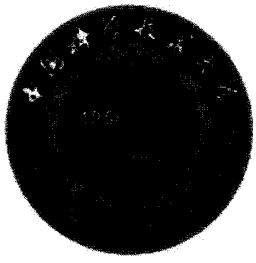
中国科学技术大学

硕士学位论文



仿真机器人足球中球员合作策略研究

作者姓名：陈荣亚

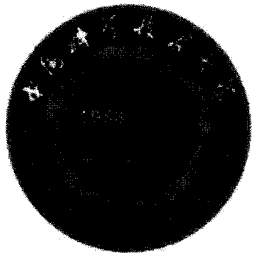
学科专业：计算机应用技术

导师姓名：陈小平 教授

完成时间：二〇一五年四月

University of Science and Technology of China

A dissertation for master's degree



Collaborative Planning in Simulation Soccer Robots

Author's Name: Rongya Chen

Specialty: Computer Application Technology

Supervisor: Prof. Xiaoping Chen

Finished time: April, 2015

中国科学技术大学学位论文原创性声明

本人声明所呈交的学位论文，是本人在导师指导下进行研究工作所取得的成果。除已特别加以标注和致谢的地方外，论文中不包含任何他人已经发表或撰写过的研究成果。与我一同工作的同志对本研究所做的贡献均已在论文中作了明确的说明。

作者签名：陈荣亚　　签字日期：2015·5·28

中国科学技术大学学位论文授权使用声明

作为申请学位的条件之一，学位论文著作权拥有者授权中国科学技术大学拥有学位论文的部分使用权，即：学校有权按有关规定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，可以将学位论文编入《中国学位论文全文数据库》等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、汇编学位论文。本人提交的电子文档的内容和纸质论文的内容相一致。

保密的学位论文在解密后也遵守此规定。

☑公开　□保密（___年）

作者签名：陈荣亚　　导师签名：陈小平

签字日期：2015·5·28　　签字日期：2015·5·28

摘 要

构造可以通过决策产生智能行为的智能体可以看作是人工智能现阶段的主要目标之一。各类决策算法使得智能体能够在多个方面近似做出人类可以做出的智能行为。在不确定性环境中，马尔科夫决策过程为智能体决策提供了基本的模型。

RoboCup 机器人世界杯成立的初衷是为了促进人工智能、机器人等领域的研究和技术的发展。最早成立的 RoboCup 仿真 2D 组是其中以智能体决策为重点内容的一个项目。

本文以 RoboCup 仿真 2D 机器人足球为实验平台，以马尔科夫决策过程相关理论为基础，来描述和处理大规模不确定性环境下的多智能体协作规划问题。本文涉及到的主要工作可以概括为以下三个部分：

- 本文实现了用于仿真 2D 比赛场景重现的 Trainer 和 rcsslogplayer 的改进，使得针对某个特定场景进行反复训练测试成为可能。由以前单纯使用比赛胜率的方式，改为从特定场景开始反复进行随机测试，其得到的结果更能证明某个方法的效果。
- 本文引入了 MDP 的分层分解技术，将其与 WrightEagle 中的反算技术相结合，并在守门员的决策中应用。通过改进守门员的站位决策，影响对方的传球行为，降低了对方突破我方防线、形成单刀球的威胁。
- 本文提出了解决多智能体协作问题的 MAXQ-MOP 方法。MAXQ-MOP 以 MAXQ-OP 算法为基本框架，引入了信念池的概念，使其应用到多个智能体的协作规划问题中。在人墙站位问题和多球员传接球协作的实验中，MAQX-MOP 表现出比传统方法更好的效果。

本文中所有的工作都是在 WrightEagle 队上改进实现的。

关键词：机器人世界杯；多智能体决策；马尔科夫决策过程；仿真 2D；MAXQ 分层分解技术

Abstract

At this stage, constructing intelligent agents which can produce intelligent behavior can be treated as main aim of Artificial Intelligence. Many decision-making algorithms can make good behaviors for agents which are approximated to what human do. Markov Decision Precess(MDP) provides basic model for agents' decision-making problem in uncertain environment.

RoboCup was originally designed to promote the development of research and technology of Artificial Intelligence, Robotics and other related areas. Simulation 2D League was found as the earliest league of RoboCup, and now it is one of the projects which focus on agents' decision-making. In this paper, our work is to process a large scale of multi-agent planning problem, with which we use Markov decision processes to describe the uncertainty of environment.

In this paper, Soccer Simulation 2D is used as experiment platform, and MDP is used to describe and handle multi-agent cooperative decision-making problem in large-scale environment of uncertainty. The work mentioned in the paper contains three parts as following:

- We realize and improve Trainer and rcsslogplayer, which can reappear scenes in Simulation 2D competition. They make it possible to test a particular scene repeatedly. We use winning ratio to decide whether a new method is effective or not in the past, and now using random testing repeated from a particular scene is more persuasive.
- We introduce a hierarchical decomposition for MDP, and combine it with inverse computation in WrightEagle, which has been tested in goalkeeper's decision-making. By improving goalkeeper's position, it affects opponents' pass behavior, and it reduces the menace of opponents' attack.
- We propose a method called MAXQ-MOP to solve the problem in multi-agent collaboration. MAXQ-MOP uses MAXQ-OP as basic framework, and by introducing the concept of belief pool, it has a better performance than traditional methods in man wall problem and pass between teammates.

In this paper, all the work is achieved on WrightEagle.

Key words: RoboCup; Multi-agent decision-making; Markov Decision Process; Simulation 2D; MAXQ hierarchical decomposition

目 录

插图目录

表格目录

算法目录

第一章 绪论

自从世界上第一台计算机 ENIAC 于 1946 年研制成功以来，计算机硬件设备和人工智能技术飞速发展，并取得了长足的进步。各种智能计算机系统已经逐渐深入到人类社会生活的方方面面，并起到越来越重要的作用。作为正式成立于 1956 年的计算机科学的分支学科，人工智能（Artificial Intelligence）也是一门广泛的交叉和前沿学科。它涉及到机器人学、数学、物理学、心理学、计算机科学、控制论等学科，涉及到的问题如：感知和交流、移动和操作物体的能力、推理和规划、知识和学习等。现阶段比较流行的研究方法包括：计算智能、统计学习、逻辑推理、数值优化等。在不确定性环境下，多个智能体的合作行为规划问题是本文主要关注点。本章剩余内容主要安排如下：第 1.1 节介绍智能体的相关概念；第 1.2 节介绍多智能体系统的相关内容；第 1.3 节分类介绍了马尔科夫决策过程的相关理论；第 1.4 节给出了本文的主要内容和结构安排。

1.1 智能体

1.1.1 智能体的定义

智能体（Agent）又称智能主体。它是可以不断观察环境并作用于环境，以完成一定目标的物理或抽象的实体。关于智能体的定义有如下几种：

- 智能体是状态有信念、能力、选择和承诺等精神组成的一个实体[1]；
- 自主的智能体存在于环境中，同时又是环境的一部分，能够不断感知环境，并且随着时间的推移，为了完成其计划而作用于环境的系统[2]；
- 智能体是可以感知和作用于环境的决策系统[3]；
- 智能体是像机器人或专家系统等、可以控制问题求解的机制的计算单元[4]。

综合来看，可以观察环境并根据环境做出反应、通过指引自己的行动以完成某些目标的自主主体可以被称为智能体，其通常具有如下特征：

- 反应性（Reactive）：智能体可以对外界环境的刺激做出一定的反应；
- 自主性（Autonomy）：智能体具有自我调节、自我管理的能力，不仅能接受外界环境的刺激，还可以根据环境的变化自动对自己的行为和状态进行调整；

- 主动性（Proactive）：智能体可以主动地采取一定的行动，来适应外界环境发生的变化；
- 可进化性（Evolvability）：智能体能够积累经验、学习新的知识，并相应地调整自己的行为；
- 社会性（Social）：智能体具有与其他智能体或人合作的能力，不仅仅是独自完成任务，不同的智能体可以根据各自的决策与其他智能体进行交互，以协同解决某一问题或达到某一共同目标[5]。

智能体可以看作是能自主决策的一类特殊智能对象，且智能体之间可以采用支持知识传递的语言进行通信。

智能体主要有：简单反射智能体（Simple reflex agents）、基于模型的反射智能体（Model-based reflex agents）、基于目标的智能体（Goal-based agents）和基于效用的智能体（Utility-based agents）。

常见的智能体技术在科技领域和日常生活中的应用有：医疗诊断系统（Medical Diagnosis System）、网络爬虫（Web Crawlers）、手机辅助软件（如 Apple 公司的 Siri、Google 公司的 Now）、智能机器人（Intelligent Robots）、炼油厂控制器（Refinery Controller）、太空探测器（Rover）、卫星图片分析系统（Satellite Image Analysis System）等等，在人类社会的发展中扮演着越来越重要的角色。

1.1.2 智能体所处的环境

智能体所处的环境有以下一些特点[6]：

- **是否完全可观察**：如果智能体在任何时候都能够及时地通过传感器获取环境的完整状态信息，则称这个环境是完全可观察的；否则就称为部分可观察的。完全可观察对于智能体来说更方便，因为不需要维护内部的状态信息来跟踪环境的变化。因为环境噪音和传感器的误差，多数研究应用的环境都是部分可观察的。
- **是否随机**：如果由当前所处的状态和智能体执行的行动，可以确定智能体所处环境的下一个状态，则称这个环境是确定性的；否则就称为随机的。
- **是否连续**：环境的状态信息、时间、智能体的感知及行动等有离散和连续的区别。

1.2 多智能体系统

1.2.1 多智能体系统的特点

多智能体系统[7]（Multi Agents System，MAS）是分布式人工智能[8]的一个重要分支。它是由多个可以互相影响的智能体组成的一个用计算机处理的系统，用来解决对于单个智能体来说很困难或者不可能完成的问题。

多智能体系统除了具有单个智能体的特点，一般还具有以下几个特点：

- 局部性（Locality）：多智能体系统中的每个智能体都拥有而且只拥有环境的部分信息；
- 分布性（Distribution）：多智能体系统中的智能体可以是同构的，也可以是异构的，即每个智能体具有不同的能力；计算过程是并行的或者异步的；数据是分布的或者分散的。

即使每个智能体所采用的个体策略（Individual Strategies）可能很简单，但是多智能体系统整体可以呈现出非常复杂的行为。

1.2.2 多智能体的决策和规划

多智能体系统中的智能体为了协同完成共同的目标时的决策行为称为多智能体决策[9]（Multi-agent Decision Making），其主要关注智能体之间的协同合作（coordination，cooperation）行为。智能体不仅需要考虑行为的当前效果，还要考虑该行为对将来决策的影响。智能体的这种多步决策问题也称作序列化决策问题[10]。一般来说，解决这类问题的过程称为规划（Planning）。规划问题的结果是一个能到达目标状态的行动序列（Action Sequence）或者选择行动的策略（Strategies），智能体按照这个行动序列执行能够完成某些目标任务。为了简化，经常把多智能体共同的目标任务细分成若干个可以并行处理的子任务（Task）。

多智能体决策的一般过程为：

1. 任务分配（Task Allocation）
2. 决策前合作（Coordination Before Planning）
3. 个体决策（Individual Planning）
4. 决策后合作（Coordination After Planning）
5. 决策执行（Plan Execution）

关于单个智能体决策问题，经典算法是基于确定的环境模型，不考虑状态转移的不确定性，如状态空间搜索，其将问题求解过程表现为在状态空间中寻找一条从初始状态到目标状态的路径。简单的状态空间盲目搜索有：深度优先搜索[11]

（DFS）、广度优先搜索[12]（BFS）、迭代加深搜索[13]（ID-DFS）、迭代加宽搜索[14]（IB-BFS）、柱形搜索[15]（Beam Search）等。

盲目搜索利用状态转移进行简单的搜索工作，没有利用状态本身的信息。如果在搜索过程中，优先访问具有特定的启发信息的节点，就有可能较快得到结果，这样的搜索称为启发式搜索。

当状态空间非常大或者无限大的情况下，局部搜索[16]只寻找一个优化的或合理的解，而并不搜索遍历整个状态空间。爬山搜索就是一种局部搜索。

1.3 马尔科夫决策过程

马尔科夫过程是具有马尔可夫性的一类过程。马尔可夫性也称为无后效性，是指过程中某阶段的状态一旦确定，则此后过程发展的概率规律与之前的历史无关的性质。

马尔科夫决策过程[17]（Markov Decision Process，MDP）是以俄罗斯数学家安德烈·马尔可夫的名字命名的具有一类普遍共性的过程。一个 MDP 可以用一个四元组 $<S,A,T,R$形式化表示，其中：

- S 表示智能体所处环境的状态集合，也称为状态空间。
- A 表示智能体可以执行的动作集合，也称为动作空间。
- $T:S\times A\times S\rightarrow[0,1]$ 是状态转移函数。$T(s'|s,a)=\Pr(s'|s,a)$ 表示智能体在状态 s 下执行动作 a 并进入下一个状态 s' 的概率。
- $R:S\times A\rightarrow$R 是报酬函数。$R(s,a)$ 表示在状态 s 下执行动作 a 后，智能体从环境中得到的报酬值（又称回报值，奖赏值）。

MDP 定义中的状态空间 S 和动作空间 A 有离散的，也有连续的。如果没有作特殊说明，本文中提到的 S 和 A 都默认是离散的。

马尔可夫决策过程与马尔可夫过程的区别是多了做决策的智能体主体。对这类决策问题的求解过程也称为规划，其求解方式可以简单地分为离线规划和在线规划两种。离线规划算法是指智能体与环境交互前计算出完整策略，然后在与环境交互的过程中仅仅需要执行策略即可。值迭代算法和策略迭代算法是经典的离线规划方法。在线规划算法是指事先不进行完整的离线计算，由智能体在与环境交互过程中实时地计算应该执行的最优行为。实时动态规划[18]（Real-Time Dynamic Programming，RTDP）算法、与或树搜索[19]（AND/OR Tree Search）算法和蒙特卡洛树搜索[20]（Monte-Carlo Tree Search）算法是典型的在线规划方法。

1.3.1 部分可观察的马尔科夫决策过程

MDP 中假定智能体可以完全确定自身所处的状态，即智能体对环境的观察是完整的。很多实际应用场景中，智能体不能完全观察到整个环境，或者智能体的观察具有一定的不确定性，这时可以用部分可观察的马尔科夫决策过程[21]（Partially Observable Markov Decision Process， POMDP）来建模。

POMDP 在形式上可以用六元组 $<S,A,O,T,\Omega,R>$ 描述，其中：

- S,A,T,R 的定义跟马尔科夫决策过程一致。
- $O:A\times S\rightarrow P(\Omega|A,S)$ 是智能体可能获得的所有观察的集合，即观察空间。
- $\Omega:S\times A\times O\rightarrow[0,1]$ 是智能体的观察函数。$\Omega(o|s,a)$ 是智能体执行了动作 a 并进入下一个状态 s 时，观察得到 o 的概率分布。

POMDP 可以转化成定义在信念状态空间上的 MDP[22][23]。其中，信念状态是智能体对当前环境状态的概率估计。

1.3.2 半马尔科夫决策过程

马尔科夫决策过程是时齐过程，即过程中任意状态之间转移需要的时间是一致的。换句话说，智能体执行任意行动的时间是相同的。非时齐马尔可夫决策过程也称为半马尔科夫决策过程 [24]（Semi-Markov Decision Process，SMDP）。

Suttona 提出了将 Option 概念作为 MDP 中的行动在 SMDP 中的替代[25]。即 Option 可以看作是对 MDP 中原子动作概念的一个扩展，其在形式上可以表示成一个三元组 $<I,\pi,\beta>$,其中：

- $I\subseteq S$ 是初始集合，即当且仅当 $s\in I$ 是，Option 开始执行；
- $\pi:S\times A\rightarrow[0,1]$ 是 MDP 的一个策略；
- $\beta:S^{+}\rightarrow[0,1]$ 是策略 π 的终止条件。

SMDP 可以看作是使用了 Option 的 MDP，一个 Option 也是一个 MDP 问题的子问题的策略。

1.3.3 分布式局部可观察马尔科夫决策过程

不论是 MDP 还是 SMDP、POMDP，建模的都是单个智能体的行动规划过程。要建模一个多智能体系统在不确定性环境中的规划过程，可以使用分布式局部可观察马尔科夫决策过程[26]（Decentralized Partially Observable Markov Decision Process，Dec-POMDP）。

Dec-POMDP 在形式上可以用一个七元组 $<I,S,\{A_i\},O,T,R,\{\Omega_i\}>$ 表示，其中：

- I 是智能体集合；
- S 表示全体智能体所处环境的状态空间；
- A_i 是智能体 i 可以执行的动作空间，$\vec{A}=\times_{i\in I}A_i$ 是全体智能体的联合行动集合；
- $O:S\times\vec{A}\times\vec{\Omega}\to[0,1]$ 是观察函数；
- $T:S\times\vec{A}\times S\to[0,1]$ 是多智能体系统的状态转移函数，$P(s'|s,\vec{a})$ 是全体智能体在状态 s 下去联合行动 $\vec{a}=\{a_1,a_2,\cdots,a_n\}$ 后系统转移到状态 s' 的概率，其中 n=$|I|$；
- $R:S\times\vec{A}\to$R 是报酬函数，$R(s,\vec{a})$ 表示在状态 s 下执行联合行动 $\vec{a}$ 后，全体智能体获得的立即报酬；
- Ω_i 是智能体 i 的观察集合，$\vec{\Omega}=\times_{i\in I}\Omega_i$ 是所有智能体的联合观察集合。

当$|I|$=1时，Dec-POMDP 模型即为 POMDP 模型；

当对于 $\forall t\forall\vec{o}\in\vec{\Omega}$ 都有 $s\in S$ 满足 $P(S(t)=s|\Omega(t)=\vec{o})=1$，Dec-POMDP 模型即为 Dec-MDP 模型。

1.3.4 部分可观察随机博弈

在博弈论中，随机博弈是一种由若干个智能体参与的、含有状态转移概率的动态博弈[27]。根据自身所处的状态，智能体选择策略并获得相应的报酬。然后博弈按照概率的分布随机地转移到下一个状态。

部分可观察随机博弈（POSG）是在随机博弈的基础上加入了部分可观察性。即智能体对当前状态不是完全可知，存在一定的不确定性。

POSG 在形式上可以用一个七元组 $<\tau,S,\{b^0\},\{A_i\},\{O_i\},P,\{R_i\}>$ 表示[28]，其中：

- τ 是智能体索引的集合，取值为 1，2,…，n。
- S 表示全体智能体所处环境的状态空间。
- $\{b^0\}:b^0\in\Delta(S)$ 是初始时刻的状态分布。
- A_i 是第 i 个智能体可以执行的动作空间，$\vec{A}=\times_{i\in\tau}A_i$ 是全体智能体的联合行动集合。
- $\{O_i\}:O_i$ 是第 i 个智能体的观察集合，$\vec{O}=\times_{i\in\tau}O_i$ 是所有智能体联合观察集合。
- P 是马尔可夫状态转移和观察概率集合，$P(s',\vec{o}|s,\vec{a})$ 是全体智能体在状态 s 下执行联合行动 $\vec{a}=\{a_1,a_2,\cdots,a_n\}$ 后系统转移到状态 s' 且获得联合观察 $\vec{o}=\{o_1,o_2,\cdots,o_n\}$ 的概率。
- $\{R_i\}:R_i$ 是索引为 i 的智能体获得的报酬函数。

一般博弈的过程是双方智能体互相追求各自报酬最大化的过程。

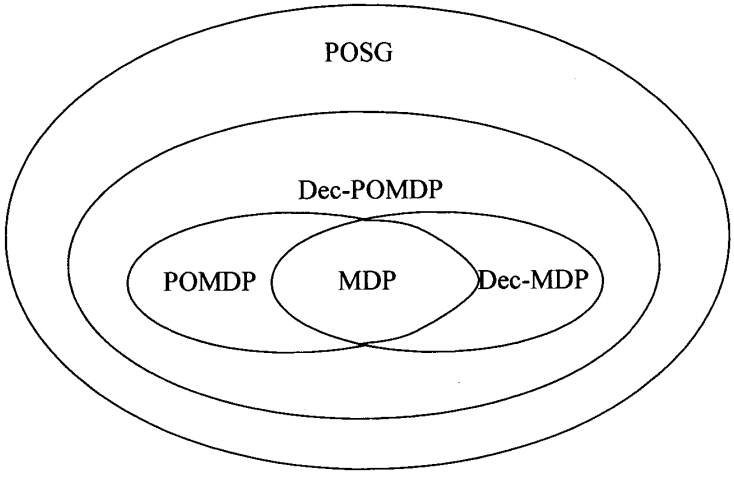


图 1.1 几种模型的关系

如图 1.1 所示，我们可以看到这一章所提到的各种模型之间的关系。其中，Dec-MDP 是 Dec-POMDP 的一种特例。一定意义上讲，Dec-POMDP 又可以看作是 POSG 的一个特例。

1.4 论文的主要内容和结构安排

本文所做的工作中，将 RoboCup 仿真 2D 足球中的每个球员建模为一个智能体，整个球队作为一个多智能体系统，利用马尔科夫决策过程相关理论对比赛过程进行建模和求解。利用分层分解技术，结合 WrightEagle 的反算技术，用于对手建模，并通过实验改进了守门员站位这一问题的效果。提出了基于 MAXQ 分层分解的多智能体在线规划算法 MAXQ-MOP。并利用 MAXQ-MOP 算法在 RoboCup 仿真 2D 足球比赛中，通过实验测试了其在人墙站位问题和多人多步传球问题。

本文大致分为三个部分：WrightEagle 仿真 2D 球队决策系统、马尔科夫决策过程的分层分解和多智能体分层协作规划。

第二章主要介绍了 RoboCup 仿真 2D 足球的一些背景内容，分析了 WrightEagle 仿真 2D 球队的理论模型。并介绍了本文的第一部分工作，也即基础工作：为了测试后续内容，改进开发了 Trainer 和 rcsslogplayer 工具。

第三章主要介绍了马尔科夫决策过程的求解算法，包括在线算法和离线算法两大类。然后重点介绍了使用分层分解技术处理马尔科夫决策过程，并用这种方法对 WrightEagle 球队进行建模。本文的第二部分工作就是将这种分层分解技术与 WrightEagle 的反算技术相结合，解决一类对手模型未知、需要对对手建模的智能体决策问题，并在守门员的站位问题中进行了实验，实验结果表明相比传统方法更为灵活有效。

第四章在第三章的基础上进一步扩展，将分层分解技术由单个智能体决策问题推广应用到多个智能体决策问题，提出了 MAXQ-MOP 方法，这是本文最重要的第三部分工作。MAXQ-MOP 方法可以用于通讯受限、队友模型已知的多智能体协作行为规划问题。通过任意球模式下的人墙排列问题、多球员传接球协作问题等典型场景对 MAXQ-MOP 和传统方法进行了检验，并对实验结果进行了分析。

第二章 WrightEagle 仿真 2D 球队决策系统

WrightEagle 机器人团队组建于 1998 年，并从 1999 年就开始参加每年的足球仿真 2D 组比赛。在最近 10 年的世界杯比赛结果中，WrightEagle 仿真 2D 队始终保持世界前两名的成绩。其中，包括 5 次世界冠军（RoboCup2006，2009，2011，2013，2014）和 5 次世界亚军（RoboCup2005，2007，2008，2010，2012）。

WrightEagle 仿真 2D 球队在备战世界杯比赛的过程中，研究多智能体系统中的决策问题和其他具有挑战性的相关问题。本文所有的工作都是在 WrightEagle 仿真 2D 球队上进行实验的。本章剩余内容主要安排如下：第 2.1 节介绍 RoboCup 仿真 2D 平台的一些相关背景知识；第 2.2 节介绍了改进开发的 Trainer 和 rcsslogplayer 程序，这是本文工作的第一部分，也是最重要的基础部分，用来在后续章节中测试相关方法的实验效果；第 2.3 节展示了 WrightEagle 的相关理论模型，包括用马尔科夫决策过程理论对球队的建模，和球队中用到的几个重要模块的结构模型；第 2.4 节给出了本章小结。

2.1 RoboCup 仿真 2D 平台

2.1.1 RoboCup 仿真 2D 机器人足球比赛

开始于 1997 年的机器人足球世界杯[1]（Robot Soccer World Cup，RoboCup）是每年举办一届的国际机器人比赛。这个项目创办的初衷是为了促进人工智能和机器人相关领域的科学研究与技术发展。同年举办的第 15 届国际人工智能联合会议（IJCAI-97）将机器人足球比赛正式列为人工智能领域的一项挑战。RoboCup 的官方目标是："到 21 世纪中叶，按照国际足联的相关比赛规则，由完全自主的人形机器人组成的足球队伍可以战胜人类足球世界杯冠军队"[29]。就近阶段而言，RoboCup 为智能机器人研究提供了广泛的技术标准问题和测试平台，可以用来检验人工智能领域前沿相关的最新研究成果。现在 RoboCup 机器人足球比赛包含很多组别，如：仿真组、小型组、中型组、人形组、标准平台组，还有一些不属于足球比赛的组别，如：家庭组、救援组等等。

仿真 2D 机器人足球（Soccer Simulation 2D）是 RoboCup 中最早的一个比赛项目。其研究内容集中于多智能体的合作与对抗[30]。仿真 2D 足球采用 Client/Server 模式，比赛服务器（Soccer Server）抽象了很多与高层决策无关的

[1] 更多信息可以访问 RoboCup 官方网站：http://www.robocup.org/

底层细节。因此，研究开发人员可以用更多的精力去关注高层算法的研究与实现，比如多智能体在有噪声的实时环境中的任务规划、学习、合作等。可以使用不同编程语言实现自主球员程序（Client）。连续的观察、状态、动作空间是 RoboCup 仿真 2D 足球的一个特点。它可以看作是一个部分可观察的多智能体系统，其涉及到的决策问题有以下几个特点：

- 决策问题规模巨大：国际象棋问题的规模约为10^{20}；围棋问题的规模约为10^{200}；在仿真 2D 足球中，即使把连续的状态空间和动作空间粗略离散化后，智能体决策问题的规模也能达到约10^{400}以上。
- 大量不确定因素：比赛环境是部分可观察而且存在噪声；行动执行的结果具有不确定性；球员智能体之间的通讯是受限的；对手球员智能体模型是未知的。
- 实时系统：球员智能体需要在比赛的每个仿真周期内做出决策并执行，如果考虑到 Client 与 Server 端通信的网络延迟，则真正提供给球员智能体做决策的时间只有几十毫秒。

2.1.2 Soccer Server

RoboCup 足球仿真 2D 组的 Soccer Server 由 RoboCup 官方发布并持续更新，开发者可以从 sourceforge 网站下载当前最新版的 rcssserver-15.2.2。比赛过程中，Soccer Server 维持整个赛场上的客观世界状态，比赛规则基本与国际足球联合会[2]（Federation International of Football Association，FIFA）的比赛规则一致。

Soccer Server 发给 Client 的信息包括视觉、听觉和自身感知等信息。场上的每个球员智能体能够获得各自局部视野内的物体信息，如相对位置、速度等。这些信息中被 Soccer Server 添加了随机误差。听觉信息包括上个周期中队友 say 的信息、己方在线教练在 50 个周期之前 say 的信息以及裁判的指令，Soccer Server 规定球员之间通信的信息不能超过 10 个字节，这也说明 RoboCup 仿真 2D 足球比赛是一个通讯受限的环境。自身感知信息主要包括球员智能体自身的体力、速度等信息。球员 Client 智能体决策后，把要执行的动作信息发给 Soccer Server。

Soccer Server 的误差模型大致有两类：一类是发送给球员智能体的时候加入的用于表现观察的不确定性的误差，主要体现在视觉信息中；另一类是模拟球场过程和球员智能体执行动作时加入的误差。

[2] 国际足球联合会的官网是：http://www.fifa.com/

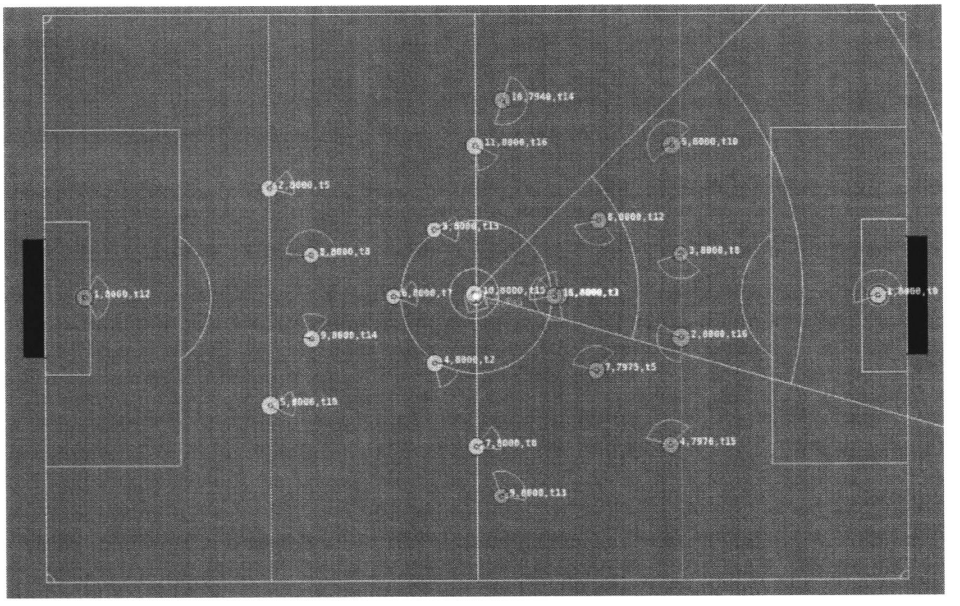


图 2.1 RoboCup 仿真 2D 比赛场景

Soccer Server 为球员智能体提供的原子动作主要有：

- kick power direction：即踢球。沿 direction 方向对球施加一个 g（power，direction）大小的冲量。
- dash power direction：即跑动。沿 direction 方向产生大小为 a（power，direction）的加速度。目前 Soccer Server 版本（rcssserver15.2.2）中，球员智能体可以向 8 个方向加速。
- turn angle：即转身 angle 角度。
- turn_neck angle：即球员智能体脖子转 angle 角度。
- tackle direction：即铲球。有一定概率成功，铲球成功时沿 direction 方向对球施加一个大小为 t（direction）的冲量，不论成功与否，球员智能体在铲球后的 10 个周期不能移动。
- say message：即智能体喊话给其他球员。
- change_view mode：即调整智能体的视野范围，有 60°、120°、180°三种，分别每周期更新、每两个周期更新、每三个周期更新一次球员智能体的视觉信息。
- catch direction： 即扑球。充当守门员角色的球员智能体可以向 direction 方向扑球。

其中，kick、dash、turn、tackle、catch 是互斥命令，球员智能体每周期只能向

Soccer Server 发送这些互斥原子命令中的一个。

仿真 2D 比赛的所有场景都可以通过官方发布的 Monitor 程序查看。开发者可以从 sourceforge 网站下载最新版的 rcsslogplayer 或者 rcssmonitor。这类软件直接与 Soccer Server 进行通信，可以使球队开发者更方便地观看比赛或者回访比赛录像，如图 2.1 所示。

2.1.3 Client

Client 是球员客户端，任何支持 UDP/IP 协议的编程语言都可以用来设计实现球队程序。比赛时，双方球队必须同时运行数量相等的 Client 端程序，因为每个 Client 端程序只允许控制一个场上球员。通常场上包括 11 个普通球员程序（Player Agent）和 1 个在线教练程序（Coach Agent）。Client 端程序之间的通信必须通过 Server 端转发，规则不允许 Client 端之间直接进行通信的行为。每场比赛开始前，双方各自的 12 个 Client 端球员程序分别与 Server 端建立连接。比赛中每个队的目标是扩大本方与对方的进球数的差值。每个仿真周期内，球员智能体接收 Soccer Server 发来的含有噪声的环境状态信息，并通过一些算法更新自身的世界状态。在完成相关任务决策后，智能体将计算得到的原子动作指令发给 Soccer Server，然后进入下一个决策周期。比赛按照这个过程不断地循环进行。

在设计实现时，球员 Player 和在线教练 Coach 都是 Client 的子类，两者有部分相同的数据成员和成员函数。这样的设计提高了代码的可重用性。

2.2 测试工具

2.2.1 场景再现工具 Trainer[3]

为了重复测试某些比赛场景，针对 RoboCup 仿真 2D 的特点，本文实现了一个 Trainer 程序作为模块化测试的工具。通常来讲，为了测试新的功能是否有助于改善之前糟糕的表现，我们需要通过一些测试来确保新的功能起了很好的效果。由于足球仿真 2D 组比赛环境的复杂性，无法有效进行单元测试；而且某些场景在比赛中较为罕见，大量测试工作非常耗时、效果不好。这时使用 Trainer 可以较好地解决这个问题。

Trainer 又称为离线教练（Offline Coach），可以很容易地重建一个所需要的场景。在由 Soccer Server 规定的环境中，Trainer 除了可以像在线教练那样喊话

[3] WrightEagle 队开源的底层代码 WrightEagleBase-4.0.0 及以后版本已经加入了 Trainer 功能，用于场景训练，可以从“蓝鹰”主页 http://www.wrighteagle.org/en/robocup/2D/index.php 下载

或者改变球员类型，还可以执行以下命令：

- 移动球员：将任意指定球员移动到指定位置，且使其具有一定的状态，这个状态包括他的速度和身体朝向；
- 移动球：将球移动到指定位置，并使其具有指定的速度；
- 恢复：恢复所有球员的体力（Stamina）和耐力能力（Stamina Capacity）；
- 改变比赛模式：改变当前的比赛模式到指定的模式，如：左方球队间接任意球模式（indirect_free_kick_r）或正常比赛模式（play_on）。

通过这些功能，某些被称作训练场景的特殊场景可以一定程度上地再现。通过设定要训练的场景所在的 rcg 文件（仿真 2D 组的比赛记录文件）、所在周期数和训练结束条件，Trainer 可以解析 rcg 文件并开始训练。

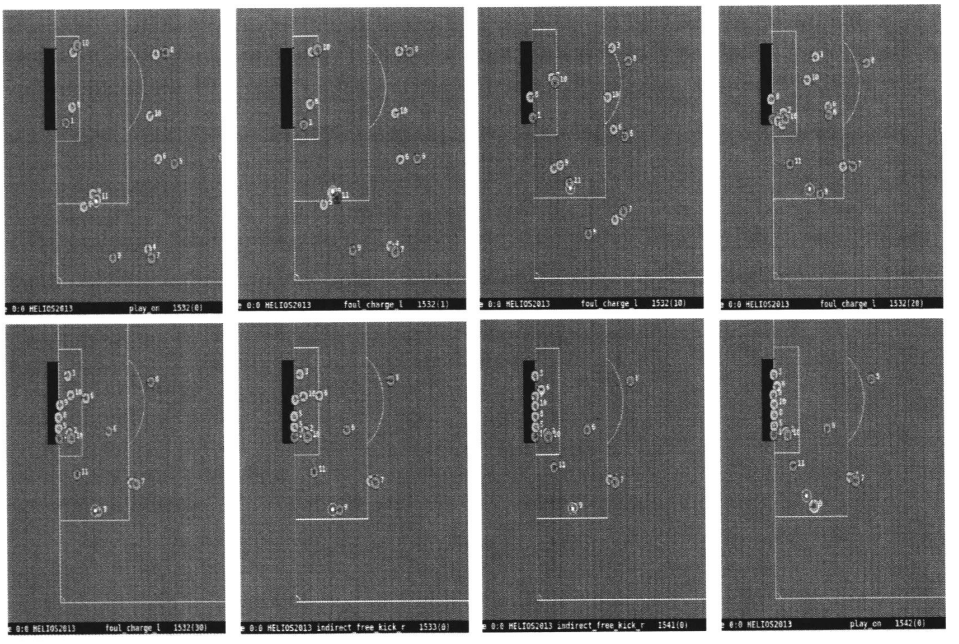


图 2.2 使用 Trainer 重现任意球等特殊比赛模式

如图 2.2 中，是某场比赛中，黄色 9 号球员在防守蓝色 11 号球员时犯规，第 1532(0)仿真周期，比赛是正常比赛（play_on）模式；从第 1532(1)至第 1532(30)仿真周期，比赛是左方球队犯规（foul_charge_l）模式；从第 1533(0)仿真周期至第 1541(0)仿真周期，比赛是右方球队间接任意球（indirect_free_kick_r）模式；从第 1542(0)仿真周期，比赛恢复到正常比赛（play_on）模式。设计和实现 Trainer 能够解析圆括号内的仿真周期数，使得 Trainer 可以用于重现 foul_charge 这种比赛模式，是后续工作的重要基础。

为了使每个球员智能体在训练场景开始时有一个精确的世界状态，Trainer 中有一个初始时间（Initial Time）。是指在训练开始前的这段时间，由 rcg 文件读入 Soccer Server 发送给球员智能体的信息，每个球员智能体据此更新自己的世界状态信息，使得训练中的球员智能体的决策依据和要测试的比赛场景中基本一致，从而可以更有针对性的对某些特殊模块进行有效的测试。这个过程类似于现实足球比赛中的球员“热身”运动，故初始时间也称为预热时间。

从技术实现方面来讲，Trainer 与球员一样使用 C++程序语言，也是 Client 的一个子类。其中，使用了一些球员 Player 简化版的数据结构，并增加了一些特有的数据结构，如：ConditionType（一次训练终止的条件）、Condition（条件）等。

2.2.2 rcsslogplayer 改进版

官方版本的 rcsslogplayer 是使用 Qt 程序语言设计实现的，可以在实时比赛中或通过解析 rcg 文件显示球场、球员等基本信息。其中，显示信息的对话框是使用 Qt Creator 设计编译生成[31]。为了方便测试，本文使用的 rcsslogplayer 程序是在官方版本的基础上修改过的，不仅可以解析正常的 rcg 文件，而且可以解析 rcg 文件中由 WrightEagle 球员程序在比赛过程中记录的一些信息。

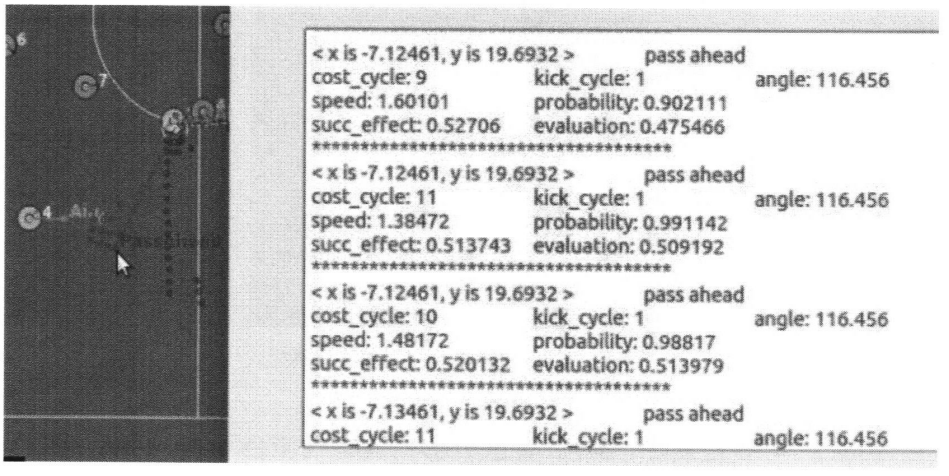


图 2.3 增强版的 rcsslogplayer 使用界面

如无特殊说明，本文中所有比赛示例图中：

- 黄色圆圈表示的球员均为 WrightEagle 队球员。
- 点、线、圆圈等信息均为比赛或者测试过程中，WrightEagle 球员程序自动记录到 rcg 文件中的信息由修改后的 rcsslogplayer 程序解析并显示出来的。

图 2.2 所示的是黄色 7 号球员进行 Pass 任务规划时，计算给黄色 4 号传球的一些目标点（红色点）。白色鼠标箭头所指的那个红色位置点是其中一个目标点，这个点部分信息如图中右边文字区域所示。其中 x 和 y 分别是指目标点在球场中的横坐标和纵坐标；cost_cycle 是指球从当前位置到目标点所需要的仿真周期数；kick_cycle 是指需要 1 个仿真周期的将球以 speed 的速度踢出；angle 是目标点相对于传球者的角度；probability 是指将球成功传到目标点的成功率；evaluation 是指对这个 Pass 行为的评估值。本文将在后续章节中对评估值进行详细介绍。

2.3 WrightEagle 的理论模型

2.3.1 马尔科夫决策模型

RoboCup 仿真 2D 足球比赛中，双方有截然相反的目标，即都希望多打进对方更多的球而自己尽可能少地丢球。而且 Soccer Server 加入的随机噪声给观察和行动带来一定的不确定性。如图 2.1 所示，以球员中心为顶点的白色扇形表示的是球员智能体的视野，图中最大的白色扇形是黄色 10 号球员的，由图看见此扇形分成三部分：黄色 10 号球员可以“看到”蓝色 6 号和 11 号所处的小扇形内所有球员的号码信息，可以“看到”蓝色 2 号、3 号和 5 号所处的扇环内所有球员的所属球队、但“看不到”号码信息，可以“看到”红色 1 号所处的扇环内所有球员、但“看不到”号码信息和球队归属，“看不到”视野之外的其他队友球员。由此可见 Soccer Server 对智能体获取视觉信息的能力进行了限制，导致智能体部分可观察的特性。所以仿真 2D 足球可以作为一个部分可观察的随机博弈问题来处理，进而可以使用有限阶段的 POSG 来对球队进行建模。

WrightEagle 队可以用一个九元组 $<\tau,S,\{b^0\},\{A_i\},\{O_i\},P,\{R_i\},K,\{B_k\}>$ 表示[32]，其中：

- τ 是场上 22 名球员智能体的索引。因为在线教练 Coach 和离线教练 Trainer 并不出现在球场上，所以 τ 的取值为 1，2,…，22。
- S 表示全体球员智能体所处环境的状态空间。这里采用因子化的方法表示状态。状态因子包括所有球员智能体和球的速度和位置，以及一些其他属性。
- $\{b^0\}$ 是比赛开始时的状态分布。通常在比赛正式开始之前，智能体仅仅会知道自己队友球员的信息。

- A_i是 Soccer Server 规定并实现了的原子动作集合。所有的球员智能体的原子动作集合都是一样的，除了双方各自充当门将角色的球员智能体多一个 catch 动作。根据 Soccer Server 的规则，有些情况下，球员智能体并不能执行某些原子动作。比如球在自己可踢范围以外时，球员就不能执行 kick 动作。
- $\{O_i\}$是 Soccer Server 发给所有智能体的视觉、听觉、智自身感知等信息。鉴于每个智能体各自所处的状态不尽相同，而且 Soccer Server 在其中随机加入了噪声，所以每个球员智能体获得的信息也基本不同。
- P是状态转移函数。根据球员智能体执行的动作，可以得到某个状态可能的后继状态和与之相对应的概率。
- $\{R_i\}: R_i$是索引为 i 的球员智能体的报酬函数。每个球员智能体所属球队不同、在队中所处的角色不同，所以不同的球员智能体有不同的立即报酬函数。
- K是行为生成器的索引集合。行为大致分为进攻和防守两大类。进攻行为包括 Shoot、Pass、Dribble 等对对方球门产生威胁的行为。防守行为包括 Block、Mark、Trap 等解除对方对我方球门的威胁的行为。
- $\{B_k\}: B_k$对应于第 k 个行为产生器。由某个球员智能体的索引值，经过行为产生器不仅可以得到当前状态下对应该行为的有效集合，而且还能得到各行为执行后的后继状态和对应的概率。

2.3.2 信息处理结构模型

上一节中提到了 Soccer Server 会向每个球员智能体发送：视觉、听觉、自身感知等信息。其中，球员智能体通过视觉信息可以获取视野内所有对象相对于自身的方向和距离等位置信息。球员智能体通过听觉信息可以获取球场上其他球员或者在线教练广播的不超过 10 个字节的字符串信息。球员智能体通过自身感知信息可以获取自身的速度、体力、胳膊指向等信息。如图 2.4 所示的每个方块即代表一个信息处理子模块，这些子模块把上述三种信息的字符串通过一定的处理，得到球场上当前的世界状态。

每个决策周期开始时，球员智能体的 Parser 将从 Soccer Server 发来的信息字符串接收、存储在 Observer 的数据结构中。而 Observer 对应 POSG 模型中的O，即智能体对环境状态的观察。这些观察大部分都是相对位置信息，如队友球员相对于智能体的位置、球相对于智能体自身的位置等。WorldState 将这些观察信息转化成状态信息，即由观察得到一个状态的概率分布。

为了使智能体更好地决策，由 WorldState 通过一系列计算得到 InfoState 和

Strategy。

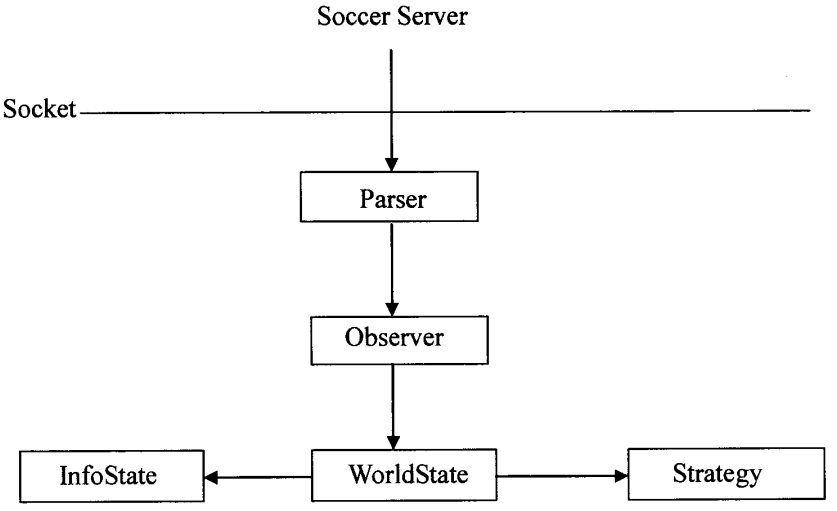


图 2.4 信息处理流程图

InfoState 是完全客观的信息状态，大致内容包括以下两点：

- PositionInfo：即位置信息，用于静态分析场上的对象。其中包括所有球员、球之间的相对角度、距离等信息，对方球队的阵型信息等。
- InterceptInfo：即拦截信息，用于动态分析场上的对象。计算出场上所有球员、球的拦截优先级信息，以便进一步分析场上的拦截攻防情况。

Strategy 是完全主观的策略信息，是 WrighEagle 特有的信息。如进攻时带球智能体在当前形势下的目标前进方向和最多可以前进的距离，防守时不同智能体分配到不同的防守任务等。

2.3.3 行为决策结构模型

WrightEagle 采用独立行为生成器，由状态和智能体索引得到相应的行为集合。为了充分利用决策时间，高层决策模块引入了 Anytime 思想，以保证规划出当前状态下最好的行为结果。而且球员智能体能够通过反算队友球员或者对手球员的决策，来提高与其他队友智能体之间协作行为的效率。

Anytime 算法[33][34]扩充了“输入－处理－输出”的传统求解过程，可以在计算过程中动态地输出结果。Anytime 可以在任何需要的时候被中断、返回特定质量的解。

如图 2.5 所示，矩形节点表示的是 WrightEagle 中不同的行为生成器。椭圆形节点为球场上实际状态节点，或中间过程中产生的虚拟状态节点。智能体从状

态 s_0 开始规划，经过所有的行为生成器的运算处理后，会产生如图 2.5 中的 $s_{1.1}$,$s_{1.2}$,$s_{1.3}$ 等后继状态节点。在这些状态的基础上，智能体继续进行规划，得到后续的行为，这样反复地迭代下去。

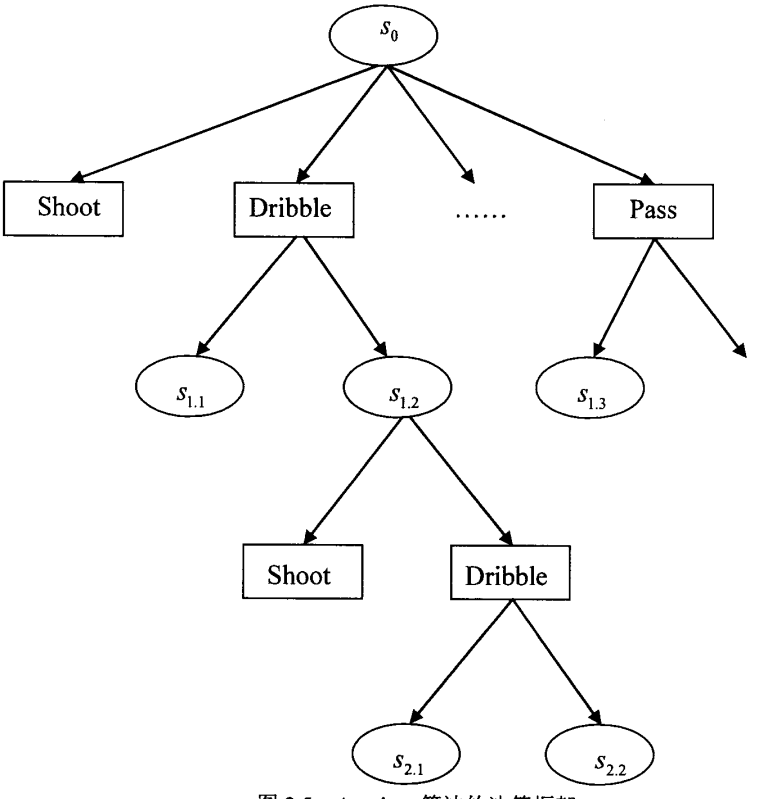
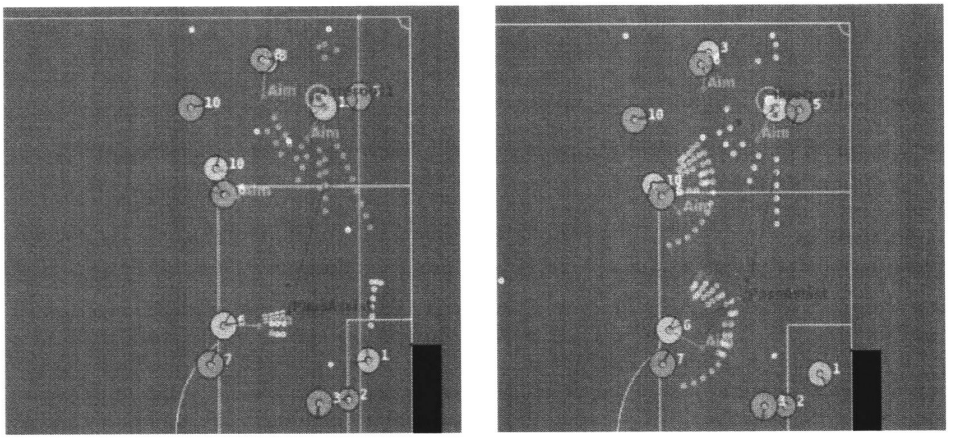


图 2.5 Anytime 算法的决策框架

正如在前面提到的，仿真 2D 足球中，留给球员智能体在线规划的时间不超过 100 毫秒，即一个仿真周期。因此必须尽快产生一个最优的行为解，然后还需要将其转化为原子动作指令发给 Soccer Server。行为生成器消耗的时间不仅与智能体所处的状态有关，而且和计算使用的硬件设备条件密切相关。采用这种可以随时中断、返回结果的计算方法，可以最大限度地利用宝贵的决策时间。基于 Anytime 的决策框架可以充分利用每个仿真周期内的剩余时间，并且使智能体在任何硬件条件，都能得到当前状态下的最优结果。

反算是多智能体决策时保证决策结果一致性的一个重要技术。在仿真 2D 足球中，对队友球员和对手球员行为的代入式计算都可以看作是反算。反算队友是为了多个队友球员之间能更好地完成协作任务。如正在执行 Position 决策的球员智能体通过反算带球的队友球员的 Pass 决策，规划出跑向带球者可能的传球区

域的行为。如果再结合球员之间的受限通信，可以一定程度上提高球员间传接球协作成功的可能性。相对的，反算对手是为了更好地预测对手球员的行为，如正在执行 Block 决策的球员智能体通过反算对方带球球员的 Dribble 决策、Pass 决策和 Shoot 决策，大致计算出对方最可能的带球路径，再据此提前做好封堵对方带球球员的准备，可以提高封堵成功的可能性。



(a)黄色 6 号球员反算出的结果　　(b)黄色 11 号球员计算出的结果

图 2.6　反算在 Position 决策中的应用

如图 2.6 所示，这是 RoboCup 仿真 2D 足球比赛中的截取的一个场景，展示的是反算在智能体的 Position 决策中的应用，通过分析这个场景来介绍反算在多智能体协作中的应用。图中蓝色圆形和红色圆形分别表示防守方的普通球员和守门员，黄色圆形代表进攻球员。每个球员的号码为显示在球员旁边的白色数字。在图中所示的仿真周期中，黄色 11 号球员正在对方角旗附近带球，距离黄色 11 号球员较近的队友黄色 3 号和黄色 10 号球员均已被蓝色防守球员牢牢盯防。如图 2.6(a)所示，黄色 6 号球员通过计算持球的队友球员很可能会将球传到图中的红点（旁边标有 PassAssist），所以黄色 6 号球员在 Position 规划的时候，选择可以保证自己第一时间能在红点附近接到球的行为并执行。同时如图 2.6(b)所示，在这个决策周期内，黄色 11 号球员也进行了 Pass 规划，其 Pass 规划用到的行为生成器和黄色 6 号球员进行 Position 规划反算其 Pass 行为时调用的是同一个行为生成器，由于每个智能体都维护了同一个信念池，虽然黄色 11 号和黄色 6 号的世界状态不是完全一样的，但是黄色 11 号球员 Pass 决策得到的传球点（图中的红色点，旁边标有 PassAssist）也在黄色 6 号球员反算出来的传球点附近。另外，关于信念池的概念在第四章将会详细介绍。

2.3.4 行为执行结构模型

上一节介绍的行为决策模块产生智能体的最优行为，需要通过行为执行模块将其转化成 Soccer Server 可以接收的原子动作。如图 2.6 中黄色 11 号的 Pass 和黄色 6 号的 Position，是不能直接被 Soccer Server 接收的，必须转化成若干周期的 kick、dash 等能被 Soccer Server 接收的原子动作指令，如：踢球（kick）、跑（dash）、铲球（tackle）、转身（turn）、转脖子（turn_neck）等。为了将行为生成器产生的不同行为转化为上述这些原子动作，WrightEagle 中设计实现了一些行为执行子模块。例如：

- Kicker：该子模块由球员智能体索引和踢球速度，规划产生若干周期的 kick 动作序列。
- Dasher：该子模块由球员智能体索引和目标点坐标，规划产生若干周期的 turn 和 dash 动作序列。

当球员智能体将原子动作指令发给 Soccer Server 后，当前决策周期就结束了。等到 Soccer Server 发来新的信息，进入下一个新的决策周期。如图 2.7 所示是行为执行的流程图。

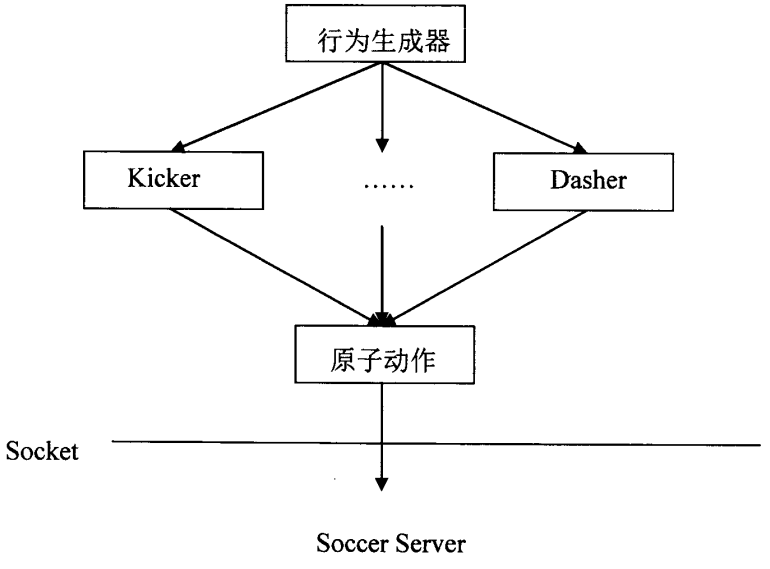


图 2.7 行为执行模块结构图

2.4 本章小结

本章介绍了本文工作的主要基础，即 WrightEagle 仿真 2D 球队决策系统和

Trainer 等场景重现工具。首先，介绍了机器人足球世界杯，以及其中的足球仿真 2D 组比赛平台的特点；其次，介绍了本文最重要的测试工具，也是本文重要基础工作：后续章节进行测试实验时所用到的场景重现工具 Trainer 和改进版本的 rcsslogplayer；然后，介绍了 WrightEagle 的抽象模型，以及信息处理、行为决策、行为执行三个模块。本文后面介绍的工作都是在此基础上实现的。

第三章 马尔科夫决策过程的分层分解

马尔科夫决策用状态转移来描述客观世界的动态过程。求解 MDP 问题的算法，可以按照是否求解问题的整个状态空间进行划分。本章剩余内容主要安排如下：第 3.1 节按照离线和在线两种方式，分别介绍了 MDP 问题的求解算法；第 3.2 节引入了马尔科夫决策过程的 MAXQ 分层分解技术；第 3.3 节详细介绍了基于分层分解技术对 WrightEagle 球队建模与分析；第 3.4 节是本文工作的第二个部分，介绍了 WrightEagle 球队的评估系统，包含对第 3.3 节中介绍的多步子任务的评估；第 3.5 节应用分层分解方法与第二章提到的反算技术，以守门员的站位问题为例，解决一类需要对手建模的智能体决策问题；第 3.6 节给出了本章小结。

3.1 马尔可夫决策过程求解算法

3.1.1 离线求解算法

离线算法在求解过程中，通过遍历 MDP 问题的整个状态空间，计算出完整策略，并在智能体与环境交互前传给智能体，使其在与环境交互时执行此策略。策略迭代[35]（Policy Iteration）算法和值迭代[36]（Value Iteration）算法是求解 MDP 问题的两个经典离线规划方法，均利用了动态规划的相关思想。

在策略迭代中，根据显式表示的策略能够计算得到对应的值函数V^{π}，可以使用使用以下公式改进策略：

$$Q^{\pi}(s,a)=R(s,a)+\gamma\sum_{s'\in S}T^{a}(s,s')V^{\pi}(s') \tag{3.1}$$

$$\pi'(s)=\arg\max_{a\in A}Q^{\pi}(s,a) \tag{3.2}$$

其中，γ是折扣因子（Discount Factor），$\gamma\leq 1$。因为可能的策略数目是有限的，而且策略迭代过程中一直在改进当前策略，所以算法经过有限步迭代之后会收敛于最优策略。如算法 3.1 所示。

在一般情况下，策略迭代要实现收敛仅仅需要较少的迭代步骤即可完成。每次迭代时，有额外的时间是用来等待“评估策略”步骤收敛——这是策略迭代算法最明显的不足。最直观的改进方法是：把“评估策略”分散到“改进策略”中，就有可能更快收敛。这也是值迭代算法的主要思想。

1. 初始策略π
2. 评估策略：

   通过 $V^{\pi}(s)=R(s,\pi(s))+\gamma\sum_{s'\in S}T^{\pi(s)}(s,s')V^{\pi}(s')$ 计算对于策略π的值函数V^{π}。
3. 改进策略：

   应用公式 3.1 和公式 3.2，优先考虑当前选择的行动。
4. 检查受否收敛：

   如果π与π'一致，则返回一个最优策略π；否则，设置π为π'，然后转到评估策略那一步。

算法 3.1 策略迭代算法

在值迭代中，没有显式表示的策略，改进值函数的方法是由 Bellman 公式不断进行迭代。在每个迭代周期内，所有状态按照公式 3.3 进行备份操作：

$$V_{t+1}(s)=\max_{a\in A}\{R(s,a)+\gamma\sum_{s'\in S}T^{a}(s,s')V_{t}(s')\} \tag{3.3}$$

不区分V_t和V_{t+1}，在原地进行备份操作的方法称作异步动态规划。这样可以一定程度上改进值迭代算法。异步迭代可以用不同的顺序来进行备份操作，这样能对还没有收敛的状态空间区域优先进行更新。如算法 3.2 所示：

1. 初始值函数V和ε
2. 提升值函数：

   应用公式 3.3，改进值函数。
3. 检查受否收敛：

   如果 Bellman 误差小于等于ε，则通过公式 3.2 返回一个ε最优策略π；

   否则，设置V为V'，然后转到提升值函数那一步。

算法 3.2 值迭代算法

其中，每次迭代时，更新前与更新后的所有状态值函数最大差值就是 Bellman 误差 r，也即是 $r=\max_{s\in S}|V_t(s)-V_{t+1}(s)|$。给定 Bellman 误差，一个最优值函数$V^{*}$的上界$V^{U}$和下界$V^{L}$可以按照下列公式计算得到：

$$\begin{aligned}V^{U}&=V^{\pi}(s)-\phi^{\pi}(s)r\\V^{L}&=V^{\pi}(s)+\phi^{\pi}(s)r\end{aligned} \tag{3.4}$$

其中 $\phi^{\pi}(s)=1+\sum_{s'\in S}T(s,\pi(s),s')\phi^{\pi}(s')$ 被称为平均初过时间，其含义是指从状态 s 开始，按照策略 π 执行相应的行为，到达任意一个目标状态的期望时间。

对于策略迭代，当前状态的值函数是最优值函数的一个下界。当上界与下界的差值 $\max_{s\in S}[V^{U}(s)-V^{L}(s)]$ 小于 ε 时，称策略为 ε 最优策略。对于给定的任意实数 $\varepsilon>0$，值迭代算法和策略迭代算法在经过有限步的迭代之后，都会收敛于 ε 最优。

3.1.2 在线求解算法

在处理像 RoboCup 仿真 2D 足球这样的大规模 MDP 问题时，基于动态规划的离线规划算法往往并不适用。因为离线求解算法会预先计算出完整的策略，就需要搜索遍历整个状态空间。但是随着状态空间维度的增加，状态空间的大小呈指数增加，这就是“维度诅咒”问题。同时，很多实际问题中的状态转移函数和报酬函数等会发生变化，导致预先计算的策略在执行时已经不再适用。为了克服“维度诅咒”问题，同时满足实时性的需求，在线求解算法的思路是：只为当前状态计算出一个最优行动策略，而不需要为整个状态空间计算出完整的策略。与或树搜索（AND/OR Tree）和实时动态规划（Real-Time Dynamic Programming，RTDP）是经典的在线规划算法。

与或树又称搜索树，是指智能体从当前状态出发，递归地评估所有可以执行的动作和可能遇到的状态的过程中构建的一个以当前状态为根节点的树形结构。

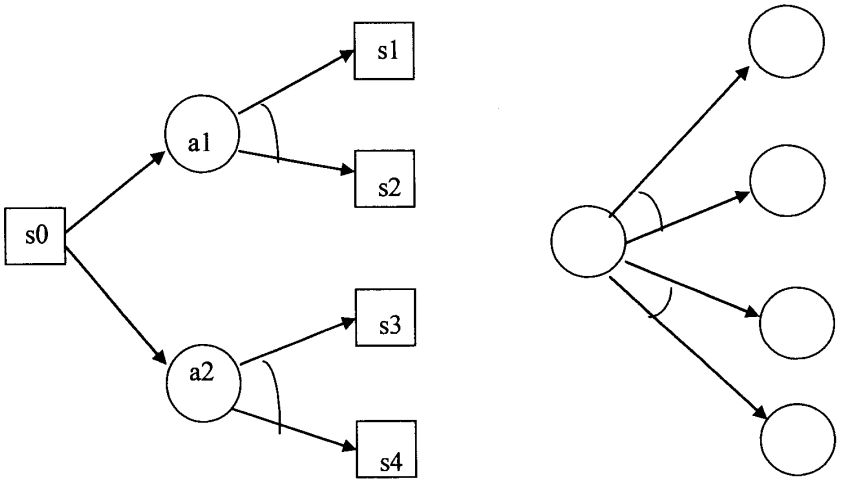


图 3.1 与或树基本结构示例图

如图 3.1 所示，其中的方形代表或节点，又称选择节点，是状态节点；圆形代表与节点，又称或然节点，是动作节点。s0 是智能体所处的当前状态，s0 有两个可执行的动作 a1 和 a2。

AO*和 LAO*是较早的基于与或树的搜索类算法，和人工智能中的 A*算法的设计思路类似，还有迭代加深的 IDA*算法基础上扩展出来的 LDFS 算法[37]。

与或图给了我们一种更好地理解马尔科夫决策过程求解过程的的基本数据结构。而这种结构在这类利用前向搜索和状态可达性的技术中都显式或隐式地用到。实时动态规划算法（RTDP）是一种基于前向搜索的技术。为了给当前状态找到最优行为策略，智能体迭代地执行试验搜索。

Trial-based RTDP 算法由一系列的试验（Trials）组成，每次试验由若干步组成。智能体采用一步前瞻搜索的方法选择每一步将要执行的行动 a，然后根据 a 的所有可能结果来更新当前状态。当达到某个目标状态或者是经过一个指定步数的更新时，试验终止。如算法 3.3 所示：

1. 以可允许的值函数开始，
2. 重复以下试验过程 n 次：
3. 试验：把当前的状态 s 设置为起始状态，重复以下的步骤 m 次或者到达一个目标状态
   （a） 根据公式 3.3，对当前状态 s 的进行备份操作来提高值函数；
   （b） 执行得到的最优动作动作并把当前的状态 s 设置为通过这个动作随机转移到的状态；
4. 根据公式 3.2，从值函数 V 中得到一个局部策略。

算法 3.3 Trial-based RTDP 算法

这种基于试验的 RTDP 算法仅仅更新那些从初始状态出发、执行基于当前值函数采用贪心策略选择得到的行动可以到达的状态，所以可以免去大量无关状态空间上的计算。Barto 证明，在资源允许的情况下，RTDP 能够渐近收敛于最优解，而不需要评估完整的状态空间，并将结果与启发式搜索关联，认为它是实时启发式搜索算法[38]（LRTA*）的一个推广。

3.2 马尔可夫决策过程分层分解

3.2.1 MAXQ 分层分解

在不确定性环境下，MDP 作为基本的理论模型，为解决单个智能体的规划问题提供了一种途径。使用 MDP 对此类问题建模的方法被应用于智能体决策相关问题求解的各个领域。一些离线计算方法，如值迭代算法、策略迭代算法、线性规划算法等，是现阶段常见的精确求解 MDP 问题的算法。遗憾的是，精确求解 MDP 问题的这些算法应用到大规模问题时，都会受制于“维度诅咒”问题，无法直接应用。“维度诅咒”是指状态空间的大小随着环境状态变量数目的增加呈指数级增加[39]。以 RoboCup 仿真 2D 足球为例，其问题的状态空间的维度是 136，这还是在只考虑球员智能体的位置坐标、速度、身体角度等主要的状态变量的情况下的数目。即使仅仅把每个连续的状态变量离散成其取值范围内的一千个值，计算得到的状态空间大小依然超过10^{408}。状态空间是如此的巨大，以至于不可能通过离线算法求解出完整策略。况且 RoboCup 中对手球员的策略会经常变化，通过离线算法求解出的策略无法有效灵活地应对这些变化。

由于在与环境交互的过程中，智能体仅仅会遇到有限的状态，不需要处理所有的状态空间。所以在线规划算法只计算出相应的局部策略，计算过程只在当前状态可以到达的状态空间内。比如在 RoboCup 仿真 2D 中，一场比赛一般包含 6000 个仿真周期，每个仿真周期是 100 毫秒。智能体只需要在这段时间内对所处的状态做出决策，状态数目远远小于完整的状态空间大小。在线规划算法递归地进行前向搜索：在当前状态下通过评估所有可行的动作集合，返回集合中的最优动作。结合各种启发式方法可以减少计算时间和空间资源的消耗，如 LAO*算法[40]、DNG-MCTS 算法[41]、UCT 算法[42]等。在线规划算法在当前状态下实时地做决策，使得其可以比离线规划算法更好地处理环境模型不确定的变化。但是，实时性是在线规划算法的关键。比如在 2D 足球中，每周期内属于球员智能体的决策时间不超过 100 毫秒，必须在这么短的时间内选出最优行动。

另一种可以把 MDP 算法扩展到大规模问题的方法是分层分解技术。分层分解技术把原问题按照分层结构分解成一系列更容易处理的子问题[43]。WrightEagle 使用 MAXQ 值函数分解策略来进行分层分解，这样原 MDP 问题的值函数被分解成一棵树状结构之上的若干个子问题的值函数之和[44]。状态抽象、时序抽象和子任务共享是 MAXQ 分层分解的理论基础。状态抽象是指每个子任务只需要关注跟当前子任务目标有关的状态变量。分层结构中的高层任务把低层任务视作宏动作，在高层 MDP 中像原子动作一样处理，这是时序抽象。子

任务共享是指一个低层子任务的策略会在不同的规划中多次重复利用。比如在本章末尾的守门员站位决策实验中，反算用到了对方前锋球员的 Pass 子任务。

在线规划方法和分层分解使得 MDP 可以扩展应用到大规模问题中。常见的分层分解技术包括 Option 理论[45]，层次抽象机[46]（Hierarchies of Abstract machines）和 MAXQ 分层分解[47]（MAXQ Hierarchical Decomposition）等可以把 MDP 问题分解成一系列更容易解决的子问题。

MAXQ 分层分解基本思想是：在任务树上，把原 MDP 问题 M 分解成多个子任务。每个子任务都是一个 MDP 问题，表示为 $\{M_0, M_1, \cdots, M_n\}$。其中，$M_0$ 是根任务。这些子任务对应的宏动作是任务树节点对应的低层子任务。求解原 MDP 问题 M 就转换为求解 M_0 对应的 MDP。分层策略 $\pi = \{\pi_0, \pi_1, \cdots, \pi_n\}$，其中，$\pi_i$: $S_i \to A_i$ 是子任务 M_i 的子策略。智能体在状态 s 下执行策略 π 直到子任务 M_i 在某个终止状态 $g \in G_i$ 终止，其获得的期望累积报酬也被称为投影值函数 $V^\pi(i,s)$。而行动值函数 $Q^\pi(i,s,a)$ 则是执行宏动作 M_a 后按照策略 π 直到子任务 M_i 终止，在这个过程中智能体获得的期望累积报酬。

分层策略 π 的值函数可以如公式 3.5 表示：

$$Q^\pi(i,s,a) = V^\pi(a,s) + C^\pi(i,s,a) \tag{3.5}$$

$$V^\pi(i,s) = \begin{cases} R(s,i), & \text{如果} M_i \text{是原子任务} \\ Q^\pi(i,s,\pi(s)), & \text{其他情况} \end{cases} \tag{3.6}$$

其中，

$$C^\pi(i,s,a) = \sum_{s',N} \gamma^N \Pr(s', N \mid s, a) V^\pi(i, s') \tag{3.7}$$

是执行宏动作 M_a 终止后，子任务 M_i 终止前，按照策略 π 的智能体获得的期望累积报酬。其中，$\Pr(s', N \mid s, a)$ 是状态 s 下执行宏动作 M_a，最后在状态 s' 终止的概率。

3.2.2 基于 MAXQ 分层分解的在线规划算法

基于 MAXQ 分层分解的在线规划算法 MAXQ-OP 在大规模在线规划算法框架里引入 MAXQ 分层分解[48]，为复杂环境中智能体的规划提供了更高效的解决方案。分层分解的应用可以更高效地在线搜索到搜索树上更深的节点。这样，根节点能获得更精确的值函数估计，可以在动态规划框架内选出更好的动作并执行。

MAXQ-OP 算法 OnlinePlanning 函数算法主要流程如下：

1. $r \leftarrow 0$
2. $s \leftarrow GetInitState()$
3. while $s \notin G_0$ do
4. $< v, a_p > \leftarrow EvaluateState(0, s, [0, 0, \cdots, 0])$
5. $r \leftarrow r + ExecuteAction(a_p, s)$
6. $s \leftarrow GetNextState()$
7. end while
8. return r

算法 3.4 OnlinePlanning 流程表

其中，由函数 GetInitState（）对状态 s 进行初始化。函数 ExecuteAction（）通过深度优先搜索的方式在决策树上进行搜索，得到当前状态应该执行的最优动作并执行。GetNextState（）函数获得环境的下一个状态。这个过程一直循环，直到环境到达某一个目标状态 $s_g \in G_0$。

3.3 WrightEagle 球队建模

WrightEagle 将 RoboCup 仿真 2D 足球建模成一个 MDP 问题[49][50]，相关定义如下：

- 状态集合：$S = \{s_0, s_1, s_2, \cdots, s_{22}\}$，包括智能体自己 $s_v, v \in [1,11]$ 在内，还有其他十个队友球员 $\{\{s_1, s_2, \cdots, s_{11}\} - s_v\}$，十一个对手球员 $\{s_{12}, s_{13}, \cdots, s_{22}\}$ 和球的状态 $s_0 = \{x, y, m, n\}$，其中，$(x, y), (m, n)$ 分别为位置和速度。对于智能体来说，$s_i = \{x, y, m, n, \alpha, \beta\}, i \in [1, 22]$，其中 α, β 分别是球员智能体的身体朝向的角度和脖子的角度。
- 动作集合 $A = \{dash, kick, tackle, turn, turn\_neck\}$，这些原子动作由 Soccer Server 定义并实现，且都带有连续型参数，使得动作空间也是连续的，这里采用因子化方法表示状态对其进行离散。
- 状态转移函数 T：Soccer Server 定义并实现了原子动作的相关模型。这里把其他球员作为环境的一部分，并假设他们的先验行为模型为：如果处于我方持球进攻状态，队友球员会优先规划与持球者的传接球配合，距持球者较近的对方球员会上前围抢；如果处于对方持球进攻状态，队友球员在防守跑位时会优先考虑临近队友球员的联合防守。通过这种方式，就把多智能体环境建模成单个智能体的马尔科夫决策过程模型。

- 报酬函数 R：在分层规划中，为每一层的子任务引入伪报酬函数和启发式函数。

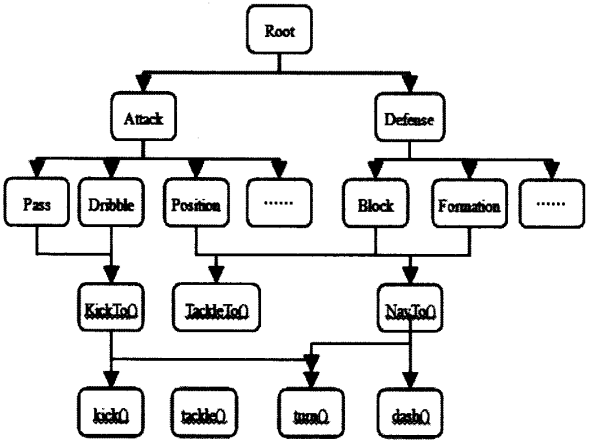


图 3.2 WrightEagle 队的 MAXQ 分层分解任务图

如图 3.2 所示，MAXQ 分层结构的基本组成 (子任务名称后面的圆括号表示这个子任务是带有参数的；图中省略号表示有未列出的子任务)：

Root：第一层是根任务。Root 会优先评估 Attack 子任务，如果无解，则继续规划 Defense 任务。

Attack，Defense：第二层子任务。Attack 和 Defense 的目标分别是主动进攻并取得进球和积极防守以阻止对方进球。两者的吸收状态分别是球被对方球员截到和球被队友球员（包括智能体球员自己）截到。这里的吸收概念是指这样一种状态 s，无论智能体执行何种行动 a，都会以概率 1 转移到状态 s。

Pass，Dribble，Position，Block，Formation 等：第三层子任务是球队的高层行为。其中，Pass 的目标是把球传给合适的队友；Dribble 的目标是带球朝某一方向移动；Position 的目标是使球队整体保持合适的进攻阵型；Block 的目标是封堵对方持球球员的进攻路线；Formation 的目标是是球队整体保持合适的防守阵型。Pass 和 Dribble 规划的前提是球对于智能体来说可踢或可铲，两者的吸收状态是球对于智能体来说不再可踢或可铲。Position 的吸收状态是对方截到球或者球对于智能体可踢。

KickTo()，TackleTo()，NavTo()：第四层子任务。其中 KickTo()和 TackleTo()的目标分是把球以某速度踢向给定的方向和铲到给定的方向，二者的吸收状态是球对于智能体来说不再可踢或者可铲；NavTo()的目标是把智能体移动到某个给

定的目标点，其吸收状态是智能体已经处在目标点的位置上。

kick()，turn()，dash()，tackle()：第五层子任务。这些都是 Soccer Server 已经定义、可以接收的原子动作，每个原子动作执行后会得到立即报酬-1，通过这种方式，使最优策略可以以最快的方式被执行。

以任务 Pass 为例，令 s 表示当前的联合状态，则有：

$$V^*(Attack,s)=\max\{Q^*(Attack,s,Pass),\cdots,\\ Q^*(Attack,s,Dribble),Q^*(Attack,s,Position)\} \tag{3.8}$$

$$Q^*(Root,s,Attack)=V^*(Attack,s)+\\ \sum_{s'}\Pr(s'\mid s,Attack)V^*(Root,s') \tag{3.9}$$

$$V^*(Root,s)=\max\{Q^*(Root,s,Attack),\\ Q^*(Root,s,Defense)\} \tag{3.10}$$

$$Q^*(Attack,s,Pass)=V^*(Pass,s)+\\ \sum_{s'}\Pr(s'\mid s,Pass)V^*(Attack,s') \tag{3.11}$$

$$V^*(Pass,s)=\max_{p}Q^*(Pass,s,KickTo(p)) \tag{3.12}$$

$$Q^*(Pass,s,KickTo(p))=V^*(KickTo(p),s)+\\ \sum_{s'}\Pr(s'\mid s,KickTo(p))V^*(Pass,s') \tag{3.13}$$

$$V^*(KickTo(p),s)=\max_{\alpha,\theta}Q^*(KickTo(p),s,kick(\alpha,\theta)) \tag{3.14}$$

$$Q^*(Kickto(p),s,kick(\alpha,\theta))=R(s,kick(\alpha,\theta))+\\ \sum_{s'}\Pr(s'\mid s,kick(\alpha,\theta))V^*(KickTo(p),s') \tag{3.15}$$

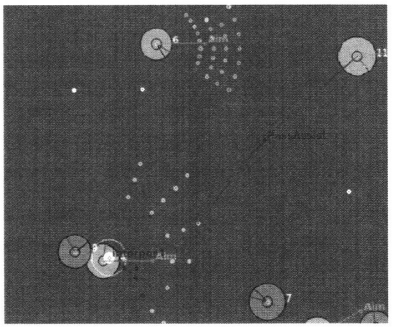
其中，$\Pr(s'\mid s,kick(\alpha,\theta))$ 由 Soccer Server 的物理模型给定。

子任务 Pass 在球对于智能体不再可踢时终止，然后返回 Attack 任务，由其进一步规划智能体是否应该 Position 以获得更好的位置等。

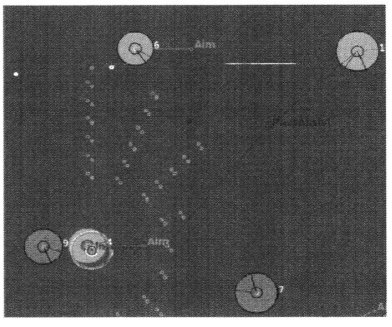
这里，针对持球者被对方防守球员压制到边线附近的情况，在第三层增加了 Dribble-Pass 子任务（与图 3.2 中的 Pass，Dribble 在同一层，同属于 Attack 的子任务），即在当前状态下，持球者无法规划出单一的传球或者带球策略，这时考虑先带球到一个位置再把球传出去，与此同时，队友球员在持球者带球的时候规划 Position 子任务，接应持球者。与 Pass、Shoot 等子任务不同的是，Dribble-Pass 是一个两步子任务，所谓的两步是指先进行 Dribble 子任务规划得到中间结果状态集合，再在中间结果状态集合里尝试进行 Pass 子任务规划，以期得到队友顺

利接到球的最终结果状态。这个过程虽然涉及到两步规划，但是作为一个整体进行子任务规划。

如图 3.3 所示，黄色 4 号球员计算出自己应该带球到某个黄色的位置点、再将球传向标有“AssistPass”的红色位置点，协同配合的工作由黄色 11 号球员完成。两张图分别显示的是黄色 4 号和 11 号球员各自的计算结果，如果忽略由于环境误差带来的计算偏差，结果是很接近的。



(a) 4 号球员规划出来的 Dribble-Pass 策略



(b)11 号球员计算出 4 号球员的 Dribble-Pass 策略

图 3.3 球员间主动的传接球协作行为

在非完全可观察的状态下，如果在 MDP 模型中引入信念状态[51]，则可以描述智能体在线决策的问题。这类问题也被称作是部分可观察马尔科夫决策过程。信念状态 b 是状态空间上的概率分布。$b(s)$ 表示当前环境状态为 S 的概率。假设环境中所有的智能体都是条件独立的，那么 $b(s)$ 可以展开表示为：$b(s)=\prod_{i\in I} b_i(s[i])$，其中 S 是完整的状态向量，$s[i]$ 是状态向量中对象 i 的状态分量，$b_i(s[i])$ 是对象 $s[i]$ 的边缘分布。每个智能体的状态样本在每个周期内通过 Soccer Server 的感知模型和运动模型进行蒙特卡洛更新[52][53]。

3.4 评估系统

3.4.1 启发式评估系统

正如本文在 3.2 节提到的，我们希望球员智能体的行为规划算法尽快收敛到 $Q^{\pi}(i,s,a)$ 的最优解。这里，可以使用启发式方法选择其扩展方向[54]。之所以考虑结合启发式方法，是因为RoboCup仿真2D足球中的报酬函数值具有稀疏特性。

从足球比赛的角度来看，如果球队取得了进球，则球队将获得一个很高的报酬值；反之，报酬值为 0。由于报酬函数值的稀疏特性，迭代式的前向搜索经常因为获得报酬为 0 而中断。这时，如果在评估系统中引入启发式方法，将会帮助我们从所有可能状态中发现那些最有可能产生进球的状态。这样既可以加速策略树中的搜索过程，又能抓住比赛中稍纵即逝的有利机会。

对于长周期的搜索过程，启发式评估系统的输入是球员智能体所处的状态，输出是从当前状态出发到达进球状态的可能性评估值。结合 MAXQ 分层分解和启发式的报酬函数，可以得到如下一个评估系统：

$$Q(i,s,a)=V(a,s)+C(i,s,a) \tag{3.16}$$

$$V(i,s)=\begin{cases} Q(i,s,a), & \text{如果 i 是复合的} \\ R(s,i), & \text{其他情况} \end{cases} \tag{3.17}$$

其中，

$$C(i,s,a)=\sum_{s',N}\gamma^{N}\Pr(s',N\mid s,a)V(i,s') \tag{3.18}$$

$$R(s,i)=\begin{cases} H(s,i), & \text{如果 s 是一个终止状态} \\ 0, & \text{其他情况} \end{cases} \tag{3.19}$$

当搜索过程从初始状态开始时，评估系统对初始状态进行评估并得到启发式的报酬值 H(s)。然后算法从当前这个状态扩展到其后继状态。一旦发现有新的状态，算法就会把前驱状态的启发式报酬用当前实际的报酬值代替。而这个报酬值在大部分时间都是等于 0 的。评估系统会给新的状态计算出新的启发式报酬值 H(s') 。

在搜索过程中，每一步的终止状态会继续扩展成一个新的内部中间状态。这样，评估方法不只取决于当前的状态，还跟这个状态相关联的前驱、后继状态序列有关。

3.4.2 状态可达性检查方法

在 RoboCup 仿真 2D 足球中，根据第 3.2.1 节的公式 3.7 可知：大多数状态之间的转移概率 $\Pr(s', N \mid s, a)$ 都是 0。所以，在产生决策的过程中只需要扩展一小部分后继状态。如果能加速识别不可到达的状态，则在固定的决策时间内可以加深搜索的深度，提高解的质量。

根据公式 3.16 中的转移函数，可以把状态可达性问题简化成：一个球员智能体从一个初始位置经过若干周期能否到达某个指定位置的问题。

为了保证状态检查的速度，采取简化版的模型适当降低对精度的要求：

- 简化 Soccer Server 中球员的运动模型：在简化版的运动模型中，仅考虑球员智能体 dash，不考虑 turn 和其他动作；
- 简化球员智能体的状态：仅仅考虑距离，不考虑身体朝向等其他信息；
- 把连续的球场空间通过格子的形式离散化。

事实上，如果计算球员智能体从一个初始位置到另一个位置的最小周期，采用简化版得到的结果总是采用完整模型计算得到的结果的下界。这意味着如果一个状态在简化版模型中是不可达的，那么在完整模型中肯定也是不可达的。但是反过来，如果一个状态在简化版本中是可达的，那么在完整模型中可能是可达的，有可能是不可达的。这种情况需要在完整模型中进行精确计算。

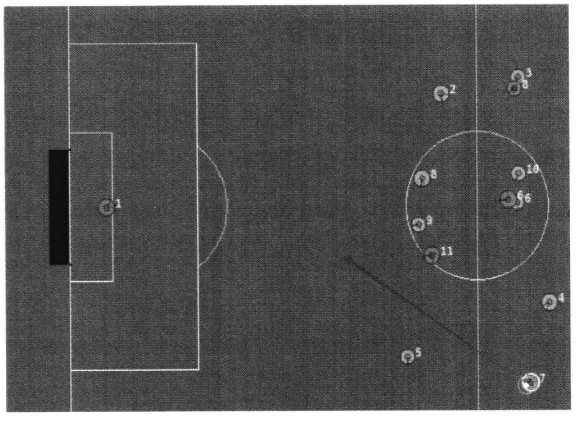
3.5 实验及结果分析

在 RoboCup 仿真 2D 中，有一类需要对未知对手建模的智能体决策问题：当球还在对方半场时，WrightEagle 对的守门员智能体一直采取站在靠近球门线的位置，只在对方进攻球员到我方禁区附近时才做出相应的移动行为。这样做最大的好处是降低了守门员的体力消耗。不利的是，有些球队进攻的时候利用了 WrightEagle 守门员与后卫球员之间的空档。如图 3.4(a)所示[4]，对方蓝色 7 号球员正在带球进攻，我方绿色 1 号球员站在本方小禁区的前沿。很明显，在 1 号守门员与由 2 号、8 号、9 号、5 号防守球员组成的后卫线之间，存在很大的防守空档区域。蓝色带球者会选择将球按照图中红色箭头所示的路径传给蓝色 11 号球员（也称为 ThroughPass，详见第 4.4 节相关介绍），造成单刀球机会，对防守方造成很大威胁。所以，对守门员的站位决策也要考虑其他球员的行为。这里的其他球员包括本方的防守球员、对方的带球者以及有可能的接球者。

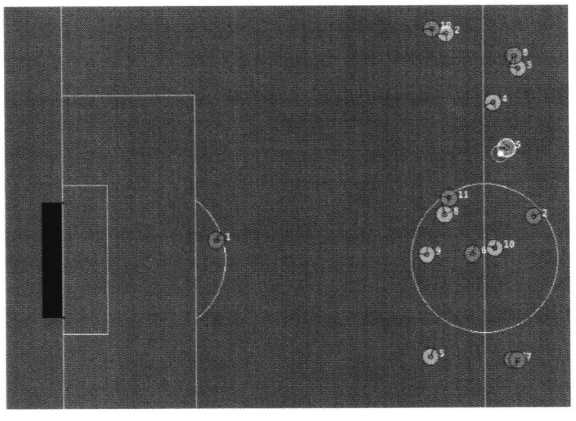
一种可行的方法是采用第 2.3 节介绍的反算技术与本章提到的 MDP 的分层

[4] 图 3.4 中的红色箭头是使用截图工具截取比赛场景后，添加上去的，不是球员记录的信息。

分解技术相结合。具体做法是：当球距离球门还较远的时候，守门员在进行决策的过程中，反算后卫的 Mark、Formation 等子任务决策以及对手前锋的 Pass、Position 等子任务决策，使用队友 Pass 决策的规划模块处理对手的状态信息（即把自己放在其他智能体的位置上去“思考”），利用上一节引入的评价系统，估算出能最大程度降低对方传球推进速度的行为并执行。



(a)守门员修改前的站位



(b)守门员修改后的站位

图 3.4 守门员站位对对方球员间传球突破行为的影响

基于前面章节中对 WrightEagle 的建模，本节在 WrightEagle 仿真 2D 足球队中实现了相关算法。如图 3.4(b) 所示，守门员的站位是前述方法计算得到的结果。如果绿色 1 号守门员还按照以前的站位方式，蓝色 5 号球员会采用如红色箭头所示的路径传球给蓝色 11 号球员。守门员的站位修改后，对方在这个场景中不会再将球传到红色箭头所示的靠近大禁区线附近。利用第 2.2 节实现的训练器 Trainer，反复重现并测试了对方利用防守空档实施突破的场景。测试结果参见表 4.5 和表 4.6。测试实验使用的硬件环境是：Intel Core i7 950，6GB Memory，软件环境是 Ubuntu 14.04 64 位版，rcssserver15.2.2 。其中：

- Goalie1：是 WrightEagle 队守门员原有的靠近己方球门的站位；
- Goalie2：通过统计的方法，进行数据分析，使守门员的站位更关注：
  (1)己方防守球员之间的空档，如在图 3.4(a)中是黄色 9 号和黄色 5 号之间，图 3.4(b)中是黄色 2 号和 8 号之间有明显空档；
  (2)对方最有威胁的球员，在图 3.4 的两个场景中，均为蓝色 11 号球员；
- Goalie3：结合了 MDP 分层分解技术的反算方法，利用了 Pass、Position 等子任务的重用。

为了排除平台环境的不确定性对测试结果的影响，测试守门员站位对方 ThroughPass 行为的影响的实验中，对每个算法分别测试包括图 3.4(a)和 3.4(b)在内的 10 个场景共 1500 轮测试。测试结果如表 3.5 所示。其中，对方 ThroughPass 数是指对方球员在 1500 轮测试中通过 ThroughPass 行为把球传出的的次数。对方 ThroughPass 成功率是指在对方球员接到 ThroughPass 传出的球，并形成单刀球的次数与 ThroughPass 行为次数的比例。

表 3.1 对方 ThroughPass 场景测试结果

	Goalie1	Goalie2	Goalie3
对方 ThroughPass 数	1406	857	539
对方 ThroughPass 成功率	0.9211	0.7013	0.6568

由结果可以看出，改进后的守门员站位策略使对方 ThroughPass 出现的次数以及成功率明显降低。而且在线规划产生的方法 Goalie3 比结合了统计方法的人工编码 Goalie2 在应对对方 ThroughPass 时，效果更好。

3.6 本章小结

本章先介绍了马尔科夫决策过程的求解算法，包括离线方法和在线方法，然后引入了分层分解技术来处理马尔科夫决策过程问题，并用这种技术对 WrightEagle 球队进行了建模分析。然后对 WrightEagle 采用的评估系统进行了简要介绍。并将本章涉及的技术与第 2 章的反算技术结合，解决一类对手模型未知、需要对对手建模的智能体决策问题，并在守门员站位问题中进行了实验，新的方法比传统方法更为灵活，取得了较理想的结果。

第四章 多智能体分层协作规划的原理和应用

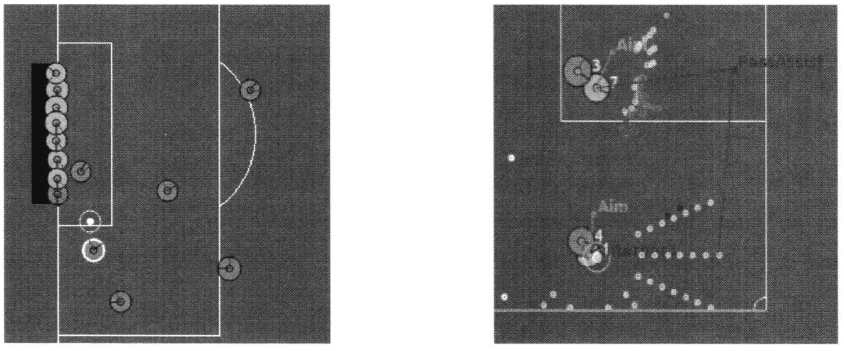
MAXQ-OP 作为单个智能体决策规划算法，在处理多个球员智能体合作的问题时，没有充分考虑其他智能体，特别是潜在的合作对象的行为决策对根任务规划执行结果的影响。故提出基于 MAXQ-OP 框架的多智能体在线规划算法 MAXQ-MOP，这是本文涉及到的第三个工作，也是全文最重要的部分。本章剩余内容主要安排如下：第 4.1 节介绍 RoboCup 仿真 2D 平台的一些背景问题；第 4.2 节引入了信念池和联合策略的概念；第 4.3 节详细介绍 MAXQ-MOP 的算法主要流程；第 4.4 节展示了 WrightEagle 如何使用 MAXQ-MOP 算法处理第 4.1 节中提到的问题；第 4.5 节提出了几种对 MAXQ-OP 算法进行优化的方法；第 4.6 节通过对比实验对相关方法的效果进行检验；第 4.5 节给出了本章小结。

4.1 问题的提出

以RoboCup仿真2D足球为例，比赛中经常会遇到如下两种场景带来的问题：

问题 1：在比赛中的任意球模式下，多名防守球员经常需要一起排成人墙队形，如图 4.1(a)所示，处于蓝色球队的任意球模式，黄色球员排出一字型人墙以阻止对方通过任意球得分。在组成人墙的过程中，需要尽可能多的球员在有限时间内占据尽可能有利的位置。一旦过了比赛规则规定的时间，比赛将进入 Play On 模式，如图 2.2 所示。这时如果人墙中有因为球员未到位而产生的漏洞，则对方选择从此处射门的概率及射门成功的概率都很高。

问题 2：在比赛中，队友球员之间经常需要进行传接球配合的协作行为。在这个过程中，一个球员将球踢给另一个球员，后者去把球截住。比赛中会出现带球球员被对方防守球员压制到边线附近，如图 4.1(b)所示，黄色球员带球进攻，蓝色防守球员把持球者压制在边线附近。图中场景所示的那场比赛中，持球者在图中所示位置选择直接将球传向黄色 7 号球员身前的红点位置，结果在调整球的位置、将球朝目标点踢出的过程中被蓝色 4 号球员将球产出界外。



(a)问题 1 中的人墙示例　　(b)问题 2 中场景示例

图 4.1　相关问题示例图

现有传球策略大多无法顺利将球传到队友球员可以利用的位置，需要队友球员及时做出接应行为。

4.2 信念池和联合策略

协调智能体之间的行为是多智能体系统规划算法的一个基本目标。在离线算法中，为了确保行为的一致，多个智能体预先计算好一致的联合策略并分布式地执行。但是在类似于仿真 2D 足球的部分可观察环境中，每个智能体获得的是不同的局部信息。在本文提到的算法中，每个智能体都维护一个关于所有智能体（至少是可能存在协作行为的智能体）的信念池 $<\{H_i^t \mid i \in I\}, B^t>$。其中，$H_i^t$ 是索引为 i 的智能体的历史序列集合，B^t 表示的则是其历史的联合信念状态集合。在策略生成的时候，使用相同的搜索树扩展方法，使得每个智能体在进行规划的时候，基于相同的联合策略。

4.2.1 信念池

以仿真 2D 中的 Defense 任务为例，防守球员经常需要规划 Mark 子任务，以盯防对方的进攻球员。这时，防守球员的信念池中会包含对方进攻球员的相关信息，如：距离自身最近的对手球员的位置、速度信息。只要队友球员之间维护相同的信念池，则通过这个信念池计算得到的联合策略理论上也会是相同的。即要使在线规划时对所有防守球员进行统一的分配、不出现误协调的情况，就需要每个防守球员智能体关于最近进攻球员的信息保持一致。误协调是指所有的球员智能体计算出的联合策略不一致；反之，如果所有的球员智能体计算出的联合策略是一致的，且一直按照这个联合策略执行相应的动作，就称球员智能体之间的

行为是协调的。

4.2.2 局部策略和联合策略

球员智能体的局部策略 $\pi_i : H_i \to A_i$，是一个从历史信息集合到动作集合的映射。π_i 可以看作是一棵独立的策略树。其中 $\pi_i(h_i)$ 表示的是由历史信息 h_i 通过策略 π_i 导出的动作。由所有球员各自的局部策略组成的一个 n 元组 $\pi = <\pi_1, \pi_2, \cdots, \pi_n>$ 被称作联合策略。π 可以看作是全部智能体各自的策略树的集合。其中 $\pi(h)$ 表示的是由联合历史 h 通过联合策略 π 导出的联合动作。策略树上的每一个节点代表智能体可能要执行的一个动作，树上的每一条边代表智能体执行动作后可能获得的状态信息。前面介绍的离线算法的主要思路就是事先计算出这样的策略树集。

在策略规划时，根据所有可能的历史信息 h^t，每个球员智能体计算联合策略 π^t。在动作执行时，索引号为 i 的球员智能体首先根据 Soccer Server 发来的信息更新自己的局部历史信息，即 $h_i^t \leftarrow h_i^{t-1} \circ s_i^t$。然后，球员智能体根据新的局部历史信息和 π^t 执行动作 $a_i^t \leftarrow \pi_i^t(h_i^t)$。接着，球员智能体再把执行过的 a_i^t 加入到 h_i^t 中。最后，每个球员智能体根据联合策略 π，来更新各自信念池中的相关信息。更新步骤如算法 4.1 所示。完成更新后进入下一个决策周期的规划、执行阶段，如此循环反复。

1. 对于 $\forall h^t \in H^t$ 执行
2. $\vec{a} \leftarrow \pi^t(h^t)$
3. $h^{t+1} \leftarrow h^t \circ \vec{a} \circ \vec{s}$ // 把 $\vec{a}$ 和 $\vec{s}$ 加入到 h^t
4. $p(h^{t+1}) \leftarrow p(\vec{s}, \vec{a} \mid h^t) p(h^t)$ //计算 h^{t+1} 的分布
5. 如果 $p(h^{t+1})$ 大于 0，则 //测试 h^{t+1} 是否可达
6. $H^{t+1} \leftarrow H^{t+1} \cup \{h^{t+1}\}$
7. 用 $b^t(\cdot \mid h^t)$，$\vec{a}$，$\vec{s}$ 更新 $b^{t+1}(\cdot \mid h^{t+1})$ //计算 h^{t+1} 的信念状态
8. $B^{t+1} \leftarrow B^{t+1} \cup \{b^{t+1}(h^{t+1})\}$
9. 返回 H^{t+1} 和 B^{t+1}

算法 4.1 历史展开与信念更新

4.3 MAXQ-MOP 算法流程

MAXQ-OP 作为单个智能体决策规划算法，在处理多个球员智能体合作的问题时，没有充分考虑其他智能体，特别是潜在的合作对象的行为决策对根任务规

划执行结果的影响。故提出了多智能体在线规划算法 MAXQ-MOP，其主要算法流程如算法 4.2 所示：

1. 对于智能体 $i \in I$ 执行：
2. 执行动作 a_i^0 ，
3. 初始化 s, h^0, H^0, B^0, G_0
4. 对于 t=1 到 T 执行
5. $h_i^t \leftarrow UpdateHistory(s_i^t, i)$ //更新自身的局部历史信息
6. $H^t \leftarrow Expand(H^{t-1})$ //历史信息展开
7. $B^t \leftarrow Expand(B^{t-1})$ //信念状态展开
8. 如果 $false = isConsistent(H^t, s_i^t)$ 且 $true = isAvailable(communication)$ 则
9. $Synch(h_i^t, j), j \in I \cap j \neq i$ //把局部历史信息同步给其他智能体
10. 如果 $true = hasCommunicated(i)$ 则
11. $h_i^t \leftarrow UpdateHistory(i)$
12. $a_i^t \leftarrow \arg\max_a Q(a, b^t(h^t))$ //由联合信念状态计算当前的最优动作
13. $H^t \leftarrow UpdateFrom(h^t)$
14. $B^t \leftarrow UpdateFrom(b^t(h^t))$
15. 否则
16. $\pi^t \leftarrow Search(H^t, B^t)$
17. $a_i^t \leftarrow GetAction(\pi^t(a_i \mid h_i^t))$
18. $H^t \leftarrow UpdateFrom(\pi^t)$ //基于联合策略更新历史信息
19. $B^t \leftarrow UpdateFrom(\pi^t)$ //基于联合策略更新信念状态
20. $h_i^t \leftarrow Update(a_i^t, i)$ //更新智能体自身的局部历史信息
21. 执行 a_i^t

算法 4.2 MAXQ-MOP 算法框架

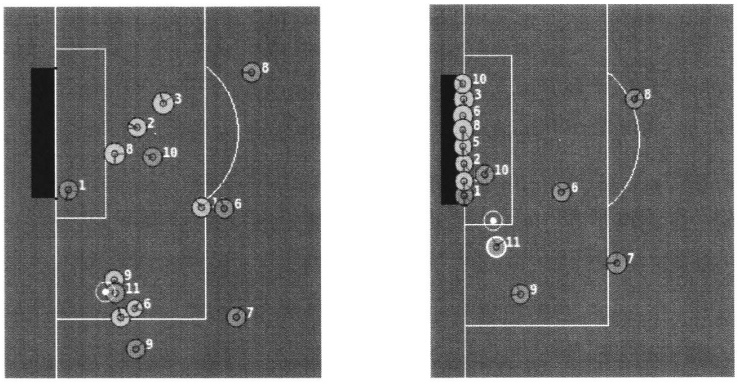
其中，$a_i^t \leftarrow \arg\max_a Q^t$ 和 $a_i^t \leftarrow GetAction(\pi^t(a_i \mid h_i^t))$ 均使用 MAXQ-OP 方法得到动作 a_i^t。在每个决策周期，每个智能体首先检测是否可以通讯。然后进行通讯决策，根据结果判断是否需要与其他智能体进行通讯[55]。这里，通讯的信息是每个智能体在上一个决策周期内获得的局部环境状态信息。

每个智能体在每个决策周期首先扩展关于所有智能体球员的信念池，并根据从环境获得的视觉、听觉信息更新自己的局部历史信息。如果自己的局部历史 h_i^t 跟联合历史信息 H^t 不一致且通讯可用，则把新的信息广播给其他智能体球员。如果智能体球员通讯过，则构建通讯联合历史信息 h^t，并从中计算联合信念状态

$b_t(h^t)$，得到可以使全体智能体球员报酬函数最大的动作，然后更新所有智能体球员的信念池信息。否则，从信念池信息中随机搜索联合策略 π^t 和动作 a^t，然后合并历史信息得到新的关于所有智能体球员的信念池。最后智能体球员根据执行过的动作更新自己的局部历史信息，再结合联合策略更新关于所有智能体球员的信念池信息。

4.4 MAXQ-MOP 算法应用

在 RoboCup 仿真 2D 比赛中，在间接任意球的比赛模式中，防守的一方一般都会排出人墙来阻挡对方的进攻。人墙的站位可以是固定的，也可以是根据场上形势动态生成的。WrightEagle 队一般情况下采用固定站位点的方式，触发任意球时联合状态为初始状态,如图 4.2(a)所示，球员可能在球场的任何位置；所有防守球员都站在球门线上为 WrightEagle 队目前设定的一种目标状态，如图 4.2(b)所示，目标状态不只一种，每个球员在人墙中的位置并不是绝对固定的。则第 2 节中的问题 1 转换成多个智能体和多个目标点之间的匹配问题，问题的难点在于每个智能体对环境的观察有不确定性，是各自不同的局部信息。



(a)初始状态　　(b)人墙完成状态

图 4.2　RoboCup 仿真 2D 中的人墙站位问题

传接球是多个智能体之间最重要的合作任务之一。根据传球目标点相对于接球球员的位置，可以分成 DirectPass，AheadPass，ThroughPass 等几种类型。其中，DirectPass 是将球直接传给某个队友球员，AheadPass 是将球踢向队友球员身前某个位置，ThroughPass 是将球踢向球场中某个空档位置、且使得队友球员能够最快抢到球，如图 3.4 所示。比赛中会出现带球球员被对方防守球员压制到边线附近、无法将球传出，最终导致带球球员自己把球带出界或者被对方防守球

员破坏出界的情况。现有传球策略大多无法顺利将球传出，需要队友及时做出接应行为，即在队友还处于护球或者带球状态、尚未将球传出时，提前通过 Position 行为到达即有利于队友接球、又有利于进攻的位置。这种传球行为由接应者主动配合跑位，与传球者共同完成。

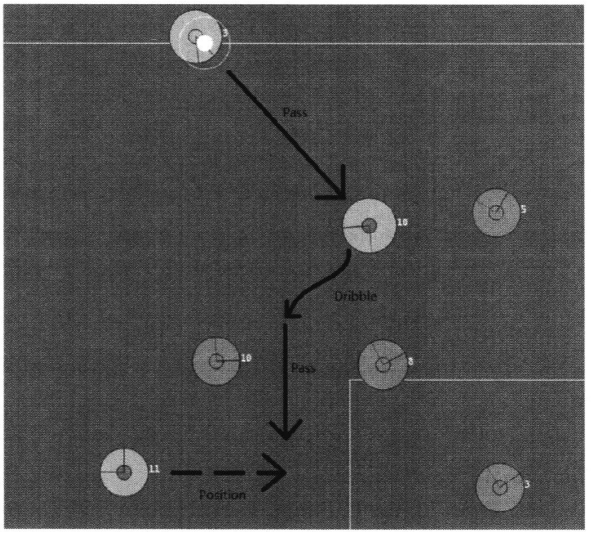


图 4.3 多球员协作示例

以图 4.3 所示的多球员之间的传球协作为例，球员智能体首先利用表 4.2 中 MAXQ-MOP 算法更新了信念池。然后，处于进攻方角色的球员智能体会优先规划 Attack 子任务，其中持球者会优先规划 Attack 的 Pass 子任务，持球者附近的其他队友球员会优先规划 Position 子任务。根据传球目标点的不同，规划 KickTo() 子任务时会有不同角度、速度等参数，产生不同的 turn 和 kick 组合序列，其中 kick 动作踢球，turn 动作调整智能体球员身体角度。黄色 3 号球员在规划 Pass 子任务的时候，通常把连续的球场空间离散化，但对同一个接球球员来说，仍然会产生大量的搜索树节点，在图 2.4(a)、图 4.1(b)中队友球员身前的白色点阵，是持球者规划 Pass 子任务时成功率高于一定阈值的目标点集，每个白色点可能代表不只一个 Pass 行为，因为还可能是不同的速度值。表 4.3 中列出的是这些节点中报酬函数值最大数值，其对应的动作从 MAXQ 分层结构底层返回给 Root 任务，然后智能体球员执行 Root 任务规划的动作。

从中可以看出，黄色 10 号球员接到黄色 3 号球员接球后，带球一段距离再把球传给黄色 11 号队友球员的成功率和报酬函数值都比直接传给黄色 11 号队友球员的高。

表 4.2 图 4.3 中各球员传球成功率及报酬函数值

传球者	接球者	成功率	报酬函数值
3 号	10 号	0.750987	0.599005
3 号	11 号	< 0.0001	< 0.0001
10 号	3 号	0.9383	0.819741
10 号不带球直接传	11 号	< 0.0001	< 0.0001
10 号带球后再传	11 号	0.578827	0.456869

4.5 MAXQ-MOP 的优化方法

4.5.1 搜索过程的加速

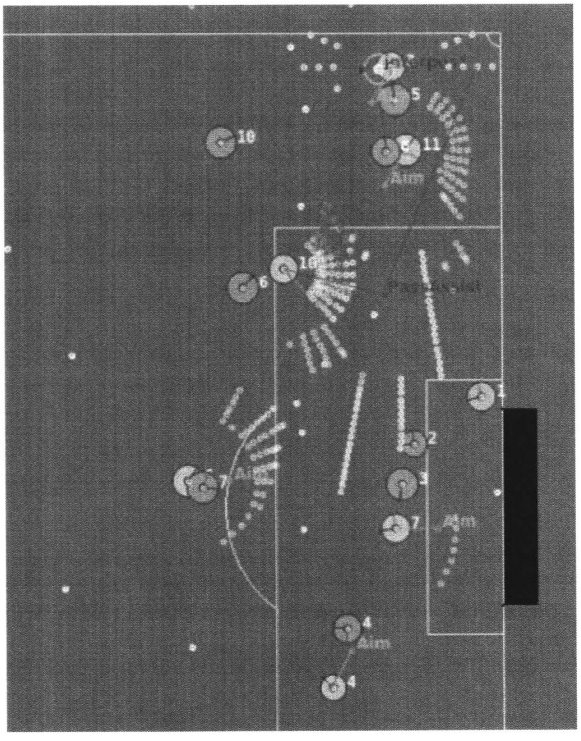


图 4.4 Pass 策略的部分计算结果示意图

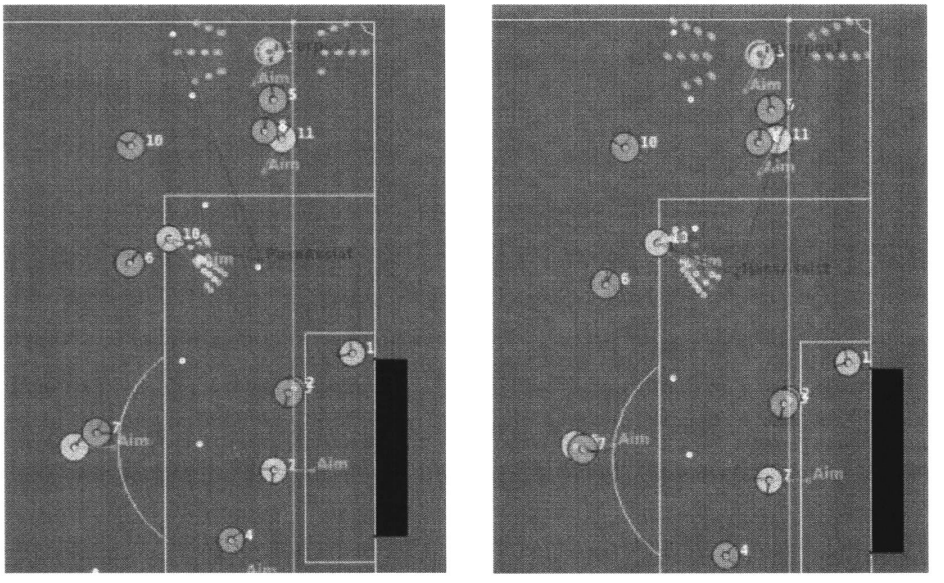
在球员智能体计算自己的或者反算队友球员的 Pass 子任务或者 Dribble-Pass 子任务时，会产生大量的行为策略结果或中间行为策略结果。这些策略可以看作是一棵策略树。如图 4.4 所示，图中黄色球员身前的白色、黄色、红色点阵是黄

色 11 号球员计算的可能给队友球员传球的位置点。而且这些显示出来的位置点是已经经过对报酬函数值筛选之后的结果，也就是说，那些报酬函数值小于一定阈值的结果并未在图中显示。这只是球员智能体一步规划产生的结果数量。如果智能体之间向要完成如图 4.3 那样的协作，需要规划多步，由于“维度诅咒”，最终目标状态空间的巨大量级使得计算变得不可能。但是，这些中间规划步骤中的某些结果并不是都能到达最终目标。所以对策略树进行剪枝是合理而且很有必要的。

以图 4.4 所示的两步规划 Dribble-Pass 子任务为例进行说明，可以考虑采用的剪枝策略大致分为两类：

（1） 去除策略树中动作执行成功概率满足一定条件的节点。这个成功概率既可以是一步行为执行成功的概率，也可以是多步行为中最后一步行为执行成功的概率。前者多用于规划一步行为，如 Dribble、Shoot 等；是后者则可以用其中每一步行为成功概率的加权累加表示，用于规划多步行为，如 Dribble-Pass。

（2） 合并策略树中报酬函数值相近且动作类型相同的节点。可以使用凝聚的层次聚类方法，合并策略树所有节点里欧氏距离小于一定阈值的那些节点。这里，简单地定义两个行为之间的欧氏距离为：$\sqrt{(x_1-x_2)^2+(y_1-y_2)^2+(p_1-p_2)^2+(r_1-r_2)^2}$，其中，$(x_1,y_1)$ 和 (x_2,y_2) 分别是两个行为目标点坐标；如果是两步行为 Dribble-Pass，则为其中第一步 Dribble 的目标点坐标。p_1 和 p_2 分别是两个行为成功执行的概率；如果是多步行为，则为两个行为各自成功执行的累积加权概率之和。r_1 和 r_2 分别是两个行为的立即报酬；如果是多步行为，则为两个行为各自的累积加权报酬之和。如图 4.5（a）和图 4.5（b）所示，是前后相邻的两个仿真周期内，黄色 10 号球员智能体反算出来黄色 3 号球员传球路径的不同：图 4.5（a）所示是黄色 3 号球员先回撤、再将球传向标有“PassAssist”的红点；图 4.5（b）所示的是黄色 3 号球员带球向对方底线附近、再将球传向标有“PassAssist”的红点。这种差异是由于沿这两条传球路径的执行的成功概率和报酬值非常接近，所以在前后两个周期出现规划出来的行为策略不一致的情况。这里，本文中选取 Dribble-Pass 子任务规划出来的行为的第一步 Dribble 的目标点位置坐标、Dribble-Pass 成功的概率和 Dribble-Pass 的报酬值作为凝聚的层次聚类的相关参数。考虑了中间步骤的坐标值，就不会把图 4.5 中两种行为通过层次聚类合并成一个新的行为了。具体算法流程可以参见算法 4.3。



（a） （b）

图 4.5 不同位置对合并行为的影响

1. 根据行为成功的累积加权概率之和与累加加权报酬之和对策略树中的策略进行剪枝，低于一定阈值的行为从候选行为集合中删除
2. 计算候选行为集合中任意两个行为之间的欧氏距离，得到相似性矩阵
3. repeat
4. 合并距离最接近且小于阈值 δ 的两个行为
5. 更新相似性矩阵，
6. until 只剩下一个行为或者剩下的行为之间的距离都大于 δ
算法 4.3 基于凝聚层次聚类的 MAXQ-MOP 优化方法

4.5.2 更有效的通讯

RoboCup 仿真 2D 足球中，球员智能体之间是单通道、低带宽、不可靠的通讯方式。也就是说话者每次 say 的内容不能超过 10 个字节，而且说话者不知道对方是否听到。通讯的信息是公开的，对手球员智能体如果这时处于 attention to 说话者的情况下，就很有可能也听到说话者所说的内容。还有一点就是听力有限，球员智能体每次智能 attention to 两个人，一个是对手球队中的某一人，另一个是队友中的某一个。

所以，对球员智能体之间的通讯进行规划也是很有必要的。如果每周期轮流

attention to 一个队友，则至少需要 10 个仿真周期才能通讯一轮，而这个时间不能满足多个球员间传接球协作的实时性要求。要改善这种情况，也有大致两种方法：

（1） 减少 attention to 的目标队友集合：考虑到球员通讯的时候一般只与阵型中相邻的队友球员联系，所以根据球队的攻防决策、球员的队中角色、正在决策的动作类型等，每个球员都会有一个通讯的优先级列表。通常进行通讯决策时，球员智能体从通讯优先级列表中选择优先级最高的队友球员执行 attention to。

（2） 减少说话者和监听者的不同步现象：基于相同的前提信息，合作中的每个智能体能计算出一致的联合策略。进而减少说话者的目标对象未听到通讯信息的情况。避免出现说话者广播了一些信息，但队友球员却没听到的情况。

MAXQ-MOP 在每周期开始时，如果检测到通讯是不可用的，就会推迟通讯到下一个决策周期。这里的通讯包括自己向外广播信息和从外界接收听到的信息。

4.6 实验及结果分析

基于前面章节中对问题 1 和问题 2 的建模，本节在 WrightEagle 仿真 2D 足球队中实现了相关算法。利用球队的训练器 Trainer 功能，反复重现并测试了对方间接任意球和多球员传球协作的场景。测试结果参见表 4.5 和表 4.6。测试实验使用的硬件环境是：Intel Core i7 950，6GB Memory，软件环境是 Ubuntu 14.04 64 位版，rcssserver15.2.25。其中：

Hand-coded：是手工编码方法。即事先给每个球员指定其在人墙中的站位。站位信息是利用 Trainer 大量测试不同场景统计得出的完成率最高的一种站位方法。

RTDP：实时动态规划算法[56][57]。球员根据值函数上界提供的启发式信息采用一步前瞻式的贪婪策略选择行动并执行。然后根据 Soccer Server 中行动模型随机选择后继状态。这里，每个球员根据自己对场上的信息的感知，在组成人墙的时候，总是把自己的目标点设置为距离自己最近的点位。

MAXQ-MOP：本文中介绍的基于 MAXQ-OP 框架的多智能体在线规划算法。

MAXQ-MOP*：MAXQ-MOP 的优化版本，在 MAXQ-MOP 的搜索状态空间

[5] 即 Soccer Server 的代码实现版本，截至目前，最新的版本由 RoboCup 官方于 2013 年 6 月 17 日发布，下载地址： http://sourceforge.net/projects/sserver/files/rcssserver/

的时候，结合一些启发式信息，对搜索树进行剪枝，加速其搜索过程。

为了排除平台环境的不确定性对测试结果的影响，测试人墙排列的实验中对每个算法分别测试了如图 4.2(a)在内的 20 个场景共 2000 轮测试。测试结果如表 4.5 所示。其中，完成数是指 2000 轮测试中完成人墙站位的次数。平均耗时是指完成人墙站位的时候所消耗的周期数（1 周期=100 毫秒）。超时数是指在比赛允许组成人墙的时间内，球员并没有完全排成应有的站位。失球数是 2000 轮测试中总的失球数目。

表 4.3 人墙站位实验测试结果

	Hand-coded	RTDP	MAXQ-MOP	MAXQ-MOP*
完成数	1237	1072	1545	1797
平均耗时	27.05	29.77	28.43	25.33
超时数	763	928	455	203
失球数	1059	1377	801	597

RTDP 方法在这里表现最差，因为实现时没有充分考虑其他智能体球员的行动，经常出现多个球员往同一个目标点移动的冲突情况，等到目标点已经确定会被队友球员占领，智能体再重新规划新的目标点，故耗时最多。Hand-coded 方法在事先已经解决了这个冲突问题，智能体球员只需要往既定的目标点移动即可，但是会出现较远的球员体力不足、未能完成人墙排列的情况。MAXQ-MOP 算法耗时比 Hand-coded 版本略长，但是人墙站位完成率为 77.25%，比前两种方法高，实际效果明显得到提升。MAXQ-MOP*则在保证了效果的情况下，减少了耗时，是测试的四种方法中效果最好的，人墙站位完成率为 89.85%。

测试球员间的传接球效果使用 Trainer 工具，反复测试持球球员在被对方防守球员压迫防守的场景。同样，为了排除测试的不确定性对结果的影响，测试传球的实验中对每个算法分别测试了如图 4.1(b)、图 4.4 等 5 个场景共 1500 轮测试。测试结果如表 4.6 所示。其中，成功数是指 1500 轮测试中持球球员成功将球传到队友脚下的次数。被截断数是指持球者传出的球第一时间被对方球员得到、进攻行为中断。出界数是 1500 轮测试中由于对方球员破坏或者自己带球失误导致球出界的数目。

表 4.4 特定场景传接球实验测试结果

	Hand-coded	MAXQ-OP	MAXQ-MOP	MAXQ-MOP*
成功数	596	1057	1168	1243
被截断数	572	215	203	144
出界数	332	228	129	113

从测试结果中可以看出，Hand-coded 方法在传球成功率最低，被对方球员截断传球的比例最高。MAXQ-OP 方法的传球成功率约是 70.4667%，而 MAXQ-MOP 方法的成功率约为 77.8667%，传球效果明显得到提升。图 4.1(b)和图 4.4 中红线是 MAXQ-MOP 算法规划出来的结果，由 WrightEagle 球员在比赛中记录到 log 文件中，再通过第二章中的改进的 rcsslogplayer 工具解析、图形化展示出来。

4.7 本章小结

本章在前面章节内容的基础上，提出了适用于多智能体的分层协作规划在线求解算法 MAXQ-MOP，并介绍了一些优化方法。然后在 RoboCup 仿真 2D 足球实际比赛的一些场景里对 MAXQ-MOP 方法及其优化算法 MAXQ-MOP*进行了对比实验。测试结果表明：在这类多智能体协作场景中，使用 MAXQ-MOP 方法比传统方法更有效合适；如果结合一些剪枝等优化手段，MAQX-MOP 的效果会进一步提升。

MAXQ-MOP 方法适用于通讯受限的、队友模型已知的多智能体协作行为规划问题。除了 RoboCup 仿真 2D 足球，该方法还可以用于现实中仓储机器人在仓库狭窄的通道里避让等问题。

第五章 总结与展望

5.1 总结

构造可以通过决策产生智能行为的智能体可以看作是人工智能现阶段的主要目标。各种决策算法使得智能体能够在多方面可以近似做出人类可以做出的智能行为。在大规模不确定性环境下，本文以马尔科夫决策过程为理论基础，描述和处理多智能体协作规划问题。RoboCup 仿真 2D 足球为实验平台，并使用的场景重现的 Trainer 作为测试工具。

首先，本文介绍了多智能体系统环境的特点，并分类介绍了马尔科夫决策过程的相关理论模型，其中马尔科夫决策过程是本文最重要的理论基础。

其次，介绍了 RoboCup 仿真 2D 足球比赛平台 Soccer Server。为了有效地对后续工作进行有效测试，使用 C++和 Qt 等程序设计语言，改进实现了 Trainer 和 rcsslogplayer 工具。针对比赛的特点，对 WrightEagle 球队进行建模分析。

然后，对单个球员智能体使用 MAXQ-OP 算法，并以 WrightEagle 球队为例，介绍了分层分解技术在马尔科夫决策过程中的应用。结合反算技术，在测试守门员站位对对方球员传球行为影响的实验中，这种方法取得比原有方法更好的效果。

最后，本文针对一类特殊的多智能体合作决策问题，提出了基于 MAXQ-OP 框架的多智能体在线规划方法 MAXQ-MOP，并在 RoboCup 仿真 2D 足球比赛的人墙站位和多人传接球中对算法进行了实验。实验结果表明，这个方法使得多个球员间合作成功的概率有了较大提高。另外，还通过凝聚层次聚类等剪枝方法对 MAXQ-MOP 算法进行了一些优化，得到表现效果更好的 MAXQ-MOP*算法。MAXQ-MOP*算法在对比实验中的表现基本的 MAXQ-MOP 更好一些。

本文所有的工作都是基于 WrightEagle 球队实现的。

5.2 展望

尽管本文在上述一些问题中的研究确定了一定的成果，但是多智能体系统的规划问题仍然有很多复杂的、具有挑战性的工作。比如：

- 由于维护一个信念池的开销较大，目前在少数的球员智能体之间的协作行为（如：两三个球员之间的传接球配合）可以有效使用，尚无法有效维护包括对方球员在内的全体球员的信念池；

- 虽然第 4.6 节的实验结果显示优化后的 MAXQ-MOP 表现效果比传统方法更好，但是还有提升的空间。

后续的工作包括继续对其进行分析，期待该方法能够应用于更多的多智能体决策问题中。

现阶段，我们正在备战在中国合肥举办的 2015RoboCup 机器人世界杯。前文中提到的后续工作也在继续推进中。希望 WrightEagle 在今年的仿真 2D 组比赛中再度夺魁！

致 谢

特别感谢我的导师陈小平教授。能成为陈老师的学生，我感到非常荣幸。他严谨的科研态度、深厚的学术功底、精益求精的治学理念、高瞻远瞩的洞察力都深深地感染了我，使我收益良多。在三年的时间内，陈老师给我们提供的不只是自由的研讨氛围和良好的学习环境，还有耐心的指导、鼓励和关心。陈老师的每句教诲都将使我终生受益。

同时，需要感谢实验室曾经和现在的黄巍教授、吉建民教授和吴锋教授。跟三位老师的很多讨论，极大地拓展了我的视野，加深了我对相关问题的理解，引领我在很多研究课题中积极探索。特别感谢三位老师对我的论文的精心指导，使我收获颇丰。

其次，我要感谢 WrightEagle 机器人足球队仿真 2D 组的所有成员们，包括：范长杰、石轲、柏爱俊、张昊翀、卢光辉、江淼、李萧等。WrightEagle 仿真 2D 球队的发展和取得的成绩，离不开所有新老成员的共同努力。还有多智能体系统实验室的其他成员，包括：靳国强、吕强、张栋翔、谢炯坤、孙昊、隋志强、郭群、柯翔、娄健、程敏、林志强、陈凯、赵哲、卢栋才、程彬、陈赢峰、唐可可、谢宗俊、王宁扬、帅威、刘江川、陈张、谷雨茂、戴元臻、刘洋等。感谢你们使我在实验室的学习和生活充满了无穷的乐趣。

还有，我还要感谢计算机科学与技术学院的各位老师，感谢老师们这些年对我们学习和生活的关怀和帮助。

最后，谨以此文献给我的父母和所有的亲人。是你们不断地鼓励和默默地支持，让我顺利完成学业，我真的非常感激。在此我向你们表达最诚挚的谢意！

2015 年 4 月

在读期间发表的学术论文与取得的研究成果

学术论文：

[1] 陈荣亚，陈小平．多智能体分层协作规划及在 RoboCup 中的应用．计算机系统应用（审稿中）．

其他研究成果：

[1] 2014 RoboCup 机器人世界杯仿真足球 2D 组比赛冠军，巴西若昂佩索阿，2014 年 7 月

[2] 2014 中国机器人大赛暨 RoboCup 公开赛仿真足球 2D 组比赛冠军，中国合肥，2014 年 10 月

[3] 2013 RoboCup 机器人世界杯仿真足球 2D 组比赛冠军，荷兰埃因霍温，2013 年 6 月

[4] 2013 中国机器人大赛暨 RoboCup 公开赛仿真足球 2D 组比赛冠军，中国合肥，2013 年 10 月

[5] 2012 中国机器人大赛暨 RoboCup 公开赛仿真足球 2D 组比赛冠军，中国合肥，2013 年 12 月

参考文献

[1] Soham Y. Agent-oriented programming. Artificial intelligence, 1993, 60(1):51-92.

[2] Frank lin S. Graesser A. Is it an Agent, or Just a Program? : A Taxonomy for Autonomous Agents. Proceedings of Proceedings of the Workshop on Intelligent Agents III, Agent Theories, Architecuures, and Languages, London, UK, UK: Springer-Verlag, 1997,21-35.

[3] Wooldridge M. Agent-based software engineering. IEEE Proceedings-software, 1997,144(1):26-37.

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

[5] Singh MP. Multiagent System: A Theoretical Framework for Intentions, Know-How, and Communications. Springer-Verlag KG, 1994.

[6] Stuart Russell, Peter Norvig. Artificial Intelligence: A Modern Approach. Prentice Hall; 3 edition December, 2009.

[7] Ferber, Jacques. Multi-agent systems: an introduction to distributed artificial intelligence. Vol. 1. Reading: Addison-Wesley, 1999.

[8] Weiss, Gerhard, ed. Multiagent systems: a modern approach to distributed artificial intelligence. MIT press, 1999.

[9] Ascent, Coordinate. Multi-Agent Decision Making. (2004).

[10] Littman, Michael Lederman. Algorithms for sequential decision making. Diss. Brown University, 1996.

[11] Tarjan, Robert. Depth-first search and linear graph algorithms. SIAM journal on computing 1.2 (1972): 146-160.

[12] Zhou, Rong, and Eric A. Hansen. Breadth-first heuristic search. Artificial Intelligence 170.4 (2006): 385-408.

[13] Korf, Richard E. Depth-first iterative-deepening: An optimal admissible tree search. Artificial intelligence 27.1 (1985): 97-109.

[14] Ginsberg, Matthew L., and William D. Harvey. Iterative broadening. Artificial Intelligence 55.2 (1992): 367-383.

[15] Steinbiss, Volker, Bach-Hiep Tran, and Hermann Ney. Improvements in beam search. ICSLP. Vol. 94. No. 4. 1994.

[16] Nilsson N J. Artificial Intelligence: A New Synthesis. Morgan Kaufmann Publishers, Inc.,

1998.

[17] Puterman, Martin L. Markov decision processes: discrete stochastic dynamic programming. John Wiley & Sons, 2014.

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

[19] Nils J Nilsson. Principles of artificial intelligence. Springer, 1982.

[20] Cameron Browne, Edward J Powley, Daniel WhiteHouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Travener, Diego Perez, Spyridon Samothrakis ,Simon Colton. A survey of Monte Carlo Tree Search Methods. IEEE Trans. Comput. Intellig. and AI in Games, 2012, 4(1):1-43.

[21] Poupart, Pascal. Partially observable Markov decision processes. Encyclopedia of Machine Learning. Springer US, 2010. 754-760.

[22] Cassandra, Anthony R., Leslie Pack Kaelbling, and Michael L. Littman. Acting optimally in partially observable stochastic domains. AAAI. Vol. 94. 1994.

[23] Cassandra, Anthony R., Leslie Pack Kaelbling, and James A. Kurien. Acting under uncertainty: Discrete Bayesian models for mobile-robot navigation. Intelligent Robots and Systems' 96, IROS 96, Proceedings of the 1996 IEEE/RSJ International Conference on. Vol. 2. IEEE, 1996.

[24] M. Baykal-Gursoy, K. Gursoy. Semi-Markov Decision Process: Nonstanard Criteria. In the Engineering and Informational Sciences, 21, 2007, 635–657. U.S.A.

[25] Sutton R S, Precup D,Singh S. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. Artificial Intelligence, 1999, 112(1): 181-211.

[26] Monahan, George E. State of the art—a survey of partially observable markov decision processes: Theory, models, and algorithms. Management Science 28.1 (1982): 1-16.

[27] Shapley, L. S. (1953). Stochastic games. PNAS 39 (10): 1095–1100. doi:10.1073/pnas.39.10.1095.

[28] Eric A. Hansen, Daniel S. Bemstein and Shlomo Ziberstein. Dynamic Programming for Partially Observable Stochastic Games. 19th National Conference on Artificial Intelligence(AAAI-04).

[29] Kitano H, Asada M. RoboCup humanoid challenge: That's one small step for a robot, one giant leap for mankind. Proceedings of Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on Intelligent Robots, volume 1, 419-424.

[30] Hiroaki Kitano，Minoru Asada，Yasuo Kuniyoshi，et al． RoboCup: a challenge problem for AI [J]． AI Magazine，1997，18(1) : 73-85.

[31] Jasmin Blanchette, Mark Summerfield 著，闫锋新等 译. C++ GUI Qt 4 编程(第二版), 电子工业出版社. 2010. 17-51.

[32] Ke Shi, Aijun Bai, Yunfang Tai, et al. WrightEagle2009 2D Soccer Simulation Team Description Paper. RoboCup International Symposium 2009, July, 2009.

[33] Joshua Grass and Shlomo Zilberstein. Programming with Anytime Algorithms. In Proceedings of the IJCAI-95 Workshop on Anytime Algorithms and Deliberation Scheduling, 2005.

[34] Nikos, Vlassis, R. Elhorst, J. R. Kok. Anytime Algorithms for Multiagent Decision Making Using Coordination Graphs. In Proc. Intl. conf. On Systems, Man and Cybernetics, 2004.

[35] Meyn, Sean P. The policy iteration algorithm for average reward Markov decision processes with general state space. Automatic Control, IEEE Transactions on 42.12 (1997): 1663-1680.

[36] Dai, Peng, and Judy Goldsmith. Topological Value Iteration Algorithm for Markov Decision Processes. IJCAI. 2007.

[37] B. Bonet, H. Geffner. Learning in depth-first search: a unified approach to heuristic search in deterministic, non-deterministic, probabilistic, and game tree settings. In: ICAPS, 2006.

[38] Hernández, C. and Meseguer P. (2005) LRTA*(k). In Proceedings of the 19th International Joint Conference on Artificial Intelligence, (IJCAI 2005), Edinburgh, Scotland.

[39] Michael L Littman, Thomas L Dean, Leslie Pack Kaelbling. On the complexity of solving Markov decision problems. Proceedings of Proceedings of the Eleventh conference on Uncertainty in artificial intelligence. Morgan Kaufmann Publishers Inc., 1995. 394-402.

[40] EA Hansen, S Zilberstein. LAO*: A heuristic search algorithm that finds solutions with loops. Artificial Intelligence, 2001, 129(1-2):35-62.

[41] Aijun Bai, Feng Wu, Xiaoping Chen. Bayesian mixture modelling and ingerence based Thompson sampling in Monte-Carlo tree search. Proceedings of Advances in Neural Information Processing System. 2013:1646-1654.

[42] L Kocsis, C Szepesvari. Bandit based Monte-Carlo planning. Proceedings of European Conference on Machine Learning, 2006. 282-293.

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

[44] Thomas G Dietterich. Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition. Journal of Machine Learning Research, 1999, 13(1): 63.

[45] RS Sutton, D Precup, S Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. Artificial intelligence, 1999, 112(2):181-211.

[46] Ronald Parr, Stuart Russell. Reinforcement learning with hierarchies of machines. Advances

in neural information processing systems, 1998. 1043-1049.

[47] Thomas G Dietterich. Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition. Journal of Machine Learning Research,1999,13(1):63.

[48] Aijun Bai, Feng Wu, and Xiaoping Chen. Online Planning for Large MDPs with MAXQ Decomposition. Proceedings of Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 3. International Foundation for Autonomous Agents and Multiagent Systems, 2012. 1215-1216.

[49] 石轲，陈小平. 行动驱动的马尔科夫决策过程及在 RoboCup 中的应用 [J]. 小型微型计算机系统，2011，32（3）：511-515.

[50] Wu Feng，Chen Xiao-ping. Solving large-scale and sparse-reward DEC-POMDPs with correlation-MDP. Proceedings of RoboCup International Symposium 2007，Atlanta，America，July 2007.

[51] Leslie Pack Kaelbling, Michael L Littman, Anthony R Cassandra. Planning and acting in partially observable stochastic domains. Artificial Intelligence, 1998, 101(1-2):99-134.

[52] Aijun Bai. Belief State in WrightEagle. RoboCup 2012 Soccer Simulation 2D Free Challenge, Mexico City, Mexico, http://home.ustc.edu.cn/~baj/publications/rc_2012_free_challenge.pdf, June,2012.

[53] Frank Dellaert, Dieter Fox, Wolfram Burgard, Sebastian Thrun. Monte Carlo localization for mobile robots. Proceedings of IEEE International Conference on Robotics and Automation, volume 2. IEEE,2001. 1322-1328.

[54] Haochong Zhang, and Xiaoping Chen. The Decision-Making Framework of WrigtEagle, the RoboCup 2013 Soccer Simulation 2D League Champion Team. RoboCup 2013: Robot World Cup XVII. Lecture Notes in Computer Science Volume 8371, 2014, pp 114-124.

[55] Feng Wu, Shlomo Zilberstein, and Xiaoping Chen. Online Planning for Multi-Agent Systems with Bounded Communication. In Artificial Intelligence(AIJ), Volume 175, Issue 2, Page 487-511, 2011.

[56] Feng Wu,Shlomo Zilberstein, and Xiaoping Chen. Trial-Based Dynamic Programming for Multi-Agent Planning. In Proceedings of the 24th AAAI Conference on Artificial Intelligence(AAAI), Page 908-914, Atlanta USA, 2010.

[57] Van Der Wal, J. 1981. Stochastic dynamic programming. In Mathematical Centre Tracts 139. Morgan Kaufmann, Amsterdam.

仿真机器人足球中球员合作策略研究

作者：陈荣亚
学位授予单位：中国科学技术大学