ms，每个决策周期开始发送比赛信息，决策周期结束前需要球员智能体发送动作指令，并且 Server 与智能体 Client 采用不可靠的 UDP/IP 协议通信。因此，保证信息收发的及时性和决策的实时性是球员智能体设计是要关键考虑的。

本文借鉴中国科学技术大学 RoboCup 2D 球队 WrightEagle 的多线程智能体框架结构，将智能体的信息解析、决策规划和动作执行三个模块分别放在感知线程、决策线程和行动线程中[9]。这样设计的智能体基本框架如下图所示：

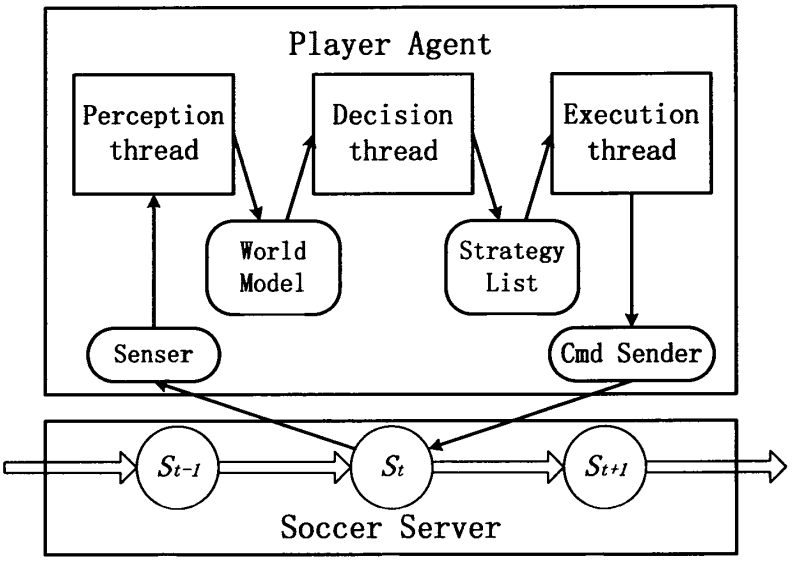


图5-2　球员智能体结构

感知器模块从每个周期开始时，从服务器接收 ASCLL 码表示的比赛信息，包括视觉信息、听觉信息和身体感知信息。通过解释器将这些信息计算状态因子的变量，使用这些变量就可以计算出各状态因子的值。5.1 节已经介绍了客户端感知信息的不完全性以及误差的存在，加上 UDP/IP 通信的不可靠性，要想直接将计算结果用于 MDP 决策求解，状态误差将会较大，因此，需要一些状态更新算法来消除这些误差，下一节介绍一种基于贝叶斯估计的方法更新世界模型。

由于每个决策周期只有 100ms，而根据经验，留给状态更新的时间最多只有 1ms 左右[29]。因此感知器线程在更新世界模型之后，马上将线程设置为等待消息状态。这样，可以确保在下一个比赛信息到达的时候，能第一时间接收到并进行下一轮的解析。感知线程就这样周而复始的执行接收—解析—等待的过程。

决策模块将决策问题建模成 MDP，基于 MAXQ 分解方法将这个 MDP 分解为层次的小的 MDPs。在每个决策周期，通过策略生成器模块产生在该状态下可行的策略集合，存放在与或图表示的结构中。图中的每条从根节点到叶节点的路径都是一条可行的从当前状态到目标状态的一个行动过程。通过比较每条路径的 MAXQ 分层值函数，选择在这种分层结构下最优的策略执行。

原子动作规划模块根据选择的策略的第一步要执行的行动，规划要实现这个行动所需要采取的原子动作及其参数，如移动的加速度值、踢球的力量值等。执行模块将最后的原子动作及参数发送给服务器，然后继续等待下一步决策的结果。类似于感知线程，执行线

程循环的执行原子动作规划—发送—等待的过程。

三个线程的执行流程如下图所示，从图中也可以看出各线程的时序信息。

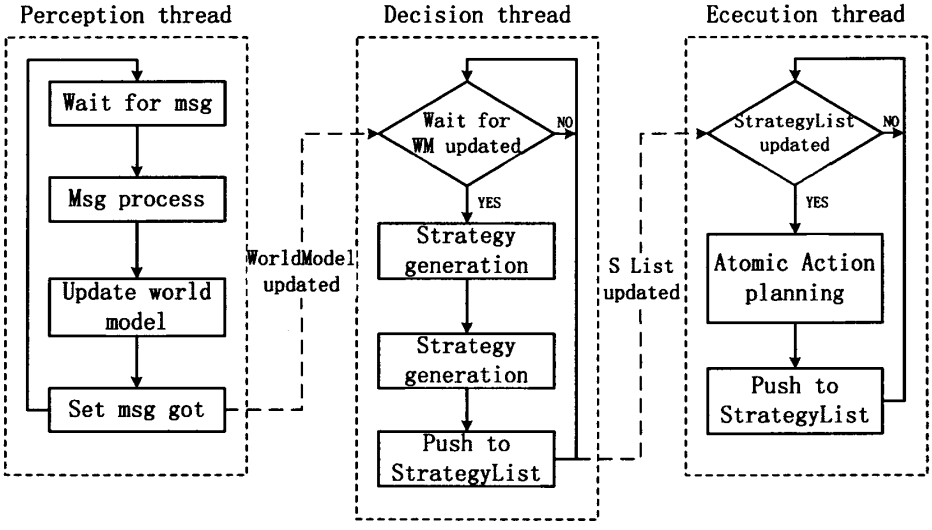


图5-3　三线程执行流程

上节讨论了不可靠通信条件下多智能体的决策基本框架。很多情况下，智能体可能不会接收到队友的消息，这时，根据观察到的场上信息独立地做决策，而在接收到队友消息的时候优先结合队友信息做决策。以便这种决策框架尽可能有效地利用比赛的条件。

5.3　基于 MAXQ 任务层次分解的决策问题建模

Robocup 2D 环境中，比赛双方共 22 名球员（Player），每名 Player 有 6 个状态变量，分别是 2 个位置分量，2 个速度分量，1 个身体朝向和 1 个头朝向。球有位置和速度共四个状态变量。每个状态变量都是连续的，即使是粗略的将每个变量离散化为 1000 个离散值，总共也将有 10^{408} 个状态，使用传统将状态平铺式（flat）MDP 求解算法显然不适合。不仅如此，单单是状态的表示都无法用这种平铺式的枚举状态表示方法。一般，研究者都会选择因子式的状态表示方法，采用因子式表示的 MDP 的策略求解算法[29]。

5.3.1　问题的基本 MDP 建模

通过上一节的世界状态更新方法，可以得到近乎真实的比赛状态信息。用得到的状态信息可以将智能体的决策过程按照 MDP 的模型建模。MDP 建模需要确定问题的状态空间、动作空间、状态转移函数和报酬函数。

- 状态空间（State Space）：这里将对手和队友都看成环境的一部分，不同的是队友是可以利用的状态信息，而对手则是要尽量远离的。通过观察信息的更新，可以用一个固定长度的、包含状态变量的向量来因子化的表示状态。状态向量包括场上 23 个对象（自身、10 个队友，11 个对手，1 个球）的位置、速度和角度等信息。
- 动作空间（Action Space）：动作空间包括 2D 平台所定义的原子动作，如 dash，kick，tackle，turn 和 turn_neck 等。这些动作都有连续的参数取值，也就构成了问题的连续动作空间。
- 状态转移函数（Transition Function）：由于队友和对手球员都可以自主决策，使得预测环境变化，即状态的转移很难显式的表示。假设其他的球员智能体都执行一种共同的行为模型：当可以踢到球时，执行一个随机的踢球动作，其他情况下执行随机的移动动作。对于原子动作，根据比赛平台的动作模型定义，可以确定个原子命令的状态转移。
- 报酬函数（Reward Function）：按照足球比赛的回报情形，进球才有 1 的回报，那么比赛中很可能上千步的原子动作后才有一个回报。直接使用这种报酬机制进行前向策略搜索很可能在搜索终止时也没有任何回报，这就无法表示策略的好坏。所以，对每一个子任务或行动定义一组伪报酬（pseudo-reward）函数，来表示每个子任务或行动的效果，以保证能够搜索到好的策略。

5.3.2 任务层次分解

本节首先介绍将原 MDP 进行层次分解，定义各个子任务，并分析分层值函数的表达。按照第 4.1 节的 MAXQ 分解方法，将上述的原始 MDP 分解成四个层次（除了根任务），如图 5-4 所示。

从图中可以看出，不同的子任务按一定的层次关系组织在一起。其中，根任务即原 MDP 的求解任务，但是按照 MAXQ 方法，要求解根任务相当于求解整个分层任务，因此，求解了分层结构就得到了原始决策问题的解。

下面自下而上分解图中每层的子任务：

- 原子动作层：kick，turn，dash 和 tackle。它们即是原始 MDP，即足球仿真平台定义的原子带参数的底层动作。这里给每一个该层的原始动作-0. 1 的即时报酬，以此保证最优策略将量快地到达一个目标状态。

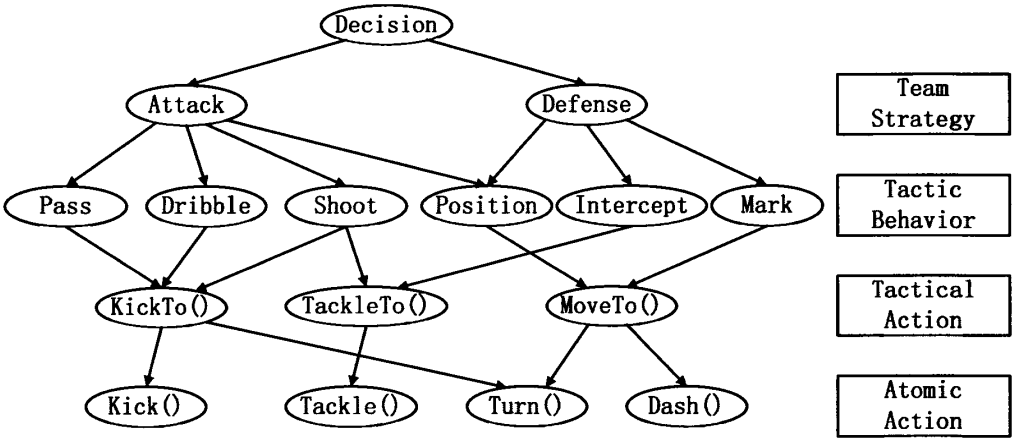


图5-4　任务分解图

- 技术动作层：KickTo，TackleTo 和 MoveTo。前两个技术动作是将球以特定的速度踢向指定的方向。MoveTo 子任务实现是智能体从一个位置移动到目标位置。
- 战术动作层：Shoot，Dribble，Pass，Intercept，Block，Trap，Mark 和 Formation。这些行为动作是结构中较高层的行为。1）Shoot 动作将球踢进对方球门；2）Dribble 将球朝着某个方向带；3）Pass 将球传向某个队友，将进攻权交给队友；4）Position 移动到一个较好的攻防位置，使整个球队保持较好的攻防阵型；5）Intercept 截球，即要尽快的夺到球；6）Block 阻挡对方带球队员；7）Trap 扰乱对方控球队员的进攻，伺机夺球；8）Mark 标记相关的对手，时刻关注该对手的行为并加以阻挠；9）Formation 保持防守阵型。
- 团队策略层：Attack 和 Defense。高层策略判断，即在一定形式下采取进攻策略，一定情况下采取防守策略。
- 根任务：Decision。即原 MDP 的任务。他通过递归地解决更低层的任务最终解决根任务，得到原问题的最优策略。

分层结构图中技术动作层和原子动作层子任务带参数，每个不同的参数对应不同的任务。另外，分层结构也隐式的引进了状态抽象。例如，MoveTo()子任务只需要考虑该智能体本身的状态信息，而无需考虑其他球员的信息，KickTo()子任务也无需考虑其他球员的状态信息。

在求解 MAXQ 分层策略时，完成函数的近似计算是至关重要的。在 RoboCup 2D 环境下计算完成函数就更难了。首先，由于在线决策时间的限制，前向搜索往往不能运行足够的深度；其次，由于环境的不确定性，以及不确定对手的行为选择，未来状态很难预测。

5.3.3 求解过程

根据 4.2 节介绍的应用不可靠通信的决策规划框架以及 4.3 节提出的实时策略规划算法，设计球员决策规划的流程如下图 5-5 所示。

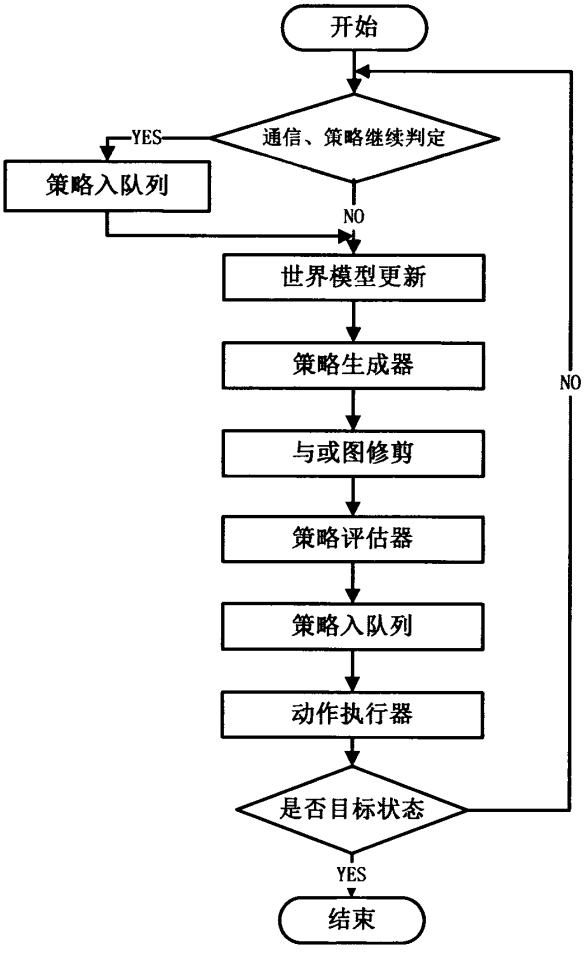


图5-5　决策规划流程图

通信内容处理和策略继续判定使用4.2节介绍的方法处理。其中策略是否继续的判定，通过计算战术行为层一步执行的后续状态概率是否与上步策略的第一步后续状态概率相近来决定。

决策系统首先将通信获得的队友策略和上一步的可继续执行的策略（如果有）放入策略列表中。为了不浪费本决策周期的时间，仍然根据当前状态进行决策。首先，策略生成器产生当前可执行的策略，并修剪掉明显不好的策略。通过策略评估器计算各策略路径的值函数，将较好的策略压入策略列表中。在决策周期近结束的时候，在列表中选出最优的策略。当然，这时首先考虑协调队友通信的策略信息。一个状态可行策略空间较大的时候，

决策周期内有可能不能评估所有可行策略，这时，将评估出的较好的策略即时放入列表中，周期结束时选择当前评估出的策略中最好的一个执行。

策略规划器根据当前状态，计算各可选行动及其后续状态和发生概率，将其存放于一个与或图中。与或图的根节点是当前状态，叶子节点是终止状态或者到达决策规划最大深度时的状态。

为了简化，我们再做另外一个假设，即在某一状态下，执行一个子任务，状态将转移到下一个固定的状态。这样假设的原因是，第一，这将有效地降低与或树的复杂度；第二，对于与或图中深度大于 2 的层，即从预测的状态生成的后续行动和状态，没有必要很精确的预测。模拟执行一个子任务，很容易计算出这个子任务的成功率，对于成功率小的子任务，没有必要进行下一步的评估。但是需要注意，在与或图中前两层，如果一个子任务可能产生多个概率接近的后续状态，仍然需要考虑多个可能结果，但是我们仍然最多取两个可能的后续状态。

下图是利用上述简化的规划器生成的与或图的一个示例：

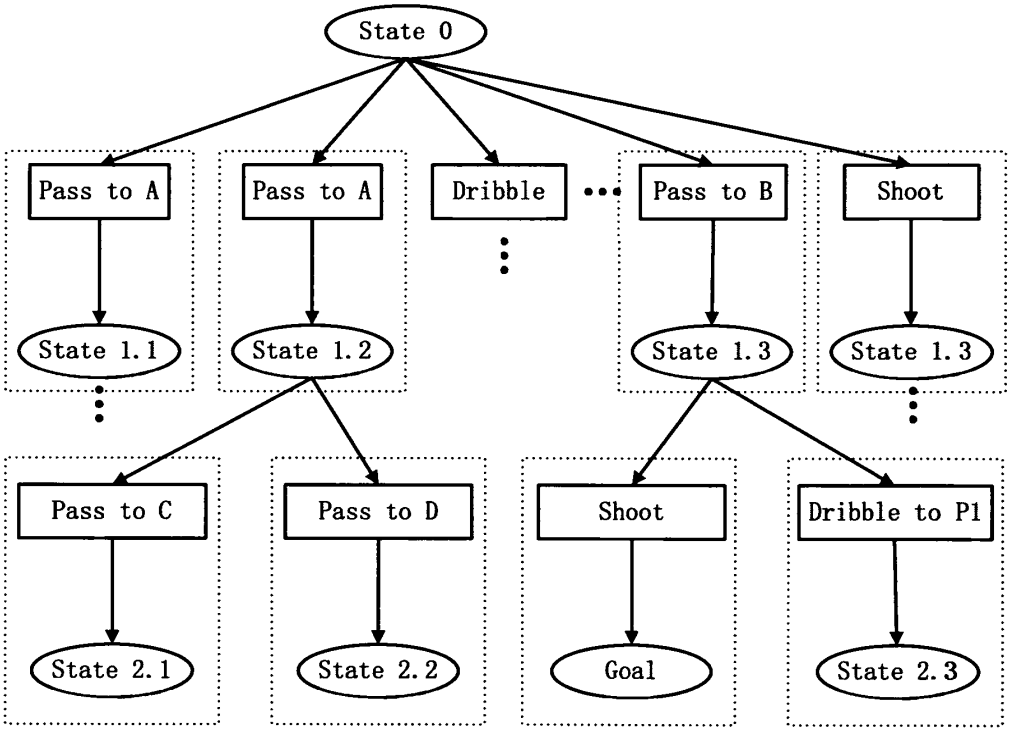


图5-6　行为生成器生成的与或图示例

图中，虚线框所示为每个可行子任务，及其固定后续状态对。这样，我们可以将以上

与或图转化为树结构，每一个子任务状态对为一个节点。其中，前两层出现一个子任务，多个后续状态的情况，可以分解成多个子任务状态对。

根据 MAXQ 分层的结构，我们知道，图中每一个行动节点本身对应 MAXQ 分层中的一个子任务，一个子任务节点由递归的包含它的孩子任务或基本原子任务构成。子任务的递归结构如 4.3 节所示，在此不再赘述。

如 4.3 节所介绍，与或图中每一条从根节点到叶节点的路径都是一个可行的从当前状态到一个终止状态的策略。评估器通过计算每个策略的分层值函数，根据分层值函数的大小来决定策略是否可执行。这里主要有两点需要考虑，首先路径上各预测状态的概率，即该策略路径的安全性；策略到达的终止状态的好坏，例如，如果两条策略路径的分层值函数差不多，那么终止状态是目标状态（如得分）比非目标状态的策略要好。通过分层值函数比较，将较好的策略，即值函数较大，且终止状态较好的策略放入策略列表中，在决策周期近结束的时候，比较列表中策略的好坏，取其中最好的一个执行。

从以上分析可见，要完成这一决策规划过程，策略生成器生成与或图的行动选择方法及后续状态预测方法，以及策略评估器，即分层值函数的计算是最关键的两点内容。将在下面介绍。

5.3.4 策略生成器

策略生成器根据当前比赛状态，生成策略路径。有两个关键步骤，第一，在一个状态下如何选择动作，第二，模拟执行一个选定的动作后，如何预测后续状态。

对于第一个问题，即生成行为的问题。首先，对于较低层的技术动作和原子动作，如4.2节所讨论，我们一般离线计算较好的子策略和值函数。所以关键在于战术动作，即Pass、Dribble、Shoot等行动的生成。对每种战术动作分别规划。例如，传球行为首先考虑可以传给队友方向上的Pass行为，如果没有好的选择，再考虑持球者周围各方向（离散化）的Pass行为；Dribble首先尝试目标点距离较远处带球的规划，在规划目标点距离中等以及较近的带球可能。各个战术行为的选择和规划不再一一赘述。

对于第二个问题，我们也分层考虑。对于与或图中前两层，即从当前实际状态预测行为的后续状态，我们用较精确的预测方法。而对于后面层次，精确地预测则不那么必要。这里，我们假设一个战术行为过程中，除了战术执行者和战术目标对象外，其他球员的位置只考虑它们的辐射范围。辐射范围的概念在3.3节有详细介绍。

5.3.5 分层值函数计算

本节首先介绍报酬函数的设计，然后说明值函数的计算方法。

1. 报酬函数

报酬函数按任务图分解图的层次设计。任务分解时介绍，最底层即原子动作层报酬值统一为-0.1。以促使智能体以尽可能少的步数完成任务。技术动作层的报酬函数可以由技术动作自身值加上该动作需要执行原子动作值的累积来表示。其中，完成该动作需要执行的除第一个外的原子动作的累积值即其完成函数。

技术动作层的报酬函数需要考虑如下（抽象）状态变量:球的位置；是否有射门路径；相对于对手防线的位置情况；与相近的对手球员的位置；执行动作的时间等。

球的位置属性即球在场地的位置与行为值的对应关系，如下图所示。在据对方球门越近的地方，有越高的值，而球进入对方球门，值最高。场上两点间球的属性值之差对应的是战术行为层的伪报酬函数，记为 $\tilde{R}_{tb}(a,s)$。而对手防守的相对位置对值的影响与场上位置的值正好相反，是一个负值，并且在离对手球员越近的位置绝对值越大。表现为场地值映射图中对手球员相对位置处的值发生凹陷。行为执行时间属性体现在完成战术动作的更低层动作中。战术动作的值函数即是一个战术动作的即时报酬和其子任务累积值的和。

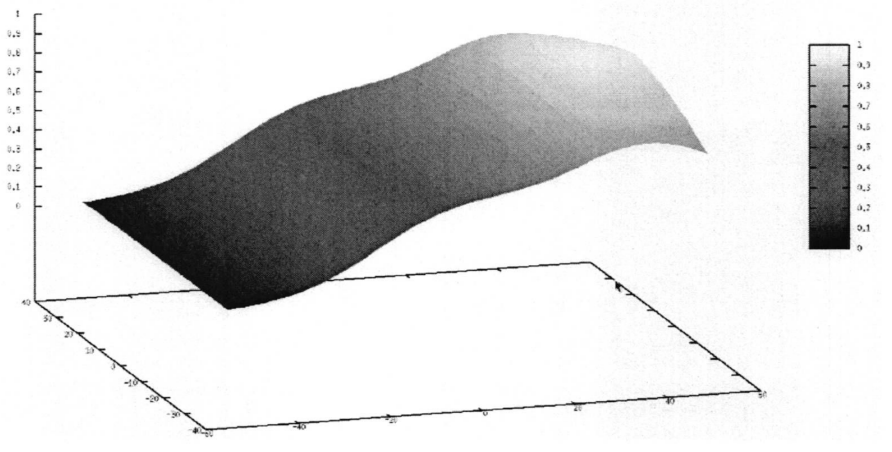


图5-7　场地值映射图

2. 分层值函数计算

4.3 节介绍了分层值函数的概念。即一个策略路径的值函数是其从根任务开始的各层子任务的递归值函数，每个子任务的递归值函数又由子任务执行的其第一个后代子任务和

完成盖子任务的完成函数值。所以分层值函数的计算需要递归的计算各层动作的值。如4.3节所述，较低层的子任务可以通过经典的值迭代或启发式搜索方法找到其最优策略和最优值，这样，较低层的值就可以离线计算，而高层需要考虑更多状态变量的子任务的值则在线计算。

在RoboCup 2D问题中，较低层指原子动作层和技术动作层。技术动作层的最优策略和最优值可以由经典的值迭代方法求得[29]。高层即战术行为层的值按照MAXQ值函数分解的定义，需要知道完成一个子任务的最优策略。这事先是不能确定的，所以我们使用一些近似方法来计算完成函数。这里使用和[20]相似的基于重要性采样的近似计算方法。在4.3节有较详细的介绍，在此不再赘述。

以上我们得到了低层子任务的值，以及完成函数。一个策略路径的值根据式4-3—式4-5的分解公式计算，其中式4-4第一项值函数$V^{\pi}(a,s)$表示在状态s下，执行子任务a的值函数大小。它由战术行为层的直接报酬函数，加上完成该战术行为的子任务的累加值函数得到，即：

$$V^{\pi}(a,s)=\tilde{R}_{lb}^{\pi}(a,s)+Q^{\pi}(i,s,m_a) \tag{5-1}$$

其中，$Q^{\pi}(i,s,m_a)$表示在状态s，执行的a子任务m_a，直到完成子任务a的值函数。由此，我们可以由式4-4得到一个策略路径的值。

5.4 RoboCup 2D决策实验

实验主要从两个方面讨论算法的性能。第一，改变决策规划的深度N，即行为预测的步数，检验怎样的决策深度最为有效；第二，用基于MAXQ值函数层次分解的算法与普通任务层次分解算法比较，检验MAXQ-RTP算法的实时性和总体效果。以下为叙述方便，将本文的球员智能体程序称为Sunflower。

从MAXQ-RTP算法步骤可以看出，决策规划深度N，即对应策略与或图层数，表示决策过程中从初始状态到预测的目标状态或最大预测步数时对应的状态。如示意图5-8所示。对于这个参数，[74]用搜索树表示策略路径，其实验得出决策步数与决策结果基本无关的结果，这个结果值得讨论。因此实验首先对该参数进行试验，分别取不同的N，与相同的对手比赛，观察并分析结果。

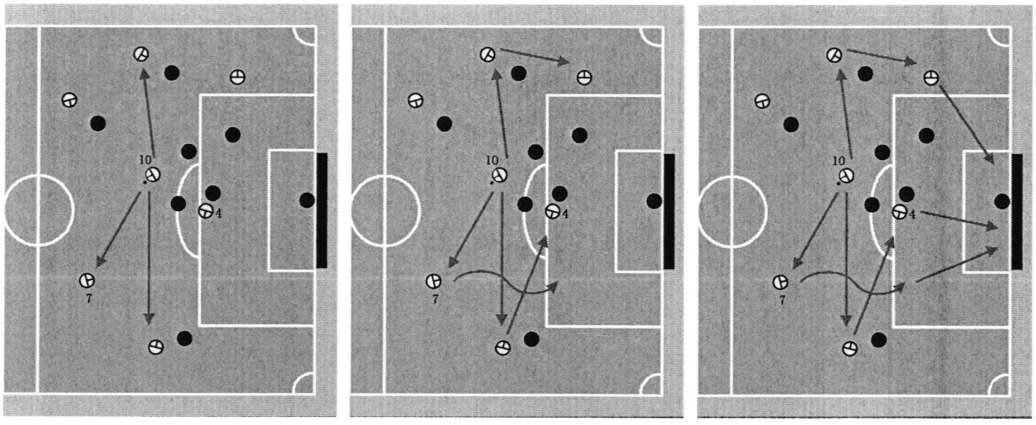


图5-8　决策深度 N 示意图

图 A、B、C 中，决策深度 N 分别为 1,2 和 3

RoboCup 2D 组委会规定，每年参赛的队伍必须在比赛结束后将自己的比赛程序发布。有些球队甚至会将自己的源代码发布，例如，中科大的 WrightEagle 队伍发布的 WrightEagleBase 和日本的 HELIOS 队伍发布的 Agent2d 程序源代码。本文的程序是基于 WrightEagleBase4.0 版本的基本框架和底层动作代码，加入本文的决策算法完成的。试验的第二部分，将以这两个队伍为对手，验证 MAXQ-RTP 算法的性能。

5.4.1　策略质量与决策深度关系实验

[74]的实验指出，决策规划的深度对球队策略质量并没有决定性的影响，即不论是进行一步预测规划，还是多步，对比赛结果没有质的影响。从经验上，我们知道在棋类等博弈游戏，对未来状态进行越远的预测应该得到越好的结果。因此，这部分我们通过实验讨论决策深度对决策质量的影响。为了便于比较，这里我们均以 WrightEagleBase 为对手，通过设置 Sunflower 的决策规划深度，来比较各深度决策的质量。每个深度进行 10 场比赛，统计结果如下表：

表5-1　决策深度与决策质量实验统计结果

决策深度 N	1	2	3	4	5
场均净胜分差	4.3	6.7	6.4	6.5	5.9

从表中可以看出除了决策深度从1改为2时，决策质量有较大提高外，再次加大决策深度对策略质量并没有质的影响，甚至深度过大时，性能还稍有降低。我们任务究其原因，

由于MDP模型对对手和队友模型做了大量简化，尤其是在状态预测时没有考虑其他球员的可能策略，在决策过程中，状态估计存在较大误差，导致较深层的状态预测不准确造成的。这个不足之处在MDP模型中很难有实质性的改善，未来可以尝试将本文的层次分解在线决策算法推广到随机博弈模型中，考虑队友和对手的策略影响进行状态预测来解决这个问题。

5.4.2 任务层次分解算法实验

本文采用 WrightEagleBase 和 Agent2d 球队作为对手。这两个队伍也采用任务层次分解的方法规划策略，其中 WrightEagleBase 采用的分层方法与本文相似，但是规划时只考虑一步战术行为的规划。Agent2d 球队也主要对战术行为层进行在线规划，但是评估时只考虑战术行为层的报酬。

实验用 Sunflower 与 WrightEagleBase 和 Agent2d 各进行 20 场比赛，Sunflower 的决策深度统一设为 3。除了统计比赛的比分等结果外，还统计了每个决策周期平均的决策时间信息，以检验算法的实时性能。下表 5-2 是比赛的结果统计，表 5-3 是决策时间的部分统计结果。

表5-2 Sunflower V.S. WrightEagleBase & Sunflower V.S. Agent2d 统计结果

对手	比赛场数	胜场	打平	负场	总比分	胜率
WrightEagleBase	20	20	0	0	164 : 36	100%
Agent2d	20	17	2	1	89 : 52	85%

表5-3 单步决策时间统计

对手	最大时间(ms)	最小时间(ms)	平均时间(ms)
WrightEagleBase	59	9	41.3
Agent2d	67	13	44.6

从比赛结果统计可以看出，Sunflower 以较大优势胜出 WrightEagleBase 队，同时也以一定优势胜出 Agent2d 队。之所以面对 Agent2d 时没有以大比分胜出，是由于 Agent2d 的底层动作算法做得比较出色，动作成功率较高。受益于任务层次分解分解方法，Sunflower 对状态空间进行抽象，通过设计的实时规划方法，可以快速的得到较好的分层策略。

从单步决策时间统计表可以看出，智能体每步决策的时间不等，并且最大单步决策时间与最小时间相差较大。这主要是由于智能体在遇到不同比赛情形时的可选动作空间相差较大，但平均起来，单步决策时间一般在 40ms 左右，相对于 RoboCup 2D 比赛 100ms 的

决策周期限制来说，决策规划的实时性完全能够满足。

5.5 本章小结

本章根据 RoboCup 2D 平台的特点，对球员智能体进行 MDP 建模，并根据 MAXQ 值函数分解方法将智能体模型进行分解，定义各子任务和值函数。根据第四章的 MAXQ-RTP 算法设计智能体的在线策略规划程序。实验结果表明该方法性能较好，能较好地处理连续状态空间，但是可以进行状态抽象的情况。在保证决策规划实时性的同时，能得到较优的策略。

但同时也可以看出，该处理方法也有不足。由于对对手和队友的决策模型做了大量的简化，一些结果受到较大影响。下一步工作主要是在随机博弈模型框架下，考虑队友和对手策略，来解决这些问题。

第 6 章　结论与展望

6.1　结论

本文旨在研究智能体的决策规划方法，使得智能体能够自主地选择理性的策略，实现相互之间的协同合作，共同完成系统的总体任务。

首先讨论了 Markov 决策过程的一些经典的求解方法，讨论了大规模不确定性决策问题的挑战和常用的解决方法。

在集中式控制模式下，基于任务层次分解的决策框架，设计了一种通过计算策略效用函数，并结合博弈论相关理论思想的方法来选择球员智能体行动的算法。实验证明，该方法能较好地实现智能体之间的协调配合，有效地完成系统总体目标。

分布式控制模式情况下，分析了该情况下的决策问题的特点，将智能体进行 MDP 建模。设计了一种有效地利用受限的感知和通信条件的多智能体决策基本框架结构。鉴于机器人足球比赛连续状态空间和动作空间的特点，以及较高实时性的要求，采用基于 MAXQ 值函数分解的方法将智能体进行任务层次分解建模。在基于与或图的策略表示方法基础上，提出了一套在线策略规划的算法 MAXQ-RTP，来求解最优分层策略。实验结果表明，该方法有较高的效率，能够快速的计算出较好的策略。

6.2　展望

同时，该方法也有不足之处。通过对队友和对手模型的简化处理，使得决策规划中预测未来状态的精度较低，影响了决策的效果。未来，需要将该任务分解方法及求解算法扩展到随机博弈模型中，通过考虑队友和对手的策略来规划最优的对策。通过本文方法处理大规模状态空间问题的能力和实时性特点，加上随机博弈模型考虑其他智能体的策略，将可以有效地解决多智能体的协调协作问题。

参 考 文 献

[1] Russell, S., P. Norvig, Artificial Intelligence:A Modem Approach[M]. 2003: Pearson Education, lnc.

[2] Wooldridge, M., N.R. Jennings, Intelligent agents: Theory and practice[J]. Knowledge Engineering Review, 1995. 10(2): 115-152.

[3] Wooldridge, M. Agent-based software engineering. in Software Engineering[J]. IEE Proceedings-Software, 1997, 144(1): 26-37.

[4] 李镇宇, 多主体系统决策问题研究及在 RoboCup 中的应用[D], 2005.

[5] 蔡自兴, 徐光佑, 人工智能及其应用[M]. 2004, 北京: 清华大学出版社.

[6] Sen, S., Adaptation, Coevolution and Learning in Multiagent Systems[J]: Papers from the 1996 AAAI Symposium, March 25-27.

[7] 李龙, 基于价值的机器学习方法及其在 RoboCup 仿真 2D 中的应用[D], 2009.

[8] Noda, I., et al., Overview of robocup-97[J]. RoboCup-97: Robot Soccer World Cup I, 1998: 20-41.

[9] 石轲, 基于马尔可夫决策过程理论的 Agent 决策问题研究[D], 2010.

[10] 王寻羽, 朱淼良, 智能机器人的分布式虚拟环境研究[J]. 计算机研究与发展, 2000. 37(6): 684-691.

[11] 范长杰, 陈小平, 实时动态规划的最优行动判据及算法改进[J]. 软件学报, 2008. 19(11): 2869-2878.

[12] 刘佳, 陈增强, 刘忠信, 多智能体系统及其协同控制研究进展[J]. 智能系统学报, 2010. 5(1): 1-9.

[13] 孟宪春,丁承君, 段萍, 多智能体技术的发展和应用现状[J]. 河北工业大学学报, 2006. 35(3): 6-12.

[14] 杨煜普, 李晓萌, 许晓鸣, 多智能体协作技术综述[J]. 信息与控制, 2001. 30(4): 337-342.

[15] 胡凡, 基于 RoboCup 仿真平台的机器人足球协作策略的研究[D], 2009.

[16] Kuo, C.-H. and T.-S. Chen. Modeling and control of autonomous soccer robots using high-level Petri nets[C]. in SICE Annual Conference 2010, Proceedings of. 2010. IEEE.

[17] 吴锋, 基于决策理论的多智能体系统规划问题研究[D], 2011.

[18] Barto, A.G., S. Mahadevan, Recent advances in hierarchical reinforcement learning[J]. Discrete Event Dynamic Systems, 2003. 13(4): 341-379.

[19] Dietterich, T.G., Hierarchical reinforcement learning with the MAXQ value function

decomposition[J]. J. Artif. Int. Res., 2000. 13(1): 227-303.

[20] Bai, A., F. Wu, and X. Chen, Online planning for large MDPs with MAXQ decomposition[C], in Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 32012, International Foundation for Autonomous Agents and Multiagent Systems: Valencia, Spain. 1215-1216.

[21] 李镇宇, 陈小平, 基于 Markov 对策的强化学习及其在 RoboCup 中的应用[J]. 计算机工程与应用, 2005. 41(27): 202-204.

[22] Newell, A., H.A. Simon, Human problem solving[M]. Vol. 14. 1972: Prentice-Hall Englewood Cliffs, NJ.

[23] Nilsson, N.J., Principles of artificial intelligence[M]. Symbolic Computation, Berlin: Springer, 1982, 1982. 1.

[24] Bonet, B., H. Geffner. Labeled RTDP: Improving the convergence of real-time dynamic programming[J]. in ICAPS. 2003, 3: 12-21.

[25] Bertsekas, D.P., et al., Dynamic programming and optimal control[M]. Vol. 1. 1995: Athena Scientific Belmont, MA.

[26] Stroock, D.W., An introduction to Markov processes[M]. Vol. 230. 2005: Springer.

[27] Bellman, R., A Markovian decision process[R], RAND CORP SANTA MONICA CA, 1957.

[28] Feinberg, E.A., A. Shwartz, and E. Altman, Handbook of Markov decision processes: methods and applications[M]. 2002: Kluwer Academic Publishers.

[29] 范长杰, 基于马尔可夫决策理论的规划问题的研究[D], 2008.

[30] Bai, A., F. Wu, and X. Chen, Towards a Principled Solution to Simulated Robot Soccer[J]. 2012.

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

[32] Bonet, B., H. Geffner. Faster heuristic search algorithms for planning with uncertainty and full feedback[C]. in International Joint Conference on Artificial Intelligence. 2003. LAWRENCE ERLBAUM ASSOCIATES LTD,2003, 18: 1233-1238.

[33] Dean, T., et al., Planning under time constraints in stochastic domains[J]. Artificial Intelligence, 1995. 76(1): 35-74.

[34] Ferguson, D., A. Stentz, Focused dynamic programming: Extensive comparative results[R], 2004, DTIC Document.

[35] Barto, A.G., S.J. Bradtke, and S.P. Singh, Learning to act using real-time dynamic programming[J]. Artificial Intelligence, 1995. 72(1–2): 81-138.

[36] McMahan, H.B., M. Likhachev, and G.J. Gordon. Bounded real-time dynamic programming: RTDP

with monotone upper bounds and performance guarantees[C]. in Proceedings of the 22nd international conference on Machine learning. 2005.

[37] Smith, T., R. Simmons. Focused real-time dynamic programming for MDPs: Squeezing more out of a heuristic[C]. in Proceedings of the National Conference on Artificial Intelligence. 2006. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

[38] Korf, R.E., L.A. Taylor. Finding optimal solutions to the twenty-four puzzle[C]. in Proceedings of the National Conference on Artificial Intelligence. 1996. Citeseer.

[39] Korf, R.E. Finding optimal solutions to Rubik's Cube using pattern databases[C]. in Proceedings of the National Conference on Artificial Intelligence. 1997. JOHN WILEY & SONS LTD.

[40] Littman, M.L., T.L. Dean, and L.P. Kaelbling. On the complexity of solving Markov decision problems[C]. in Proceedings of the Eleventh conference on Uncertainty in artificial intelligence. 1995. Morgan Kaufmann Publishers Inc.

[41] Littman, M.L., S.M. Majercik. Large-scale planning under uncertainty: A survey[C]. in Workshop on Planning and Scheduling for Space. 1997, 27: 1-8.

[42] Barry, J., L.P. Kaelbling, and T. Lozano-Pérez, Hierarchical solution of large Markov decision processes[J]. 2010.

[43] Nadarajah, S. and K. Sundaraj, A survey on team strategies in robot soccer: team strategies and role description[J]. Artificial Intelligence Review, 2011: 1-34.

[44] Kontes, G., M.G. Lagoudakis. Coordinated team play in the four-legged robocup league[C]. in Tools with Artificial Intelligence, 2007. ICTAI 2007. 19th IEEE International Conference on. 2007. IEEE, 2007, 1: 109-116.

[45] Lau, N., et al. Multi-robot team coordination through roles, positionings and coordinated procedures[C]. in Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on. 2009. IEEE.

[46] Playne, D. Knowledge-based role allocation in robot soccer[C]. in Control, Automation, Robotics and Vision, 2008. ICARCV 2008. 10th International Conference on. 2008. IEEE.

[47] 黄波, 闫丽娜, 石兴喜, 基于优度值评价的多机器人任务分配方法[J]. 华中科技大学学报(自然科学版), 2010. v.38;No.316(01): 101-104+113.

[48] 陈建平, 杨宜民, 魏良, 基于效用值的足球机器人系统任务分配方法[J]. 计算机工程, 2011. v.37;No.371(01): 197-200.

[49] 章小兵, 刘艳春, 陈黎, 基于传球评价函数的 Robocup 传球策略[J]. 安徽工业大学学报（自然科学版）, 2011. 28(2): 171-174.

[50] 彭军, 吴敏, 基于行为预测的多智能体协作模型[J]. 计算机工程与应用, 2005. 41(9): 23-25.

[51] Wang, J., et al. Short-term forecast research based upon multi-robot soccer match[C]. in Intelligent Control and Automation (WCICA), 2010 8th World Congress on. IEEE, 2010: 4722-4725.

[52] Ghazikhani, A., H.R. Mashadi, and R. Monsefi. A novel algorithm for coalition formation in Multi-Agent Systems using cooperative game theory[C]. in Electrical Engineering (ICEE), 2010 18th Iranian Conference on. IEEE 2010: 512-516.

[53] Wang, G., et al. A multi-agent model based on market competition for task allocation: a game theory approach[C]. in Networking, Sensing and Control, 2004 IEEE International Conference on. IEEE, 2004, 1: 282-286.

[54] 刘小梅, 田彦涛, 杨茂, 基于博弈论的多机器人任务分配算法[J]. 吉林大学学报(信息科学版), 2010. 28(3): 256-263.

[55] Li, P., Y.-m. Yang. Layered Task Allocation in Multi-robot Systems. in Intelligent Systems[C], 2009. GCIS'09. WRI Global Congress on. IEEE 2009,1: 62-67.

[56] 吴广谋, 吕周洋, 博弈论基础与应用[M]. 2009.

[57] Parsons, S., M. Wooldridge, Game theory and decision theory in multi-agent systems[J]. Autonomous Agents and Multi-Agent Systems, 2002. 5(3): 243-254.

[58] 魏良, 杨宜民, 一种中型组机器人足球系统的任务分解方法[J]. 电脑与电信, 2011(3).

[59] 李继耀, 机器人足球仿真比赛中多智能体协作策略的研究[D], 2008.

[60] 罗俊涛, MiroSot 机器人足球比赛决策系统研究[D], 2008.

[61] Singh, S.P., Transfer of learning by composing solutions of elemental sequential tasks[J]. Machine Learning, 1992. 8(3): 323-339.

[62] McGovern, A., et al. Hierarchical optimal control of MDPs[C]. in Proceedings of the Tenth Yale Workshop on Adaptive and Learning Systems. 1998: 186-191.

[63] Dietterich, T.G. The MAXQ method for hierarchical reinforcement learning[C]. in Proceedings of the fifteenth international conference on machine learning. 1998,8.

[64] Barry, J.L., Fast approximate hierarchical solution of MDPs[D], Massachusetts Institute of Technology, 2009.

[65] Bertsekas, D.P., J.N. Tsitsiklis. Neuro-dynamic programming: an overview[C]. in Decision and Control, 1995., Proceedings of the 34th IEEE Conference on. IEEE 1995, 1:560-564.

[66] Parr, R.E., Hierarchical control and learning for Markov decision processes[D], University of California, 1998.

[67] Barry, J.L., et al., DetH: approximate hierarchical solution of large Markov decision processes[C], in Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Volume Three2011, AAAI Press: Barcelona, Catalonia, Spain. 2011: 1928-1935.

[68] Roy, N., G.J. Gordon, and S. Thrun, Finding approximate POMDP solutions through belief compression[J]. J. Artif. Intell. Res. (JAIR), 2005. 23: 1-40.

[69] Wu, F., S. Zilberstein, and X. Chen, Online planning for multi-agent systems with bounded communication[J]. Artificial Intelligence, 2011. 175(2): 487-511.

[70] Dietterich, T., An overview of MAXQ hierarchical reinforcement learning[J]. Abstraction, Reformulation, and Approximation, 2000: 26-44.

[71] Akiyama, H. and I. Noda, Multi-agent positioning mechanism in the dynamic environment[M]. RoboCup 2007: Robot Soccer World Cup XI, 2008: 377-384.

[72] Thrun, S., et al., Robust Monte Carlo localization for mobile robots[J]. Artificial Intelligence, 2001. 128(1): 99-141.

[73] 宋志伟, 基于逻辑马尔可夫决策过程的关系强化学习研究[D], 2006.

[74] Akiyama, H., S. Aramaki, and T. Nakashima. Online Cooperative Behavior Planning Using a Tree Search Method in the RoboCup Soccer Simulation[C]. in Intelligent Networking and Collaborative Systems (INCoS), 2012 4th International Conference on. IEEE 2012: 170-177.

致　谢

值此论文完成之际，对曾经提供指导帮助，和一起学习的人们充满感激之情。

首先衷心的表示对恩师杨马英教授的深深谢意。在读硕士的三年学习和生活中，杨老师给予了我悉心的指导和关怀，在自己的科研工作中，每一步走来都离不开杨老师给予的帮助、支持以及鼓励。此外，她不但为我提供了宽松的学习科研环境，更培养了我独立从事科研的能力。杨老师敏锐的学术洞察力，广博的知识，以及严谨的治学态度，令我深深钦佩，而她忘我的工作精神与平易近人的作风为我树立了榜样，对我潜移默化的熏陶将使我受益终身。

另外感谢机器人项目组的张聚老师，吴根忠老师，作为机器人项目组的导师，它们的工作和指导使我受益良多。

感谢工大仿人足球机器人团队的李晓瑜，李彤斐，唐俊淮，孙青，以及所有曾跟自己并肩拼搏过的队友，那是一段人生中永远珍贵的回忆。

感谢实验室的所有同学以及工大生活中的挚友，篮球场上的兄弟们，他们为我带来了很多快乐。无论以后走到哪里，我都会记得他们曾经出现在自己生命中的那些时刻。

感谢我的女友在我研究工作中的大力支持，她对我学习工作上的帮助，生活上的关心照顾，在此深表谢意！

最后，谨以此文献给我最亲爱的父母!他们对我最无私的爱以及默默的支持永远是我前进的动力!

攻读学位期间参加的科研项目和成果

参加的科研项目

浙江省项目：浙江省重点科技创新团队项目（2009R50014）。

发表的论文

[1] 贾玉博,杨马英. 基于行为效用预测的足球机器人角色实现[A]. 第三十一届中国控制会议论文集 D 卷[C]:,2012:6

[2] Yang Maying, Jia Yubo. Action Utility Prediction and Role Task Allocation in Robot Soccer System. 12th International Conference on Control, Automation, Robotics & Vision. 2012:112-117

[3] 贾玉博,杨马英. 基于任务层次分解的足球机器人决策规划[A]. 第二十四届中国过程控制会议. （已投寄）

奖励

2012 年 3 月，获浙江工业大学“运河杯”科技竞赛三等奖；

2012 年 11 月，获 RoboCup 中国公开赛仿人组技术挑战赛亚军和季军各一项。