中国科学技术大学

博士学位论文

基于马尔科夫理论的不确定性规划和感知问题研究

作者姓名： 柏爱俊

学科专业： 计算机应用技术专业

导师姓名： 陈小平 教授

完成时间： 二〇一四年九月

University of Science and Technology of China

A dissertation for doctor's degree

Markov Theory based Planning and Sensing under Uncertainty

Author :	Aijun Bai
Speciality :	Computer Science
Supervisor :	Prof. Xiaoping Chen
Finished Time :	September, 2014

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：_____ 签字日期：_____

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：_____ 导师签名：_____

签字日期：_____ 签字日期：_____

摘 要

在人工智能研究领域，基于智能体的形式化方法为智能系统的建模、设计和实现提供了统一的框架。智能体的一个基本特征是其在动态不确定环境中自主感知、行动和学习的能力。处理日益复杂实际问题的智能体，特别是各种形式的智能机器人，已经在人们日常生活和世界经济中扮演着越来越重要的角色，其影响范围可以说是近到人手一部的智能手机，远到遥远太空的人造卫星。通常来讲，感知信息总是不可避免带有各种误差和噪音的；执行机构的执行结果也具有不可预知性，甚至失败的情况。同时，还可能会有各种无法直接观测的隐藏信息。诸如此类的不确定性，为智能体的感知和规划任务带来了巨大的挑战。

以马尔科夫决策过程（MDP）和部分可观察马尔科夫决策（POMDP）为代表的决策论规划理论为这类问题的最优化求解提供了重要的理论和算法基础。完全求解 MDP 和 POMDP 都面临所谓“维度诅咒”问题——即状态空间大小随状态变量的数目呈指数级增加。通过采用在线规划、分层规划、蒙特卡洛仿真、粒子滤波等技术设计 MDP 和 POMDP 的近似求解算法是目前的研究热点。本文以 MDP 和 POMDP 为主要理论依据，主要探讨大规模不确定性环境下的自动感知和规划问题，重点是为大规模 MDP 和 POMDP 问题设计高效的近似算法。特别地，本文提出基于 MAXQ 分层分解的 MDP 在线规划算法——MAXQ-OP，基于后验动作采样的 MDP 和 POMDP 蒙特卡洛在线规划算法——DNG-MCTS 和 D²NG-POMCP，以及基于 POMDP 信念更新模型的集合粒子滤波多对象跟踪算法——PFS。

本文提出的分层在线规划算法——MAXQ-OP，同时结合了分层规划和在线规划的优势，为大规模 MDP 问题的分层在线求解提供了原理性解决方案。具体地，MAXQ-OP 利用问题本身的 MAXQ 分层结构在线求解大规模 MDP 问题，使用启发式方法高效地搜索动作和宏动作空间，并使用启发函数给出搜索树上的终端节点值函数的估计值。MDP 标准测试问题——出租车问题——上的实验结果显示 MAXQ-OP 相比传统在线规划算法，以极少的计算资源消耗，在线找到问题的近似最优解。作为 MAXQ-OP 算法的长期主要实验平台，RoboCup 机器人世界杯—仿真 2D 机器人足球是一个规模特别巨大的完全分布式多智能体随机系统。以 MAXQ-OP 为主要决策框架的算法成功应用到科大“蓝鹰”仿真 2D 机器人足球队中，取得了 RoboCup 2D 比赛多项世界冠军和全国冠军的好成绩，显示了 MAXQ-OP 算法应用于规模巨大的实际问题的重要潜力。

近年来，蒙特卡洛树搜索（MCTS）在不确定性规划和学习领域引起了广泛的研究兴趣。MCTS 的一个基本问题是利用和探索之间的平衡。本文针对 MDP 和 POMDP 的在线规划问题，提出新颖的基于后验动作采样的 MCTS 算法——DNG-MCTS 和 D²NG-POMCP。基本思想是把蒙特卡洛搜索树上某一节点执行

某一动作并服从树上策略的前向仿真过程的累积回报看成是服从某一未知分布的随机变量，引入必要的隐藏变量来参数化这一未知分布，并根据贝叶斯方法更新隐藏变量的后验分布。进一步，使用 Thompson 采样根据某一动作成为最优动作的后验概率来随机选择该动作，以进行树上搜索。本文针对 MDP 和 POMDP 问题，分布提出 DNG-MCTS 和 D²NG-POMCP 算法，实验结果显示提出的算法在多个标准测试问题里面比领域最先进的算法（包括 UCT 和 POMCP）效果更好，表明其有望适用于规模巨大的实际问题，并取得好的实验结果。

自主机器人在动态环境中识别、跟踪和确认潜在的多人状态的能力对成功完成社会化的人—机器人交互任务起到非常关键的重要作用。在线多人跟踪问题等价于复杂 POMDP 的实时信念更新。主要挑战包括：事先不知道实际有多少人；基于计算机视觉算法的人的探测结果不可避免有误报和漏报情况；并且，人和机器人都处于复杂的相对运动当中。针对这些挑战，本文把多人集合看成联合状态，多人探测结果看成联合观察，近似计算相应的联合观察函数，最终提出新颖的基于集合定义的粒子滤波算法——PFS。针对个体确认问题，提出基于期望最大化（EM）的个体确认算法从更新后的联合粒子集合中辨认并报告每一个人的状态信息。较传统多对象跟踪算法而言，基于集合的形式化使得 PFS 不需要进行显式的观察到目标的数据关联，从而在具有复杂噪音和错误的观察情况下具有更好的容错性和鲁棒性。最终的完整 PFS 算法在 PETS2009 数据集中取得了，就 CLEAR MOT 指标而言，比领域前沿算法更好的实验结果。真实机器人平台 CoBot 上的测试结果显示 PFS 应用到实际机器人上的有效性。

关键词： 马尔科夫决策过程，部分可观察马尔科夫决策过程，决策论规划，分层在线规划，蒙特卡洛树搜索，多对象跟踪

ABSTRACT

In the research of Artificial Intelligence, agent-based paradigm aims to provide a unifying framework for conceptualizing, designing, and implementing intelligent systems, that sense, act and learn autonomously in dynamic and/or stochastic environments, to solve a growing number of complex problems. Agents, particularly various kinds of robots, are playing more and more important roles in world economics and people's everyday life, from satellites to smartphones. Generally speaking, perceptional inputs from sensors have inevitable noises and errors. The effects of actuators have also unpredictable impact with noises, or even failures. There may also exist different levels of hidden information that can not be observed directly. Such uncertainties have brought huge challenges to the problem of agent planning and sensing.

Markov decision processes (MDPs) and partially observable Markov decision processes (POMDPs) provide important basis in terms of theory and algorithm to optimal planning and sensing under uncertainty. However, solving large MDPs and POMDPs exactly is usually intractable due to the "curse of dimensionality" – the state space grows exponentially with the number of state variables. To address this challenge in practice, researchers are usually utilizing approximation techniques such as online planning, hierarchical planning, Monte-Carlo simulation, particle filtering, etc. Following the theories of MDP and POMDP, this thesis is focusing on developing efficient approximate algorithms for large MDPs and POMDPs. Specifically, we propose a MAXQ hierarchical decomposition based online planning algorithm – MAXQ-OP; we develop DNG-MCTS and D²NG-POMCP algorithms which apply the idea of Thompson sampling to Monte-Carlo planning in MDPs and POMDPs; and, we develop a particle filtering over sets (PFS) approach to multi-human tracking problem.

The proposed hierarchical online planning algorithm, namely MAXQ-OP, is a novel algorithm that combines the advantages of both online planing and hierarchical planning. It provides a more sophisticated solution for programming autonomous agents in large stochastic domains. Specifically, we perform online decision-making by following MAXQ value function decomposition. We empirically evaluate our algorithm on the Taxi problem – a common benchmark for MDPs. The experimental results show that MAXQ-OP is able to find near optimal policy online, with extremely less computation time comparing to traditional online planning algorithms. The RoboCup soccer simulation 2D domain is a very large test-bed for the research of artificial intelligence. The key challenge lies in the fact that it is a fully distributed, multi-agent stochastic system with continuous state, action and observation spaces. We have conducted a

long-term case study in RoboCup 2D domain and developed a team named WrightEagle that have won multiple world champions and national champions in RoboCup annual competitions. The results of our case study confirm MAXQ-OP's important potential of scaling up to very large domains.

Monte-Carlo tree search (MCTS) has been drawing great interest recently in domains of planning and learning under uncertainty. One of the key challenges is the trade-off between exploration and exploitation. We develop novel approaches, namely DNG-MCTS and D²NG-POMCP, to MCTS by using posterior action sampling to select actions for online planning in MDPs and POMDPs. Specifically, we treat the cumulative reward obtained by taking an action in a search node and following a tree policy thereafter over the Monte-Carlo search tree as a random variable following an unknown distribution. We parametrize the distribution by introducing necessary hidden parameters, and infer the posterior in Bayesian settings. Thompson sampling is then used to exploit and explore the search tree, by selecting an action based on its posterior probability of being optimal. Experimental results confirm that the proposed algorithms outperform the state-of-the-art approaches with better values on several benchmark problems, showing the potential of successfully applying to very large real-world problems.

The ability for an autonomous robot to detect, track and identify potentially multiple humans is essential for socialized human-robot interactions in dynamic environments. Online multi-object tracking problem is equivalent to real-time belief update of a complex POMDP. The main challenge is that, without the knowledge of actual number of humans, the robot needs to estimate each human's state information in real-time from sequentially ambiguous observations, including inevitable false and missing detections, while both the robot and humans are constantly moving. In this thesis, we propose a novel particle filtering over sets (PFS) approach to address this challenge. We define joint states and observations both as finite sets, and develop motion and observation functions accordingly. The target identification problem is then solved by using the expectation-maximization (EM) method, given updated particles. The set formulation enables us to avoid directly performing observation-to-target association, leading to high fault-tolerance and robustness in complex dynamic environments with frequent noises and errors in terms of detections. The overall PFS algorithm outperforms the state-of-the-art, in terms of CLEAR MOT metrics, in PETS2009 dataset. We also demonstrate the effectiveness of PFS on a real robot, namely CoBot.

Keywords: Markov Decision Process, Partially Observable Markov Decision Process, Decision-Theoretic Planning, Hierarchical Online Planning, Monte-Carlo Tree Search, Multi-Object Tracking

目 录

表格索引

插图索引

算法索引

主要符号对照表

MDP	马尔科夫决策过程
POMDP	部分可观察马尔科夫决策过程
DEC-POMDP	分布式部分可观察马尔科夫决策过程
BRL	贝叶斯强化学习
HRL	分层强化学习
MAB	多臂赌博机问题
MOT	多对象跟踪问题
MAXQ	MAXQ 分层分解
MAXQ-OP	基于 MAXQ 分层分解的在线规划算法
MCTS	蒙特卡洛树搜索算法
UCB	置信度上界启发值
UCT	置信度上界树上搜索算法
POMCP	部分可观察蒙特卡洛规划算法
DNG-MCTS	基于后验采样的蒙特卡洛树搜索算法
D²NG-POMCP	基于后验采样的部分可观察蒙特卡洛规划算法
PFS	集合粒子滤波算法
$\mathcal{S}$	状态空间
$\mathcal{A}$	动作空间
$\mathcal{T}$	状态转移函数
$\mathcal{R}$	立即回报函数
$\mathcal{O}$	观察空间
Ω	观察函数
$\mathcal{B}$	信念状态空间
$\mathcal{P}$	粒子集合
$\mathcal{H}$	状态集合
Pr	概率质量或密度函数
E	期望
Var	方差
Cov	协方差
$\mathcal{N}$	正态分布
$\mathcal{U}$	均匀分布

第一章　绪论

内容提要

人工智能（Artificial Intelligence）研究的主要目标是设计和制造出可以完成一般被认为需要智能（Intelligence）才能完成的复杂任务的自主机器或多机器系统，常见的具体实例比如“深蓝”超级电脑、“沃森”问答系统、谷歌无人驾驶汽车、科大“蓝鹰”（WrightEagle）机器人足球、科大“可佳”（KeJia）智能服务机器人等。人工智能的研究领域一般包括自动规划和调度、智能感知、知识表示和推理、机器学习、自然语言处理、计算机视觉、机器人等。人工智能常用的研究方法包括逻辑推理、概率演算、状态空间搜索、数值优化、统计学习等。本文主要关注不确定性环境下的自动感知和规划问题。本章首先给出智能体（Agent）的概念，介绍不确定环境下的智能体感知和规划任务，然后引入动态系统的马尔科夫性质（Markov Property），并着重介绍几类马尔科夫模型，最后概括本文的主要工作和剩余章节的安排。

1.1　引言

在人工智能研究领域，智能体的概念试图为动态和/或随机环境中具有自主感知、规划和学习能力的智能系统的建模、设计和实现提供统一的框架[1]。智能体正在世界经济和人们的日常生活中扮演着越来越重要的角色。常见的智能体技术的应用包括智能机器人（Intelligent Robots）、网络爬虫（Web Crawlers）、手机助理软件（如苹果公司的 Siri）、火星探测器（Mars Rover）等。智能体必须能够自主地综合各种传感器提供的感知信息，更新*信息状态*（Information State），做出实时决策，发送行动指令，等待新的感知信息，并进入下一个决策周期。信息状态是指可以概况智能体与环境交互过程的所有历史信息（包括过去的所有行动和观察序列）的*统计充分量*（Sufficient Statistics）。智能体维护了当前的信息状态，就相当于维护了过去所有的历史信息，进一步就可以忘掉历史。简单来说，历史本身就可以作为信息状态。如果系统状态是完全可感知的，并且系统满足无记忆性，或马尔科夫性，即系统未来的*状态*（State）分布仅依赖于当前状态，而不依赖于任何过去的状态，那么就可以使用系统状态作为信息状态；如果系统状态不是完全可感知的，但系统状态满足马尔科夫性，那么可以引入状态空间上的概率分布作为信息状态——又称*信念状态*（Belief State）。本文重点关注满足马尔科夫性质的环境模型。作为例子，图 1.1展示了一个典型的马尔科夫系统中智能体与环境的交互模型[2]。图中，智能体从环境接受*观察*（Observation），估计环境状态（State Estimation），更新自身关于环境的信念状态，基于信念状态做出决策，并通过*行动/动作*（Action）改变环境状态。智能体所处的环境特征决定了感知和规划问题的复杂性。一般需要从以下几个维度

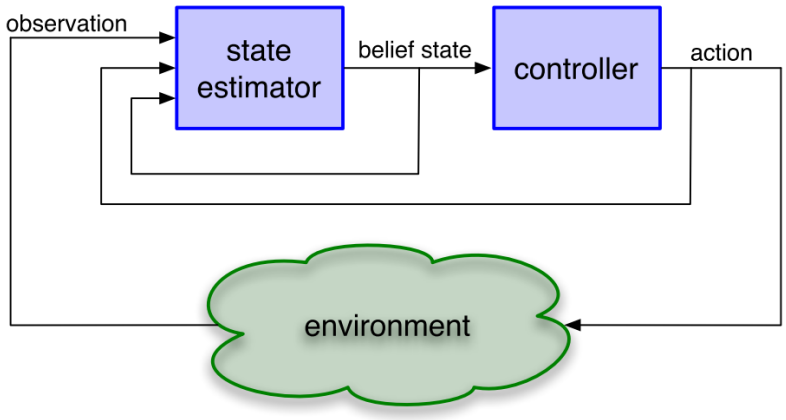


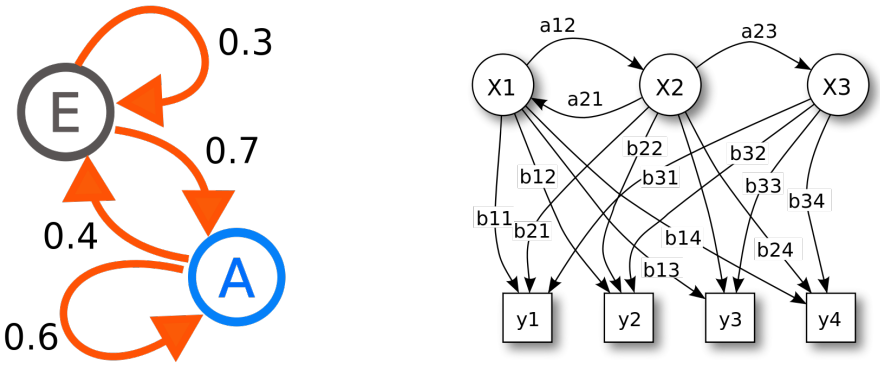
图 1.1　典型的智能体与环境的交互模型

来区分不同的环境：1. 完全可观察性和部分可观察性：完全可观察环境下智能体可以感知到环境的完整状态，部分可观察环境下智能体只能获得状态相关的观察，但不能重构出环境的完整状态；2. 确定性和不确定性：确定性环境的下一个状态是由当前状态和行动唯一确定的，不确定性环境的状态转移带有随机性；3. 周期性和非周期性：周期性环境下，智能体做决策时不仅要考虑当前状态，还要考虑未来情况，非周期性环境下，智能体只需要考虑当前状态；4. 静态性和动态性：静态环境下，环境模型是稳定并且不变的，动态环境下，环境模型可能会随着时间而改变；5. 离散性和连续性：离散环境具有有限多的状态数和动作数，连续环境下动作和/或状态的描述需要用到连续变量。很多现实问题中，来自传感器的感知信息不可避免会包含很多错误和误差；执行机构的执行结果也具有不可预知性，甚至失败的可能。同时，环境中也会存在给种各样的无法直接观察到的隐藏信息（Hidden Information）。这些不确定性为智能体的感知和规划带来了很大的挑战[3]。如何在观察、行动不确定性条件下，设计出高效、健壮的智能体感知和决策算法是当前人工智能研究的热点问题之一，也是本文关注的主要科学问题，具有十分重要的研究和应用前景。

智能体与环境成功交互的首要前提是感知环境，即通过综合和分析各种传感器数据以推测出系统的真实状态或真实状态的概率分布，这一过程常被称为状态估计或*信念更新*（Belief Update）。多传感器系统的感知任务通常又被称为*信息融合*（Information Fusion），或*传感器融合*（Sensor Fusion）[4]。由于测量手段的限制，智能体通常只能获得间接的测量数据；并且，测量数据往往不可避免带有各种噪音和误差。典型的例子比如摄像头可以获得成像范围内的逐像素色彩的测量值，但不排除会有很多噪点；雷达可以获得反射源的距离和角度测量值，但反射源并不一定就是真实目标；机器人路径规划时[5]，里程计（Odometry）可以获得机器人移动距离的测量值，但无法简单消除地面打滑的影响。满足马尔科夫性的观察不确定性环境下，信息更新的一般方法是建立状态空间上的观察模型，并根据*贝叶斯方法*（Bayesian Method）更新以获得信念状态。观察

模型给出特定状态下获得不同观察的概率分布。在很多情况下，还会遇到所谓*数据关联*（Data Association）的问题，即存在多个信息源时，需要弄清楚每一个匿名的测量值最可能源于哪个信息源[6]。比如，在*多对象跟踪*（Multi-Object Tracking，MOT）问题里面，需要确定空间相邻的若干测量值分别可能来自哪些潜在目标；移动机器人的同时定位与地图创建（Simultaneous Localization and Mapping, SLAM）问题里面，需要确定空间相邻的若干测量值分别来自哪些地标（Landmark）或地图特征点。理论上虽然可以维护所有数据关联的可能性，并计算出每一种可能性的概率，但可能性总数随时间和信息源的数目呈双指数增加，导致维护所有可能性的方法实际上不可解。实际问题中只能近似解决数据关联，一般的方法是在给定过去数据关联的假设之下，维护一个或多个当前最有可能的数据关联，并以此进行信息更新。观察不确定性，特别是隐藏信息的存在，给智能体感知带来了极大的挑战。本文主要关注不确定性感知和规划任务中，单个传感器的信念更新模型。

智能体完成信息感知任务，更新好信息状态后，便可以进一步进行*决策*（Decision Making）。通常来讲，每一步的决策不仅需要考虑当前的信息状态，还需要考虑未来可能的演变情况。智能体实际需要解决的是序列化决策问题（Sequential Decision-Making Problem），求解这一问题的一般过程被称为*规划*（Planning）[1]。某种意义上讲，*调度*（Scheduling）问题也可以被看成是规划问题的一个特例[7–9]。定义一个完整的规划问题一般要包含以下几个部分：1. 状态集合，即智能体和环境所有可能所处状态的描述；2. 动作集合，即智能体可以采取的行动集合；3. 状态转移函数，即下一个状态关于当前状态和当前状态上执行动作的函数；4. 目标函数，即智能体行动过程中需要优化的目标函数[10]。规划的总体目标是在任意当前状态下给出一组*控制策略*（Control Policy），使得智能体执行这个策略后产生的行动序列可以最终到达某些目标状态，或最大化某些*效用函数*（Utility Function）。形式上，控制策略就是状态到动作的映射。经典规划（Classical Planning）问题假设状态是完全可观察的，并且状态转移没有不确定性，一般可以简化成状态空间内的最短路径问题（Shortest Path Problem），问题的输入是一个带权的连通图：图上的节点表示状态，节点之间的边表示状态转移的路径，边上的权值表示节点转移的行动成本（Action Cost）；图上另外有一个作为当前状态的起点和若干作为目标状态的终点。规划的过程就是搜索到一条从起点到任意终点的路径，使得路径加权和最小。很多规划问题的状态空间都非常巨大，这时并不需要完整表示整个状态空间，而只需要在从当前状态可达的局部空间内进行搜索就可以了，可以一边搜索一边扩展局部状态空间，常见的搜索技术包括：深度优先搜索（Depth-First Search）、广度优先搜索（Breadth-First Search）、反向搜索（Backward Chaining）、启发式搜索（Heuristic Search）等。许多实际遇到的问题并不满足经典规划的基本假设，往往包含观察不确定性，以及行动不确定性：例如机器人手臂行动过程中具有不可预知的误



(a) 马尔科夫链　　(b) 隐马尔科夫模型

图 1.2　马尔科夫链和隐马尔科夫过程举例（图片来自 Wikipedia）

差，每一步的误差可能都很小，但累积误差可能会非常可观，导致末端执行器不能正确到达目标状态。这种情况下，机器人就需要主动考虑每一步的可能误差，以保证在行动过程中可以即使修正和抵消累积误差的影响。当然前提是机器人需要能够通过行动的反馈感知到系统的实时状态[11]。可见，不确定性环境下的规划问题势必要求能够同时处理不确定环境下的感知问题。值得一提的是，经典的强化学习（Reinforcement Learning）问题也可以转化成部分可观察环境下的规划问题[12]，一般称为基于模型的强化学习（Model Based Reinforcement Learning）或贝叶斯强化学习（Bayesian Reinforcement Learning，BRL）。强化学习致力于让智能体在跟环境交互的过程中最大化某一目标函数，即学习到完成某一任务的策略[13–18]。强化学习问题中智能体事先并没有完整环境模型，如动作转移函数等。如果把隐藏的环境模型看成状态的一部分，那么强化学习问题就可以转化成在环境模型和环境状态的联合状态空间中的规划问题，规划的目标是通过行动改变环境状态，并得到更精确的环境模型，做到*利用*（Exploitation）和*探索*（Exploration）的平衡，以最大化目标效用函数。利用是指尽量使用已有的次优环境模型进行最优决策，保证策略不至于太糟糕；探索是指为了能获得未来可能更高的回报，尽量尝试没有执行过的动作，或没有访问过的状态。利用和探索之间的权衡是强化学习的基本问题之一[19–22]，可以通过规划的手段进行最优控制。以*马尔科夫决策过程*（Markov Decision Process, MDP）[23] 和*部分可观察的马尔科夫决策过程*（Partially Observable Markov Decision Process, POMDP）[24] 为代表的马尔科夫决策理论（Markov Decision Theory）为解决满足马尔科夫性质的不确定性环境下感知，规划和学习问题提供了统一的理论框架和丰富的数学模型，也是本文的主要理论基础。

1.2　马尔科夫模型

在概率理论中，随机过程（Stochastic Process）是描述不确定性系统的主要数学工具[25]。给定状态空间 S，本文把一个随机过程定义成随机变量 $X \in S$ 关于时间的一个序列 $\{X_t \mid t \in T\}$，描述一个系统关于时间的变化情况，其中 $T = \{0, 1, 2, \ldots\}$ 是时间指标（Time Index）的集合。跟经典的微分方程（Differential Equation）描述方法相反，一个随机过程，即使初始状态 X_0 是已知和确定的，由于系统具有的不确定性，系统仍然可能会向不同方向演变，并且经常最终会有无限多种可能。本文主要关注满足马尔科夫性质的随机过程，通常被称为马尔科夫过程（Markov Process），定义在离散状态空间上的马尔科夫过程又被称为马尔科夫链（Markov Chain）。马尔科夫性由俄国数学家马尔科夫（Andrey Markov）于 1906 年提出[26]，一般形式化表示为：

$$\Pr(X_{t+1} \mid X_0, X_1, \ldots, X_t) = \Pr(X_{t+1} \mid X_t). \tag{1.1}$$

马尔科夫性的假设使得对一些原本难解的随机过程的推理和计算成为可能。一个简单的具有两个状态的马尔科夫链的例子见图 1.2(a)。对于这个马尔科夫链，任意时刻，如果系统处于状态 A，那么下个时刻，系统有 40% 的概率转移到状态 E，60% 的状态仍然处于状态 A；如果系统处于状态 E，那么下个时刻，系统有 70% 的概率转移到状态 A，30% 的概率仍然处于状态 E。

如果一个马尔科夫链的状态是部分可观察的（Partially Observable），则称这个马尔科夫链为隐马尔科夫模型（Hidden Markov Model, HMM）[27, 28]。隐马尔科夫模型的观察虽然是跟系统状态直接相关的，但观察本身不足以推断出系统的真实状态，一般还具有观察不确定性，即给定状态下只能按照一定的概率获得不同的观察。作为例子，一个具有 3 个状态和 4 个观察的隐马尔科夫模型见图 1.2(b)。图中，x 是系统状态，y 是观察，a 是状态转移概率，b 是观察模型。注意到，引入当前的状态分布为信念状态，则一个隐马尔科夫过程退化为一个定义在信念状态空间（Belief Space）上的连续马尔科夫过程。

引入智能体和智能体行动，一个受控的马尔科夫过程就成为一个马尔科夫决策过程（MDP）。MDP 中，状态转移概率不仅依赖于当前的状态，还依赖于作用于当前状态的动作，具有行动不确定性。求解一个 MDP 的过程即是求解得到一个最优控制策略以最大化某效用函数的期望值。类似地，一个受控的隐马尔科夫模型构成一个部分可观察马尔科夫决策过程（POMDP）。同理，POMDP 可以看成是信念空间上的连续 MDP。表 1.1简明总结了以上几种马尔科夫模型。

具体来说，MDP 建模了完全可观察、不确定性环境下，单个智能体跟环境交互的过程。智能体从环境获得输入，输出行动指令，并影响环境状态。MDP 模型假设虽然动作的执行效果具有不确定性，但环境的状态（包括智能体自身的状态）是完全可观察的。在每一个时间周期内，系统处于某一个完全可知的状态 s，假设智能体决策后采取的行动为 a，那么系统会按照一定的概率分布转移到

表 1.1　几种马尔科夫模型分类

	系统状态完全可观察	系统状态部分可观察
系统是自主的	马尔科夫链（Markov Chain）	隐马尔科夫模型（HMM）
系统是受控的	马尔科夫决策过程（MDP）	部分可观察马尔科夫决策过程（POMDP）

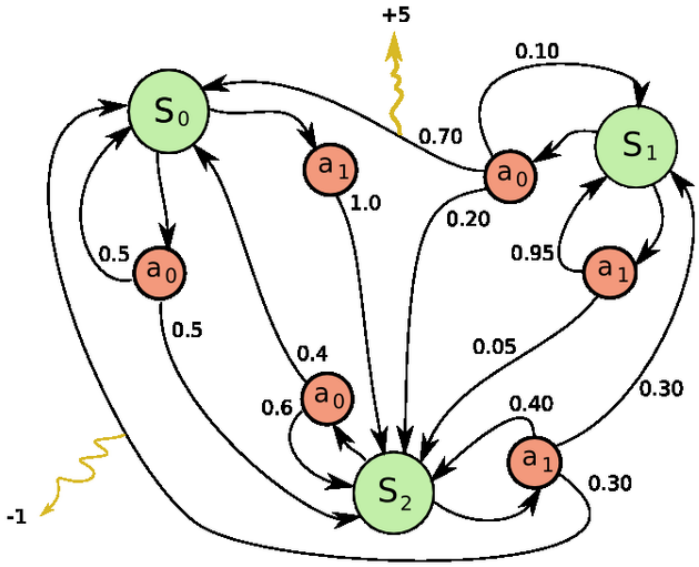


图 1.3　马尔科夫决策过程举例（图片来自 Wikipedia）

下一个状态 s'，同时智能体获得一个立即回报值（Immediate Reward）$R(s, a, s')$。系统转移到下一个状态的概率是由当前状态和当前智能体采取的行动一起决定的，形式上，这个概率由状态转移函数 $\Pr(s' \mid s, a)$ 给出。MDP 假设，给定状态 s 和行动 a，系统转移到某一状态 s' 的概率跟过去的状态和动作是条件独立的，即系统满足马尔科夫性。智能体的决策目标是优化总的规划时限（Planning Horizon）内的累积回报值（Cumulative Reward）的期望值。规划时限可以是有限的也可以是无限的。MDP 模型在机器人、自动控制、经济学和制造业等各种优化问题（Optimization Problem）领域有着广泛的应用[29]。MDP 的求解方法一般包括动态规划（Dynamic Programming）、强化学习、蒙特卡洛（Monte-Carlo）仿真等[30]。已经被证明 MDP 问题的复杂度是 P[31]，即存在多项式时间的最优算法。但由于状态空间的大小关于状态变量的数目呈指数增加，很多大规模 MDP 问题的直接求解仍然是非常困难的，常用的近似技术包括在线求解、分层分解等。作为例子，图 1.3展示了一个包含三个状态两个动作的简单 MDP 模型。图中，S 表示状态，a 表示动作，边上的数字为状态转移概率，$+5$ 和 -1 表示动作执行后的立即回报值。

POMDP 模型将 MDP 模型扩展到更一般的部分可观察、不确定性环境中。

POMDP 假设系统的动态模型（System Dynamics）由一个底层 MDP 完全决定，这个 MDP 模型是已知的，但智能体不能直接观测到系统状态。相反，智能体只能获得状态相关的若干观测值，为了进行有效决策，智能体必须要根据历史观察、行动序列来维护系统所处不同状态的概率分布——即信念状态。当前的信念状态提供了跟过去所有历史行动和观察等价的信息。如果把信念状态看成新的状态，那么 POMDP 问题可以转化称信念空间上的连续 MDP 问题，转化后的问题又称为信念 MDP（Belief MDP）问题。求解信念 MDP 跟求解一般 MDP 没有本质区别，需要注意的一点是信念 MDP 中的状态转移不能简单的由环境模型给出，而是需要进行信念更新：即给定当前信念状态和智能体执行的动作，计算出系统下一个信念状态的概率分布。信念更新一般按照贝叶斯方法进行。信念 MDP 上的动态规划就可以精确求解原 POMDP 问题。POMDP 模型可以用来广泛建模很多现实世界中的时序决策问题，常见的应用包括机器人感知和规划、人机对话系统、传感器管理等。理论上证明，POMDP 问题的复杂度是 PSPACE。由于状态空间随状态变量的数目呈指数增加，因此 POMDP 问题的复杂度随状态变量数目是呈双指数增加的。对于一般 POMDP 问题的精确求解是非常困难的，近似求解的技术一般包括基于信念点的算法、启发式搜索、蒙特卡洛仿真等。POMDP 的一大优势是可以在统一的回报函数框架内，自动处理感知动作和一般动作之间的权衡。在状态空间里面，一般的动作（比如机器人底盘移动、手臂抓取物体等）会改变环境状态，但感知动作（比如视觉观察、激光扫描等）不会改变环境状态，但在信念空间里面，这两类动作实际上没有区别，他们都起到改变信念状态的作用：一般动作改变了系统所处不同物理状态的主观概率，感知动作改变的也正是同一概率分布。以一个机器人在房间内搜寻某个物品为例，假设视觉模块告知决策模块在某一个地方识别到目标，但并不能 100% 确信。这种情况下，POMDP 可以让机器人以总体成本最优的方式做出当前是应该继续在原地多观察一段时间，还是向视觉提供的目标位置进行移动的决策。如果用完成任务的总体时间来衡量回报/成本，那么决策的关键就取决于，观察的时间消耗和移动的时间消耗，信念状态不同的情况下，最优决策是不一样的：对视觉模块提供的识别结果比较确信的情况下，直接移动到目标会有很高的概率提前完成任务；对目标不太确信的情况下，多观察一会是更明智的选择。

MDP 和 POMDP 建模的都是环境中只有一个智能体的情况，很多实际问题涉及到**多智能体系统**（Multi-Agent Systems）[32, 33]。多智能体系统中，多个独立的智能体共同与环境进行交互，通过合作（或对抗）的方式共同参与完成某一任务。每一个智能体都是一个独立的决策实体，分别从环境中获得局部观察，各自进行决策并行动。智能体一起通过**联合行动**（Joint Action）来改变环境状态。为了成功完成任务，每个智能体做决策时必须考虑到其他智能体可能采取的行动，以及所有智能体的联合行动对环境的影响。常见的多智能体系统有机器人足球、传感器网络、人—机器人交互系统等。把 MDP 和 POMDP 扩展到多智能

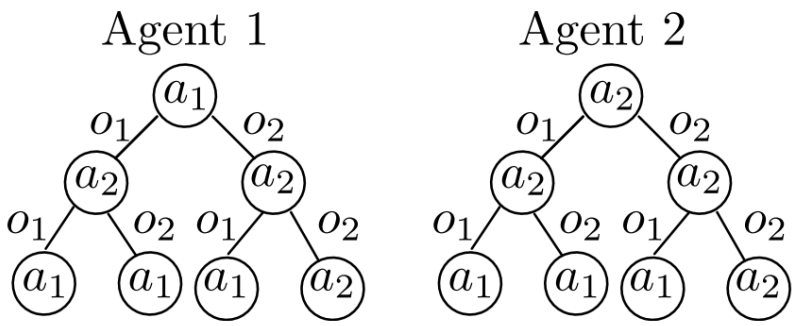


图 1.4　两个智能体的策略树举例

体系统的一个自然方案是把其他智能体都看成是环境的一部分，并假设其他智能体执行的是固定策略，这样就可以明确给出联合状态和联合行动的联合转移函数，多智能体问题退化成单个智能体的问题。这样处理的明显缺点是不能很好的适应其他智能体策略改变的情况，也不能处理嵌套信念（Nested Belief）的问题。所谓嵌套信念是指智能体 A 在思考智能体 B 可能行动的同时，智能体 B 可能也在思考智能体 A 在怎样思考智能体 B 的可能行动，为此智能体 A 就需要进一步思考智能体 B 可能在怎样思考智能体 A 在怎样思考智能体 B 的可能行动，这种嵌套可以是无限递归下去的。一般需要假设一个最大嵌套深度，但这个最大嵌套深度一般是没有办法事先确定的。并且不同的最大嵌套深度假设有可能导致最优策略差异性非常大，事实上成为了系统不稳定的潜在因素。典型的例子，比如经典的双人石头—剪刀—布的游戏中，假设的最大嵌套深度每增加一层，智能体的最优策略都不一样。交互式 POMDP（I-POMDP）[34] 就是按照这个思路来处理多智能体系统的规划问题的，由于不好处理信念嵌套的问题，所以 I-POMDP 本身的求解非常困难，通常只存在理论上的意义。另外一个常见的扩展方案是每个智能体都意识到其他智能体也是自主决策实体，而不把其他智能体归入环境。智能体自身决策时，充分考虑所有其他智能体可能的行动，在联合状态和联合动作空间内进行“集中式”的决策，选择出最优联合动作后，分布式执行各自的行动。分布式马尔科夫决策过程（Decentralized MDP, DEC-MDP）和分布式部分可观察马尔科夫决策过程（Decentralized POMDP, DEC-POMDP）就是此类扩展思路的典型代表[35]。以 DEC-POMDP 模型为例，常用的求解方案是在策略空间（Policy Space）内进行搜索，每个智能体独立构建包括其他智能体在内的联合策略树（Decision Tree）。策略树的节点表示动作，边表示可能获得的观察，策略树的深度即为规划时限。所有智能体策略树的集合给出了联合动作、观察历史情况下每个智能体应该执行的动作。图 1.4给出了两个智能体 2 步规划的联合策略树的一个例子[36]。图中，α 是动作节点，o 是观察。构建策略树的常见技术包括动态规划、启发式搜索等。DEC-POMDP 模型的潜在的问题是，联合的状态、动作和观察空间规模非常巨大，关于空间维度呈多重指数增

加。给定底层模型后，DEC-POMDP 的计算复杂度是 NEXP 的，关于规划时限和智能体数目呈双指数增加。完全按照 DEC-POMDP 模型求解哪怕只有两个智能体的问题也是非常困难的。本文重点关注单个智能体的规划和感知问题，遇到多智能体系统时，通过假设其固定策略给出联合转移函数。

1.3　论文的内容与组织结构

本文以马尔科夫决策理论（MDP 和 POMDP）为主要理论依据，着重解决在大规模不确定性环境下的感知和规划问题。特别地，本文提出一个新颖的基于 MAXQ[37] 分层分解的大规模 MDP 问题在线规划算法——MAXQ-OP[38–41]；以 Thompson 采样[42] 为动作选择策略的 MDP 和 POMDP 蒙特卡洛在线规划算法——DNG-MCTS 和 D²NG-POMCP[43, 44]；以及，人机交互领域中，基于集合粒子滤波的多对象跟踪算法 PFS（Particle Filtering over Sets，PFS）[45]。主要研究平台除了包括各种标准测试问题和数据集，还包括 RoboCup 仿真 2D 机器人足球比赛平台[46]。* 以下是剩余章节的主要内容安排。

第二章给出相关马尔科夫决策基础理论的调研及综述。分别介绍了 MDP 和 POMDP 问题的数学模型和常见求解算法。第三章到第五章介绍了本文的主要工作。每个工作自成一章，首先给出该工作的主要动机，然后比较相关工作，进一步详细介绍理论和算法的技术细节，最后给出实验结果。第六章总结全文工作，并讨论下一步研究工作。本文的主要工作部分内容安排如下：

第三章提出了一个大规模 MDP 问题的分层在线规划算法，即 MAXQ-OP。MAXQ-OP 利用原始问题本身所具有的分层结构，根据 MAXQ 值函数分解理论把原始问题分解成一系列的小问题，在线规划过程中同时求解这些小问题，最终完成原始问题的求解。MAXQ-OP 得益于分层结构上的时序抽象、状态抽象和子任务共享，比直接求解原始问题高效。创新点是通过提出终止概率和完成函数的近似计算方法，把分层理论应用到在线规划算法中，从而可以求解比现有文献范围内规模更大的问题，成功应用到科大“蓝鹰”仿真机器人足球队中。

第四章展示了新颖的基于后验动作采样的贝叶斯蒙特卡洛树搜索（Monte-Carlo Tree Search, MCTS）算法——DNG-MCTS 和 D²NG-POMCP。该工作使用正态分布的混合分布建模蒙特卡洛搜索树上的长期回报的未知分布，使用 Dirichlet-NormalGamma 混合分布作为其共轭分布，根据贝叶斯推理理论对其进行后验更新，设计出基于后验动作采样的蒙特卡洛在线 MDP 规划算法 DNG-MCTS，证明其收敛性，并在 MDP 标准测试问题上取得了比本领域内先进算法（UCT 相关算法）更好的实验结果。创新点是根据马尔科夫链上的中心极限定理，推导出来蒙特卡洛搜索树上的长期回报的近似分布，从而可以使用贝叶斯方法对其进行建模和推理，进一步通过使用 Thompson 采样方法达到较好的实验结果。同时，本文进一步将提出的 DNG-MCTS 算法推广到 POMDP 问题，使

*更多关于 RoboCup 2D 的信息见官方 Wiki：http://wiki.robocup.org/wiki/Soccer_Simulation_League。

用 Dirichlet-Dirichlet-NormalGamma 混合分布作为长期回报分布的共轭分布，使用贝叶斯方法对其进行建模和更新，设计出基于后验动作采样的蒙特卡洛在线 POMDP 规划算法 D²NG-POMCP，在 POMDP 标准测试问题上取得了比本领域内先进算法（POMCP 算法）更好的实验结果。创新点是提出把环境状态和智能体的信念状态看成一个联合状态，从而可以把给定策略的 POMDP 问题转化成一个马尔科夫链，在这个马尔科夫链上根据中心极限定理，推导出蒙特卡洛搜索树上长期回报分布的近似分布，从而可以使用贝叶斯方法对其进行建模和推理，进一步通过使用 Thompson 采样方法达到较好的实验结果。

第五章提出了集合粒子滤波的多对象跟踪算法——PFS。多人识别和跟踪对成功的社会化人—机器人交互至关重要。本文提出集合空间上的粒子滤波算法及其相关技术来处理这一问题，主要贡献有：1. 关联分配、误报检测和漏报检测的剪枝算法；2. 基于数据关联的粒子改进策略；3. 基于贝叶斯的运动和提议权值的概率密度估计方法；4. 基于期望值最大化（Expectation-Maximization, EM）的个体确认算法。传统多对象跟踪算法总是假设一个或多个数据关联，使用假设的数据关联来分别独立地更新目标。对于此类方法而言，如果某个周期的假设跟实际不相符，那个周期的更新就会出错，并且此后很难再恢复。本文提出的方法使用集合来表示联合状态和联合观察，直接在集合空间上进行观察似然性（Observation Likelihood）的计算，避免了直接进行数据关联的步骤。集合的表示方法编码了多目标系统的所有状态信息，包括目标个数、目标状态以及所有可能的潜在数据关联方式。在流行的 PETS2009[47] 数据集上的测试表明，算法的整体效果优于目前最领先的算法，并且仍然保持了可以在线运行的良好特征，所以可以直接应用到人—机器人交互领域。CoBot 实体机器人的实验进一步验证了 PFS 算法的有效性。

本文主要的三个工作相互对立，但又彼此联系。他们在一起从不同侧面探讨了智能体在不确定性环境下的自动感知和规划任务所需要考虑的所有要素，为智能感知和规划任务提供了基于马尔科夫理论的原理性解决方案。本文工作在实际问题中得到了验证和检验，证明了有效性，具有很强的扩展性，有望应用到规模特别巨大的复杂实际问题中。

第二章　马尔科夫决策理论

内容提要

马尔科夫决策理论为本文的主要工作提供了理论基础。本章详细介绍马尔科夫决策理论及常用求解算法，特别地包括马尔科夫决策过程（MDP）和部分可观察马尔科夫决策过程（POMDP）。算法部分主要从离线求解算法和在线求解算法两方面来介绍。MDP 离线规划算法部分主要介绍基于动态规划的策略迭代和值迭代算法，在线规划算法部分主要介绍与或树搜索算法、实时动态规划算法和蒙特卡洛树搜索算法；类似地，POMDP 离线规划算法部分主要介绍传统的基于动态规划的值迭代算法、QMDP 算法和基于信念点的近似算法，在线规划算法部分主要介绍启发式搜索、实时动态规划算法以及蒙特卡洛树搜索算法。本章还介绍了半马尔科夫过程，以及 MDP 分层分解的一般技术。

2.1　马尔科夫决策过程基本模型

马尔科夫决策过程为解决不确定性环境下单个智能体规划问题提供了基本的理论模型。MDP 假设环境状态满足马尔科夫性质，状态转移虽具有不确定性，但环境状态是完全可观察的。MDP 框架下，单个智能体周期性地跟环境交互，从环境获得完整的状态观察，决策以选择一个行动作用于当前状态后，环境转移到下一个状态，并进入新的决策周期。

定义 2.1.1 (马尔科夫决策过程). *马尔科夫决策过程形式上可以被定义成一个四元组* $\langle S, A, T, R \rangle$, *其中：*

- S *是状态空间（State Space），即所有可能环境状态的集合；*
- A *是动作空间（Action Space），即智能体所有可选行动的集合；*
- $T : S \times A \times S \to [0, 1]$ *是状态转移函数（Transition Function），* $T(s' \mid s, a) = \Pr(s' \mid s, a)$ *给出状态* s *下执行动作* a *后，系统转移到状态* s' *的概率；*
- $R : S \times A \to \mathbb{R}$ *是回报函数（Reward Function），* $R(s, a)$ *给出智能体在状态* s *下执行动作* a *后从环境立即获得的回报值（又称奖赏值）。*

定义2.1.1中，MDP 的状态空间 S 和动作空间 A 可以是离散的也可以是连续的，如无特殊说明，本文认为状态空间 S 和动作空间 A 都是离散的。通常有两种方式来描述状态，即*平铺表示*（Flat Representation）和*因子化表示*（Factored Representation）[48]。平铺表示把每个状态进行编号，以 $s_1, s_2, s_3, \ldots$ 的方式索引每一个状态。平铺表示下，状态转移函数一般以查表法（Tabular Method）给出，即对每个 s—a—s' 的组合直接存储其状态转移的概率 $T(s' \mid s, a)$。因此，存储完整的转移函数需要 $|S| \times |A| \times |S|$ 大小的存储空间。因子化表示使用一个高维空间中的特征向量来描述每一个状态，即 $s = [x_1, x_2, \ldots, x_n]$，其中 n 为状态空

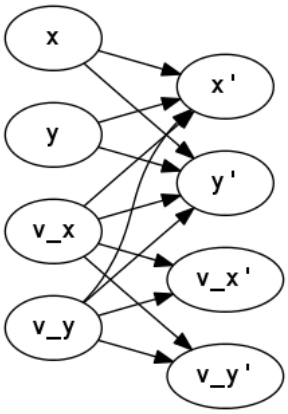


图 2.1　平面运动物体状态转移 DBN 举例

间的维度，x_i $(1 \leqslant i \leqslant n)$ 为状态变量。每个状态变量在各自定义范围内取值，枚举所有取值即表达了整个状态空间。举例来说，一个平面上运动物体的状态可以表示为 $[x, y, \dot{x}, \dot{y}]$，其中 (x, y) 为位置，$(\dot{x}, \dot{y})$ 为速度。采用因子化表示时，状态转移函数和回报函数可以自然设计成数学函数的形式。以状态转移函数为例，输入 s，a 和 s'，函数内部使用状态变量进行数值计算，并输出状态转移概率 $T(s' \mid s, a)$。状态变量之间一般存在各种条件独立的关系，所以又可以进一步结合使用*动态贝叶斯网络*（Dynamic Bayesian Network, DBN）[49] 来表达状态转移函数，每个动作对应于一个 DBN。DBN 是一个两层的有向无环图，第一层节点表示状态 s，第二层节点表示状态 s'，节点之间的边表示状态变量之间存在的因果关系，没有边连接的节点互相条件独立。令 $\mathbf{u}_i$ 为状态 s 中跟状态变量 x_i 不独立的状态变量的集合，那么状态转移函数可以进一步分解为：

$$T(s' \mid s, a) = \prod_{1 \leqslant i \leqslant n} \Pr(x_i \mid \mathbf{u}_i, a). \tag{2.1}$$

可见，相比于平铺表示，因子化表示是一种更为高效和紧凑的表示方法，并且可以利用状态变量条件独立性等领域知识分解状态转移函数。图 2.1给出了平面运动物体状态转移 DBN 的一个例子。图中，(x, y) 为位置，(v_x, v_y) 为速度。对于平面运动动作而言，新的位置跟当前的所有状态变量都相关，但新的速度只跟当前速度相关。

智能体跟环境交互的广义控制策略可以看成是状态空间到动作空间上概率分布的映射。数学上，策略 π 可以表示成 $\pi : S \times A \to [0, 1]$，$\pi(s, a)$ 给出智能体在状态 s 下执行动作 a 的概率。广义的控制策略是*随机策略*，策略 π 总是按照一定的概率分布来选择动作，所以在相同状态下可能会执行不同的动作。如果策略 π 在任意状态 s 上总是选择一个确定的动作，则称该策略为*确定性策略*，数学上表示为 $\pi : S \times A \to \{0, 1\}$。确定性策略也可以看成是状态空间到动作空间的映射，即 $\pi : S \to A$，其中 $\pi(s)$ 给出智能体在状态 s 下应该执行的动作。如无特别说明，本文主要考虑确定性策略。

给定策略 π 和初始状态 s_0，智能体服从（Follow）策略 π 与环境进行交互，假设智能体的规划时限是 H，策略 π 在状态 s_0 下的效用函数 $U(\pi \mid s_0)$ 定义为智能体未来 H 步内期望获得的累积回报值：

$$U(\pi \mid s_0) = E\left[\sum_{0 \leqslant t < H} R(s_t, \pi(s_t))\right], \tag{2.2}$$

其中，$R(s_t, \pi(s_t))$ 是智能体在第 t 步获得的立即回报。无限规划时限的情况下，引入折扣因子 $\gamma \in (0, 1]$，以保证效用值 $U(\pi \mid s_0)$ 能够收敛，无限规划时限的效用函数定义如下：

$$U(\pi \mid s_0) = E\left[\sum_{t \geqslant 0} \gamma^t R(s_t, \pi(s_t))\right]. \tag{2.3}$$

可见，公式 2.3的定义下，越早获得的回报对最终效用值的贡献就越大。另一方面，折扣因子越大，未来的回报对当前决策的影响就越大。这个效用值也可以看成智能体在每步交互过程都有 $1-\gamma$ 的概率终止的情况下可以获得的无折扣期望累积回报。在有限规划时限的情况下也可以引入折扣因子 $\gamma = 1$ 而不失一般性。容易看出，无论是有限规划时限还是无限规划时限情况下，如果 $\gamma \neq 1$，效用函数都有上界 $R_{max}/(1-\gamma)$，其中 R_{max} 是智能体单步可获得立即回报的最大值。求解一个 MDP 问题的目标就是找到一个最优策略（Optimal Policy）可以最大化效用函数[50, 51]，记最优策略为 π^*，则：

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, U(\pi \mid s_0). \tag{2.4}$$

如果确定性策略 π 每次遇到状态 s 总是选择相同的动作，则称策略 π 是稳定策略。相反，不稳定策略在状态 s 下选择的动作还跟当前的时间 t 有关，一般引入时间作为下标来索引在某具体时刻的策略：策略 π_t 即智能体在还剩 t 步规划时限时（即当前是第 $H-t$ 步，对应的状态为 s_{H-t}）所执行的策略。不稳定策略可以表示为不同时刻具体策略的集合 $\pi = \{\pi_H, \pi_{H-1}, \ldots, \pi_1\}$。有限规划时限问题的最优策略一般是不稳定的，智能体在生命周期最后一步选择动作的方式往往跟在生命周期的第一步选择动作的方式很不一样。举例来说，机器人足球比赛中，假设智能体所在的队伍目前领先，那么在相同的状态下，智能体更倾向于“保守”才是最优策略；相反，如果对方领先，那么在相同状态下，应该更倾向于“激进”策略。无限规划时限问题里面，智能体在任意时刻，总是有恒定的剩余期望时间，所以智能体无需关于时间改变自己的策略，也就是说，无限规划时限问题具有稳定的最优策略。

在有限规划时限情况下，假设智能体服从策略 $\pi = \{\pi_H, \pi_{H-1}, \ldots, \pi_1\}$，定义智能体在还剩 t 步规划时限情况下，从状态 s 开始可以获得的期望累积回报值为策略 π 的值函数（Value Function），记为 $V_t^\pi(s)$。显然，$V_1^\pi(s) = R(s, \pi_1(s))$，即智能体在生命周期最后一步可以获得的期望累积回报就是该步可以获得的期

望立即回报。进一步，定义智能体在还剩 t 步规划时限情况下，从状态 s 开始，首先执行动作 a，然后服从策略 π，可以获得的期望累积回报值为策略 π 的*行动值函数*，记为 $Q_t^\pi(s,a)$，则：

$$Q_t^\pi(s,a) = R(s,a) + \gamma \sum_{s' \in S} T(s' \mid s, \pi_t(s)) V_{t-1}^\pi(s'). \tag{2.5}$$

也就是说，状态 s 下首先执行动作 a，再服从策略 π，可以获得的期望累积回报值为本周期的立即回报值，$R(s,a)$，与乘以折扣因子后的 $t-1$ 步值函数关于状态转移概率的期望值之和。注意到，为了正确评估未来情况，必须根据状态转移函数 $T(s' \mid s, \pi_t(s))$ 来考虑所有可能遇到的下一个状态 s'。所以，值函数可以用行动值函数表示为：

$$V_t^\pi(s) = Q_t^\pi(s, \pi_t(s)). \tag{2.6}$$

递归求解上式就得到了一组不同剩余规划时限下的值函数。有限规划时限情况下的值函数最终可以表示成它们的集合，即 $V^\pi = \{V_H^\pi, V_{H-1}^\pi, \ldots, V_1^\pi\}$。给定值函数 V_H^π，策略 π 在初始状态 s_0 下的效用函数可以用值函数表示成：

$$U(\pi \mid s_0) = V_H^\pi(s_0). \tag{2.7}$$

在无限规划时限情况下，稳定策略 π 在状态 s 下的值函数 $V^\pi(s)$ 定义为智能体从状态 s 开始执行策略 π 所能获得的期望累积回报值。类似地，定义智能体在状态 s 下首先执行动作 a，然后服从策略 π 所能获得的期望累积回报值为行动值函数 $Q^\pi(s,a)$，即：

$$Q^\pi(s,a) = R(s,a) + \gamma \sum_{s' \in S} T(s' \mid s, \pi_t(s)) V^\pi(s'). \tag{2.8}$$

同理，$V^\pi(s)$ 可以递归表示成：

$$V^\pi(s) = Q^\pi(s, \pi(s)). \tag{2.9}$$

公式 2.9实际上是一组关于 $V^\pi(s)$ 的线性方程。该方程组有唯一解，即为 V^π。类似地，有限规划时限情况下，策略 π 在初始状态 s_0 下的效用函数用值函数表示为：

$$U(\pi \mid s_0) = V^\pi(s_0). \tag{2.10}$$

至此，无论是在有限规划时限还是无限规划时限情况下，效用函数的完整计算方式已经由公式 2.7和公式 2.10分别给出。令 V^* 和 Q^* 分别为最优策略 π^* 对应的值函数和行动值函数，那么根据*贝尔曼最优性等式*（Bellman Optimality Equation）[52]，最优值函数 V^* 和最优行动值函数 Q^*，在有限规划时限和无限规划时限情况下，分别满足：

$$V_t^*(s) = \max_{a \in A} Q_t^*(s,a), \tag{2.11}$$

和

$$V^*(s) = \max_{a \in \mathcal{A}} Q^*(s, a). \tag{2.12}$$

展开后，得到：

$$V_t^*(s) = \max_{a \in \mathcal{A}} \left\{ R(s, a) + \gamma \sum_{s' \in S} T(s' \mid s, a) V_{t-1}^*(s') \right\}, \tag{2.13}$$

和

$$V^*(s) = \max_{a \in \mathcal{A}} \left\{ R(s, a) + \gamma \sum_{s' \in S} T(s' \mid s, a) V^*(s') \right\}. \tag{2.14}$$

从另一个角度说，如果把公式 2.13和公式 2.14看成方程组，那么它们的解分别就是有限规划时限和无限规划时限情况下的最优值函数 V^*。给定最优值函数 V^*，有限规划时限和无限规划时限情况下，最优策略 π^* 可以分别由以下公式得出：

$$\pi_t^*(s) = \operatorname*{argmax}_{a \in \mathcal{A}} V_t^*(s), \tag{2.15}$$

和

$$\pi^*(s) = \operatorname*{argmax}_{a \in \mathcal{A}} V^*(s). \tag{2.16}$$

可以证明，公式 2.15和公式 2.16分别是公式 2.4在有限规划时限和无限规划时限情况下的解。所以精确求解贝尔曼最优性等式就可以找到 MDP 问题的最优值函数和最优策略。然而，由于计算复杂度方面的原因，精确求解一个 MDP 往往是不可行的，实际应用中也并不一定需要找到最优策略。很多情况下，*近似最优策略*（Near-Optimal Policy）就可以满足问题的需求。如果一个策略 π 对于所有状态 s 满足 $V^*(s) - V^\pi(s) \leqslant \epsilon$，则称策略 π 为原问题的 ϵ-最优策略。

有限规划时限问题直接迭代求解贝尔曼最优性等式容易找到最优解，即由 V_1^π 得出 V_2^π，进而得出 V_3^π，……，一直到 V_H^π 终止。无限规划时限问题的最优值函数原理上可以通过持续增加一个有限规划时限问题的规划时限来逐步逼近，即当 $H \to \infty$ 时，$V_H^* \to V^*$。然而，无论是有限规划时限还是无限规划时限问题，盲目地迭代只会浪费计算资源，如何在收敛的情况下，尽快找到收敛后的解才是求解 MDP 问题的关键。

值得一提的是，作为规划和学习问题基本构成部分的*多臂赌博机*（Multi-Armed Bandit，MAB）问题可以被看成是 MDP 问题的特例。该 MDP 问题只有一个状态 s，同时有一个随机的回报函数 $R(s, a) := X_a$，其中 X_a 是一个具有未知分布 $f_{X_a}(x)$ 的随机变量。在每一个时间周期 t 内，智能体必须选择一个动作 a，随后获得一个随机的回报值 X_{a_t}。MAB 问题的策略是行动—回报历史到行动的映射。通常情况下，求解 MAB 问题的目标是找到一个能够平衡探索和利用的最优策略以最小化*累积剩余值*（Cumulative Regret，CR），定义为：

$$R_T = E\left[\sum_{t=1}^{T} (X_{a^*} - X_{a_t}) \right], \tag{2.17}$$

其中 a^* 是实际未知的最优动作。另外，对于一个纯探索性策略，定义简单剩余值（Simple Regret，SR）如下：

$$r_n = \mathrm{E}\left[X_{a^*} - X_{\bar{a}}\right], \tag{2.18}$$

其中 $\bar{a} = \mathrm{argmax}_{a \in A} \bar{X}_a$ 是具有最大实验平均回报值的动作。

以上给出的定义和模型都是基于回报模型的，文献中还会经常看见一类 MDP 的定义是基于成本的模型，也就是说，成本函数 $c(s,a)$ 取代了回报函数 $R(s,a)$，min 操作取代了 max 操作。类似地，求解基于成本模型的 MDP 等价于找到一个策略可以最小化期望累积成本。基于回报模型的 MDP 算法只要把贝尔曼更新公式中的 max 替换成 min 就可以适用于基于成本模型的 MDP。如无特殊说明，本文主要考虑基于回报模型的 MDP 问题。

2.2　马尔科夫决策过程求解算法

通过直接求解贝尔曼最优性等式来求解 MDP 等随机最优控制（Stochastic Optimal Control）问题的方法一般被称为动态规划（Dynamic Programming）。动态规划方法面临所谓“维度诅咒”（The Curse of Dimensionality）的问题，即其计算复杂度关于状态空间的维度呈指数增加，但动态规划方法仍然是目前最广泛使用的通用求解算法。事实上，几乎所有的算法都可以被看成是在某种程度上试图以更小的计算资源消耗或不完整的环境模型来近似实现动态规划算法的效果。本节将从离线和在线两个方面分别介绍 MDP 问题求解算法。离线（Offline）算法是指算法事先计算出完整策略，智能体获得策略后，只需要在与环境交互过程中执行策略即可，无需更多计算；在线（Online）算法是指，智能体事先不进行或只进行不完整的离线计算，智能体与环境交互过程中，每个决策周期内实时计算当前状态下应该执行的最优或近似最优动作。

2.2.1　离线求解算法

离线规划算法遍历整个状态空间，计算出完整策略供智能体与环境交互时执行。本节以无限规划时限问题为例主要介绍经典的策略迭代（Policy Iteration）和值迭代（Value Iteration）算法。

2.2.1.1　策略迭代

策略迭代算法交替进行策略评估（Policy Evaluation）和策略改进（Policy Improvement）两个步骤。策略评估是指给定任意策略 π 计算出其值函数 V^π 的过程。策略评估可以直接根据值函数的定义进行迭代计算，即任意初始化值函数 v，然后不断根据公式 2.9重新计算得到新的值函数 v，直至 v 收敛，此时 $V^\pi = v$。策略改进是指给定某策略 π 的值函数 V^π，改进原策略得到一个更好

Input: An MDP $\langle S, A, T, R \rangle$, and a small positive number ϵ
Output: A near-optimal policy π
Initialize $\pi(s) \in A$ arbitrarily for all $s \in S$
repeat
  repeat
    $\Delta \leftarrow 0$
    foreach $s \in S$ **do**
      $V'(s) \leftarrow R(s, \pi(s)) + \lambda \sum_{s' \in S} T(s' \mid s, a) V(s')$
      $\Delta \leftarrow \max\{\Delta, |V(s) - V'(s)|\}$
      $V(s) \leftarrow V'(s)$
    end
  until $\Delta < \epsilon$
  converged $\leftarrow$ True
  foreach $s \in S$ **do**
    $\pi'(s) \leftarrow \operatorname{argmax}_{a \in A} \left\{ R(s, a) + \lambda \sum_{s' \in S} T(s' \mid s, a) V(s') \right\}$
    if $\pi'(s) \neq \pi(s)$ **then**
      converged $\leftarrow$ False
    end
    $\pi(s) \leftarrow \pi'(s)$
  end
until converged $=$ True
return π

算法 2.1: 无限规划时限 MDP 策略迭代算法

的策略 π' 的过程。一个简单的改进策略是使用*贪心算法*（Greedy Algorithm）：$\pi'(s) = \operatorname{argmax}_{a \in A} Q^{\pi}(s, a)$，其中 Q^{π} 是策略 π 的行动值函数。容易证明，$V^{\pi'}(s) \geqslant V^{\pi}(s)$ 对所有状态 s 都成立，可见策略 π' 确实是一个改进后的策略。如果交替进行策略评估和策略改进步骤，直到策略收敛，那么收敛的策略就是需要寻找的最优策略，即：

$$\pi_0 \xrightarrow{PE} V^{\pi_0} \xrightarrow{PI} \pi_1 \xrightarrow{PE} V^{\pi_1} \xrightarrow{PI} \pi_2 \xrightarrow{PE} \dots \xrightarrow{PI} \pi^* \xrightarrow{PE} V^*, \tag{2.19}$$

其中 $\xrightarrow{PE}$ 表示策略评估操作，$\xrightarrow{PI}$ 表示策略改进操作。每一次迭代步骤计算得到的新策略总是比前一个策略好，由于 MDP 的策略总数 $|S|^{|A|}$ 是有限的，所以策略迭代可以在有限步以内找到最优策略和最优值函数[53]。算法 2.1给出了完整的策略迭代算法。该算法接受一个完整的 MDP 模型和一个足够小正数 ϵ 作为输入，任意初始化一个起始策略，迭代地进行近似策略评估和策略改进直到策略收敛，然后输出收敛后的策略。

2.2.1.2　值迭代

策略迭代算法一般情况下只需要很少的迭代步骤就可以很快收敛，但策略迭代算法的一个缺点是，每次迭代步骤需要花费额外时间等待策略评估操作收敛。如果把策略评估操作分散到策略改进操作中，有可能获得能更快收敛的算

Input: An MDP $\langle S, A, T, R \rangle$, and a small positive number ϵ
Output: A near-optimal policy π
Let $V(s) \leftarrow 0$ for all $s \in S$
repeat
  $\Delta \leftarrow 0$
  foreach $s \in S$ **do**
    foreach $a \in A$ **do**
      $Q(s,a) \leftarrow R(s,a) + \lambda \sum_{s' \in S} T(s' \mid s,a) V(s')$
    end
    $\pi(s) \leftarrow \operatorname{argmax}_{a \in A} Q(s,a)$
    $\Delta \leftarrow \max\{\Delta, |V(s) - Q(s,\pi(s))|\}$
    $V(s) \leftarrow Q(s,\pi(s))$
  end
until $\Delta < \epsilon$
return π

算法 2.2: 无限规划时限 MDP 值迭代算法

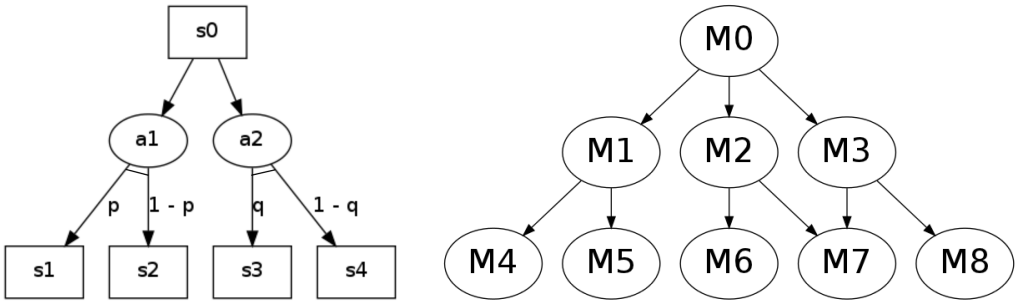
法，这就是值迭代算法的主要思想。值迭代算法在每个迭代周期内，对所有状态按照以下公式进行*备份操作*（Backup Operation）：

$$V_{t+1}(s) = \max_{a \in A} \left\{ R(s,a) + \lambda \sum_{s' \in S} T(s' \mid s,a) V_t(s') \right\}, \tag{2.20}$$

其中 V_t 是第 t 次迭代后的值函数。注意到，公式 2.20其实就是把贝尔曼最优性等式作为更新规则来迭代计算值函数。值迭代算法的一个简单的改进是使用*异步动态规划*（Asynchronous Dynamic Programming），即在原地进行备份操作，而不区分 V_t 和 V_{t+1}。异步迭代允许更灵活的按照不同的顺序进行备份操作，比如可以优先更新尚未收敛的状态空间区域。算法 2.2给出了完整的异步值迭代算法。跟算法 2.1类似，算法 2.2接受一个完整的 MDP 模型和一个足够小正数作为输入，输出一个近似最优策略。可以证明算法 2.2返回的是一个 $2\epsilon\frac{\gamma}{1-\gamma}$-最优策略[24]。值迭代算法高效地在一次状态空间*扫描*（Sweep）过程里面同时集成了策略评估和策略改进，因此常常比简单应用策略迭代算法有更快的收敛速度。

2.2.2　在线求解算法

基于动态规划的离线规划算法对应大规模问题往往并不实用，因为离线规划算法要求遍历状态整个状态空间计算出完整的策略，而状态空间大小跟状态空间维度呈指数增加——所谓“维度诅咒”问题。另一方面，很多实际问题的环境模型（主要指状态转移函数和回报函数）会随时间而变化，使得离线计算得到的策略往往在在线执行时已经过时。在线规划算法尝试只在当前状态下计算出一个最优行动（或局部策略）来克服以上难点。在线规划算法的主要理论依据是，智能体在环境中交互时只会遇到很有限的状态，从而无需为整个状



(a) 与或树　　(b) MAXQ 任务图

图 2.2　与或树和 MAXQ 任务图举例

态空间计算出完整策略。本节主要介绍*与或树搜索*（AND/OR Tree Search）[54]、*实时动态规划*（Real-Time Dynamic Programming, RTDP）[55] 和*蒙特卡洛树搜索*（Monte-Carlo Tree Search）[56]。

2.2.2.1　与或树搜索算法

智能体为了在当前状态下做出最优决策，必须从当前状态出发，递归地评估所有可以执行的动作和可能遇到的状态，这一过程构建了一个以当前状态为根节点的搜索树，又称*与或树*（AND/OR Tree）[54]。树上的与节点是动作节点，或节点是状态节点。图 2.2(a)给出了一棵简单与或树的例子。图中，s_0 是当前状态，s_0 下有两个可执行的动作 a_1 和 a_2。状态转移概率是：$\Pr(s_1 \mid s_0, a_1) = p$，$\Pr(s_2 \mid s_0, a_1) = 1 - p$，$\Pr(s_3 \mid s_0, a_2) = q$，以及 $\Pr(s_4 \mid s_0, a_2) = 1 - q$。

AO* 是一个求解与或树搜索的*最优优先搜索*（Best-First Search）算法。算法首先初始化一个只包含根节点的部分图 G，然后迭代地进行以下操作：1. 构建 G 上的最优部分策略并扩展图 G 的一个非终端叶子节点；2. 重新根据贝尔曼最优性等式计算扩展后的图 G 上的最优值函数。当图 G 的所有叶子节点都是原问题的终端节点时，算法终止，图 G 上的最优策略就是原问题的最优策略。跟动态规划算法不同，AO* 不需要遍历所有状态空间，而仅在当前状态的可达状态内进行搜索，算法终止时可以找到当前状态下的最优局部策略。另外，使用紧致的启发函数来扩展叶子节点又可以进一步加快搜索。算法 2.3给出了完整的 AO* 算法[57]。

2.2.2.2　实时动态规划算法

实时动态规划（RTDP）算法迭代地进行*试验搜索*（Trial-based Search）来为当前状态找到“最优”动作。每次试错搜索都从当前状态出发，根据一定的动作选择策略选择一个动作去执行，根据底层 MDP 模型采样得到立即回报和下一个状态，根据贝尔曼最优性等式更新上一个状态的值函数，重复这个过程直

Input: An MDP $\langle S, A, T, R\rangle$, graph G initially empty, heuristic function h, current state s_0, and planing horizon H
Output: An action a

Let $G \leftarrow G \cup (s_0, H)$
Let $V(s_0, H) \leftarrow h(s_0, H)$
Initialize best partial graph to G

while *True* **do**
  Let $(s, d) \leftarrow$ non-terminal tip node in best partial graph
  if (s, d) *is null* **then**
    break
  end
  foreach $a \in A$ **do**
    Add node (a, s, d) as child of s, d
    foreach $s' \in S$ **do**
      if $T(s' \mid s, a) > 0$ **then**
        Add node $s', d-1$ as child of (a, s, d)
        if $d - 1 = 0$ **then**
          $V(s', d-1) \leftarrow 0$
        end
        else
          $V(s', d-1) \leftarrow h(s', d-1)$
        end
      end
    end
  end
  foreach $s \in G$ *in a bottom-up way* **do**
    $Q(s, a, d) \leftarrow R(s, a) + \gamma \sum_{s' \in S} T(s \mid s, a) V(s', d-1)$
    $V(s, d) \leftarrow \max_{a \in A} Q(s, a, d)$
    if $V(s, d) = Q(s, a, d)$ **then**
      Mark state s and action a
    end
  end
  Recompute best partial graph to G by following marked actions
end

return marked action for state s_0 in best partial graph to G

算法 2.3: 有限规划时限 MDP 问题的 AO* 算法

到遇到终止状态，或搜索深度达到事先设计的规划时限。重复若干次试错搜索，最后根据树上的值函数返回当前状态应该执行的“最好”动作。RTDP 试错搜索的动作选择策略一般采用贪心策略。跟 AO* 类似，RTDP 也可以使用启发函数来初始化新的节点。RTDP 算法只更新搜索过程中通过采样遇到的状态，无需遍历整个状态空间。另外，可以证明，计算资源允许的情况下，RTDP 可以渐近找到最优解。算法 2.4展示了有限规划时限 MDP 问题完整的 RTDP 算法[58, 59]。

Input: An MDP $\langle S, A, T, R \rangle$, heuristic function h, current state s_0, and planing horizon H
Output: An action a

foreach $s \in S$ **do**
    Initialize $V(s) \leftarrow h(s)$
end

repeat
    Let $s \leftarrow s_0$
    Let $d \leftarrow 0$
    while *True* **do**
        $d \leftarrow d + 1$
        foreach $a \in A$ **do**
            $Q(s, a) \leftarrow R(s, a) + \gamma \sum_{s \in S'} T(s' \mid s, a) V(s')$
        end
        Let $a^* \leftarrow \text{argmax}_{a \in A}\, Q(s, a)$
        Update $V(s) \leftarrow Q(s, a^*)$
        Sample $s' \sim T(s' \mid s, a^*)$
        if s' *is goal state* **or** $d > H$ **then**
            break
        end
        Let $s \leftarrow s'$
    end
until *resource budgets reached*

return $a^* \leftarrow \text{argmax}_{a \in A}\, Q(s_0, a)$

算法 2.4: 有限规划时限 MDP 问题的 RTDP 算法

2.2.2.3　蒙特卡洛树搜索算法

蒙特卡洛树搜索（MCTS）基于蒙特卡洛仿真通过构建一棵非对称的最优优先搜索树来找到当前状态下的“最好”动作[60]。MCTS 算法的优势是不需要事先知道完整的 MDP 模型，而仅需要一个底层 MDP 的*生成模型*（Generative Model）即可——也就是*仿真器/模拟器*（Simulator）。MCTS 的主要思想是通过从一个状态节点出发获得的仿真数据来评估这个状态。具体来说，对每一个状态节点，MCTS 根据一个动作选择策略来选择一个动作；通过蒙特卡洛仿真执行这个动作；观察新的状态，如果这个状态已经在树上就递归评估这个状态，否则就把这个状态节点插到树上，并从这个状态出发仿真运行默认的 Rollout 策略，直到遇到终止条件；最后，通过反向回溯来更新路径上每个状态的统计量。图 2.3概括了 MCTS 算法的几个主要步骤[61]。迭代式地重复这个过程，MCTS 最终构建了一棵非对称的搜索树，使得 MCTS 更倾向于搜索状态空间中更有潜力的局部区域。图 2.4展示了一棵非对称搜索树的例子[62]。MCTS 是 Anytime 算法，任意时刻被打断，MCTS 根据当前树上的值函数报告一个最优动作供智能体执行。值得一提的是，MCTS 是可以高度并行化的。以上特性，决定了 MCTS 可以适用

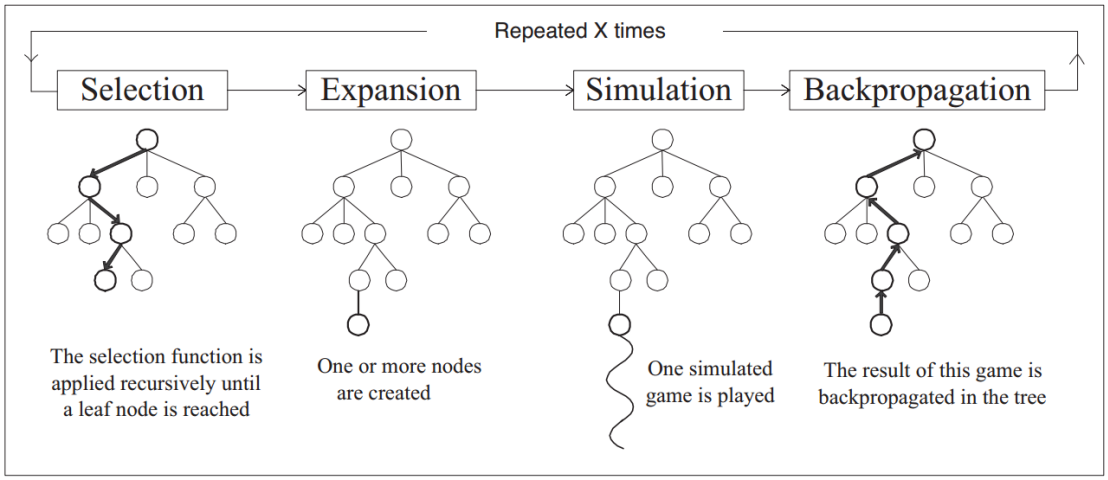


图 2.3　蒙特卡洛树搜索的主要步骤举例

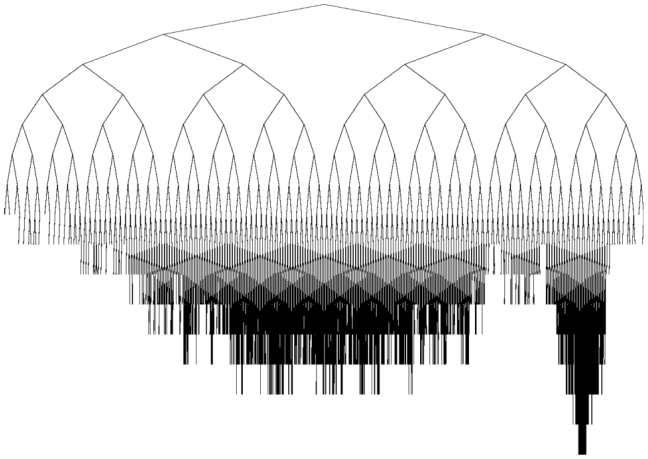


图 2.4　蒙特卡洛树搜索获得的非对称性搜索树举例

于规模特别巨大的实际问题的潜力。

UCT 可以说是目前最流行的 MCTS 实现之一[56]。UCT 把每个决策节点看成是一个 MAB 问题，根据 UCB[63] 启发函数来选择动作，具体地：

$$\mathrm{UCB}(s,a) = \bar{Q}(s,a) + c\sqrt{\frac{\log N(s)}{N(s,a)}}, \tag{2.21}$$

其中 $\bar{Q}(s,a)$ 是所有仿真中动作 a 作用于状态 s 的平均行动值，$N(s,a)$ 是所有仿真中动作 a 作用于状态 s 的次数，$N(s) = \sum_{a\in A} N(s,a)$ 是访问状态 s 的次数，c 是一个利用—探索平衡因子。已经被证明，c 取值合适的情况下，如果计算资源允许，UCT 算法将以概率 1 找到最优策略。算法 2.5给出有限规划时限基于成本模型的 MDP 问题的 UCT 算法示例[57]。

Input: An MDP simulator sim, current state s_0, search graph G initially empty, rollout policy π, planning horizon H, exploration constant C
Output: An action a

```
UCT(s : state, d : depth, sim : simulator, G : graph, π : policy,
H : horizon, C : constant)
if d ⩾ H or s is terminal then
    return 0
end
if node (s, d) ∉ G then
    Add node (s, d) to graph G
    Initialize N(s, d) ← 0 and N(s, a, d) ← 0 for all a ∈ A
    Initialize Q(s, a, d) ← 0 for all a ∈ A
    Play rollout policy π from s for H − d steps according to simulator sim
    Let r ← the sampled cumulative discounted reward
    return r
end
else
    foreach a ∈ A do
        if N(s, a, d) > 0 then
            Let Bonus(a) ← C√(log N(s, d)/N(s, a, d))
        end
        else
            Let Bonus(a) ← ∞
        end
    end
    Select a* ← argmax_{a∈A} {Q(s, a, d) + Bonus(a)}
    Sample s′ ~ T(s′ | s, a) according to simulator sim
    Let nv ← R(s, a) + γUCT(s′, d + 1, sim, G, π, H, C)
    Increment N(s, d) and N(s, a, d)
    Update Q(s, a, d) ← Q(s, a, d) + (nv − Q(s, a, d))/N(s, a, d)
    return nv
end

repeat
    UCT(s_0, 0, sim, G, π, H, C)
until resource budgets reached

return a* ← argmax_{a∈A} Q(s_0, a, 0)
```

算法 2.5: 有限规划时限 MDP 问题的 UCT 算法

2.3　马尔科夫决策过程分层分解

分层分解技术把原 MDP 问题分解成一系列可以被更容易解决的子问题[64]。常见的分层分解技术包括 *Option 理论*[65]，*层次抽象机*（Hierarchies of Abstract Machines）[66] 和 *MAXQ 分层分解*（MAXQ Hierarchical Decomposition）[37]。Option 理论在马尔科夫模型里面引入*宏动作*（Macro Action）——即 Option。

每个 Option o 包括一个初始状态集合 $I_o \subseteq S$，一个子策略策略 π_o 和一个终止条件 $\beta_o : S \to [0,1]$，其中 $\beta_o(s)$ 给出状态 s 下 Option o 终止的概率。一般每个 Option 都被设计成完成一个原问题的子任务。层次抽象机把原 MDP 的策略分解成若干子“程序”，每个子程序在各自的状态空间内运行——构成一个有限状态机。当前子程序运行结束时，高层策略会面临一次子程序选择，即选择下一个将要执行的子程序。MAXQ 分层分解把原 MDP 问题分解成一棵任务树上的若干子任务，每个子任务都是一个以其树上子节点对应的其他子任务为宏动作的 MDP。求解根节点对应的 MDP 就等价于求解了原 MDP 问题。这三类主要方法都依赖于*半马尔科夫决策过程*（Semi-Markov Decision Process, SMDP）基本理论。本节首先介绍 SMDP 基本理论，然后着重介绍 MAXQ 分层分解技术。

2.3.1　半马尔科夫决策过程

半马尔科夫决策过程（SMDP）为 MDP 分层分解提供了基本的理论支持。MDP 模型只关注动作执行的顺序，而不关心动作执行需要的时间。MDP 假设每个动作都可以在一个决策周期内执行完毕。SMDP 模型允许执行每个动作执行需要的周期数可以不同，甚至可以服从某一概率分布。令随机变量 $\tau \in \mathbb{N}^+$ 表示状态 s 下执行动作 a 需要等候的时间，SMDP 模型下的状态转移函数 $T(s', \tau \mid s, a)$ 给出状态 s 下执行动作 a 经过时间 τ 后转移到状态 s' 的联合概率；立即回报函数 $R(s, a)$ 给出状态 s 下动作 a 执行期间累积获得的期望回报。SMDP 模型关于值函数的贝尔曼最优性等式为：

$$V^*(s) = \max_{a \in A} \left\{ R(s,a) + \sum_{s' \in S, \tau \in \mathbb{N}^+} \gamma^\tau T(s', \tau \mid s, a) V^*(s') \right\}. \tag{2.22}$$

动态规划算法可以直接扩展到 SMDP 模型。SMDP 模型下的分层规划为处理大时间尺度下的规划提供了更自然的方式。比如人类做长远规划时，不会也没有必要考虑好每一秒钟应该执行的行动，而是只会规划好几个阶段——也就是分层规划中的宏动作，每个阶段可以需要不同的时间，并且每个阶段内的策略也可能相对独立。

2.3.2　MAXQ 分层分解

MAXQ 分层分解把原 MDP 问题 M 分解成一棵任务树上的若干子 MDP 问题，表示为 $\{M_0, M_1, \cdots, M_n\}$。每一个子 MDP 被认为是一个子任务。特别地，$M_0$ 是根任务，求解 M_0 即求解了原始 MDP 问题 M。一个不带参数的子任务 M_i 可以被定义成一个三元组 $\langle T_i, A_i, \tilde{R}_i \rangle$，其中：

- T_i 是子任务 M_i 的终止条件，意味着该任务有组*活动状态*（Active States）S_i，和一组*终止状态*（Terminal States）G_i——即子目标；

- A_i 是子任务 M_i 可以执行的行动集合，可以是原子动作，也可以是分层结构树上的下层子任务；
- $\tilde{R}_i$ 是子任务 M_i 内部的（可选）伪回报函数，给出从任意活动状态 $s \in S_i$ 转移到终止状态 $g \in G_i$ 的回报。

就像程序语言里面的函数一样，子任务也可以带参数。绑定不同参数的子任务实际上就给出了同一子任务的不同实例。原子行动被看成原子任务。原子任务总是可以执行的，并且他们在执行后立即返回。MAXQ 分层结构可以用*任务图*（Task Graph）来表示。图 2.2(b)给出了一个 MAXQ 任务图的例子。图中，*根任务*（Root Task）M_0 有三个宏动作：M_1、M_2 和 M_3。子任务 M_1、M_2 和 M_3 共享原子任务 M_i （$4 \leqslant i \leqslant 8$）作为他们的宏动作。

给定原问题的分层结构，MAXQ *分层策略*（Hierarchical Policy）π 定义成对每个子任务而言的一组策略 $\pi = \{\pi_0, \pi_1, \cdots, \pi_n\}$，其中 $\pi_i : S_i \to A_i$ 是子任务 M_i 的子策略。*投影值函数*（Projected Value Function）$V^\pi(i,s)$ 定义成在状态 s 下执行策略 $\pi = \{\pi_0, \pi_1, \cdots, \pi_n\}$ 直到子任务 M_i 终止于某个终止状态 $g \in G_i$ 的过程中智能体能获得的期望累积回报。类似地，定义行动值函数 $Q^\pi(i,s,a)$ 为首先执行宏动作 M_a，然后服从策略 $\pi = \{\pi_0, \pi_1, \cdots, \pi_n\}$ 直到子任务 M_i 终止的过程中智能体能获得的期望累积回报。容易看出，对于原子任务 M_a 而言，$V^\pi(a,s) = R(s,a)$。

分层策略 π 的值函数可以被递归表示为：

$$Q^\pi(i,s,a) = V^\pi(a,s) + C^\pi(i,s,a), \tag{2.23}$$

其中

$$V^\pi(i,s) = \begin{cases} R(s,i), & \text{如果 } M_i \text{ 是原子任务} \\ Q^\pi(i,s,\pi(s)), & \text{其他情况} \end{cases} \tag{2.24}$$

这里，$C^\pi(i,s,a)$ 是*完成函数*（Completion Function），给出子任务 M_i 执行宏动作 M_a，在 M_a 终止后，M_i 终止前智能体服从策略 π 能获得的期望累积回报。完成函数定义如下：

$$C^\pi(i,s,a) = \sum_{s',N} \gamma^N \Pr(s', N \mid s, a) V^\pi(i, s'), \tag{2.25}$$

其中，$\Pr(s', N \mid s, a)$ 给出状态 s 下执行宏动作 M_a，在 N 步以后于状态 s' 终止的概率。*递归最优策略*（Recursively Optimal Policy）π^* 对应的值函数称为递归最优值函数，满足：

$$Q^*(i,s,a) = V^*(a,s) + C^*(i,s,a), \tag{2.26}$$

其中

$$V^*(i,s) = \begin{cases} R(s,i), & \text{如果 } M_i \text{ 是原子任务} \\ \max_{a \in A_i} Q^*(i,s,a), & \text{其他情况} \end{cases} \tag{2.27}$$

并且 $C^*(i,s,a) = C^{\pi^*}(i,s,a)$ 为递归最优策略 π^* 的完成函数。子任务 M_i 的递归最优策略进一步可以由贪心法给出：

$$\pi_i^*(s) = \underset{a \in A_i}{\operatorname{argmax}}\, Q^*(i,s,a). \tag{2.28}$$

2.4　部分可观察马尔科夫决策过程基本模型

MDP 模型假设环境状态完全可观察，但很多实际问题并非如此。比如机器人不可能准确观察环境状态的每个部分，环境中多多少少会存在隐藏信息，比如被障碍物遮挡住的部分、传感器感知范围以外的部分等。部分可观察马尔科夫决策过程（POMDP）引入观察和观察模型把 MDP 扩展到部分可观察环境[67]。观察是跟环境状态相关的随机变量，表示智能体对环境状态的测量值，观察的具体取值由环境的观察模型决定。

定义 2.4.1 (部分可观察马尔科夫决策过程)**.** 部分可观察马尔科夫决策过程形式上可以被定义成一个六元组 $\langle S, A, O, T, \Omega, R\rangle$，其中：

- S, A, T, R 的定义跟马尔科夫决策过程一致（见定义*2.1.1*）；
- O 是观察空间（*Observation Space*），即智能体所有可能获得的观察集合；
- $\Omega : S \times A \times O \to [0,1]$ 是观察函数（*Observation Function*），$\Omega(o \mid s, a)$ 给出智能体执行完行动 a，环境转移到状态 s 后，观测到观察 o 的概率。

POMDP 可以被转化称定义在信念状态空间上的 MDP。信念状态指智能体对目前环境所处状态的概率估计，是所有过去观察、行动历史的统计充分量。记信念状态为 b，则 $b(s)$ 给出当前环境处于状态 s 的概率，即 $b(s) = \Pr(s \mid b)$。给定初始信念状态 b_0 和观察—行动历史 $h = (a_0, o_1, a_1, o_2, \ldots a_{t-1}, o_t)$，则当前的信念状态可以递归地根据贝叶斯推理方法唯一确定，即：

$$b'(s') = \eta \Omega(o \mid s', a) \sum_{s \in S} T(s' \mid s, a) b(s), \tag{2.29}$$

其中 $\eta = 1/P(o \mid b, a)$ 是一个正则化因子，展开为：

$$\eta = \frac{1}{\sum_{s \in S'} \Omega(o \mid s', a) \sum_{s \in S} T(s' \mid s, a) b(s)}. \tag{2.30}$$

公式 2.29即完成了信念状态的更新，以可以写成关于 b，a 和 o 的函数形式：$b' = \zeta(b, a, o)$。ζ 函数一般被称为贝叶斯滤波器（Bayesian Filter）。

令 $\mathcal{B}$ 为所有信念状态的集合——即信念空间，则 POMDP 的策略 π 定义成信念空间到动作空间的映射，即 $\pi : \mathcal{B} \to A$。跟 MDP 类似，求解 POMDP 的目标就是找到最优策略 π^* 使得期望累积回报最大。引入信念状态以后，POMDP 被转化称定义在信念状态上的 MDP，又称贝叶斯自适应 *MDP*（Bayesian-Adaptive

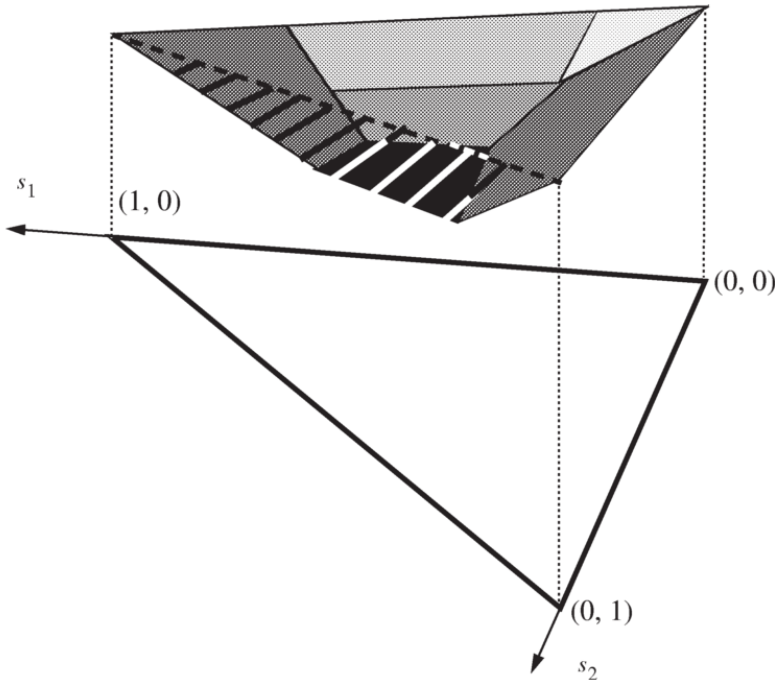


图 2.5　POMDP 值函数的分段线性凸性质

MDP, BAMDP)：$\langle \mathcal{B}, A, T^+, r \rangle$，其中 $\mathcal{B}$ 是 BAMDP 的状态空间，A 是行动空间，$r(b,a) = \sum_{s \in S} b(s) R(s,a)$ 是立即立即回报函数，T^+ 是状态转移函数，定义为：

$$T^+(b' \mid b, a) = \sum_{o \in O} \mathbf{1}[b' = \zeta(b,a,o)] \Omega(o \mid b, a), \tag{2.31}$$

其中 **1** 指示函数（Indicator Function）。BAMDP 的贝尔曼最优性等式是：

$$V^*(b) = \max_{a \in A} \left\{ r(b,a) + \gamma \sum_{o \in O} \Omega(o \mid b, a) V^*\left(\zeta(b,a,o)\right) \right\}. \tag{2.32}$$

给定最优值函数 V^*，最优策略 π^* 可以由下式得到：

$$\pi^*(b) = \operatorname*{argmax}_{a \in A} \left\{ r(b,a) + \gamma \sum_{o \in O} \Omega(o \mid b, a) V^*\left(\zeta(b,a,o)\right) \right\}. \tag{2.33}$$

BAMDP 的最优策略，加以正确的信念状态更新就是原 POMDP 问题的最优策略。剩下的问题就是如何求解 BAMDP，但这并不容易，因为 BAMDP 是定义在信念空间上的连续 MDP，下一节将利用 BAMDP 值函数所具有的特殊性质来求解这一问题。给定一个初始信念状态，信念状态和历史是可以互换使用。本文描述算法时为了方便主要使用信念状态，但在实现算法时主要使用历史。

2.5　部分可观察马尔科夫决策过程求解算法

跟 MDP 问题类似，如果可以求解出 POMDP 问题的最优值函数，也就得到了其最优策略。本节主要从离线规划和在线规划两个方面介绍 POMDP 的精确

和近似求解技术。连续信念空间上 POMDP 的精确求解主要依赖于其值函数具有的分段线性凸（Piecewise-Linear and Convex）性质[68]，剩余规划时限为 t 的值函数 V_t 可以表示成一个 $|S|$ 维超平面的集合：$\Gamma_t = \{\alpha_0, \alpha_1, \ldots, \alpha_m\}$。每个超平面又被为一个 α-向量，实际是信念空间上某一行动 $a \in A$ 的行动值函数。信念状态 b 上的值函数，就是该信念状态关于所有 α-向量行动值函数的最大值。最优动作也就是最大行动函数所对应的动作：

$$V_t(b) = \max_{\alpha \in \Gamma_t} \sum_{s \in S} \alpha(s)b(s). \tag{2.34}$$

图 2.5展示了 3 个状态的 POMDP 问题分段线性凸值函数的一个例子[24]。注意到 3 状态 POMDP 的信念空间其实是一个 2-单纯形。

2.5.1　离线求解算法

离线求解算法试图建立整个信念空间上的完整策略。本节主要介绍精确求解的值迭代算法和近似求解的基于信念点的算法。

2.5.1.1　值迭代

MDP 的值迭代算法可以直接应用于 POMDP 问题，需要注意的如何快速从 V_{t-1} 获得 V_t 的 α-向量表达。主要挑战是精确表达信念空间所需要的 α-向量的个数关于观察数目呈指数增长，也就是 $|\Gamma_t| = \mathcal{O}(|A||\Gamma_{t-1}|^{|O|})$。规划时限为 t 时，POMDP 问题的实际计算复杂度是 $\mathcal{O}(|A||Z||S|^2|\Gamma_{t-1}|^{|Z|})$。为此，精确求解的值迭代算法主要关注于如何对 Γ_t 集合进行有效的剪枝以减少对总体计算资源的需要。常见的剪枝技术包括 One-Pass 算法[69]、Witness 算法[70]、增量裁剪算法等[71]。完整值迭代算法的计算复杂度非常高，更为实用的是近似算法，比如 QMDP 算法、基于信念点的算法等。

2.5.1.2　QMDP

完整的 POMDP 模型是包含了底层 MDP 模型的，QMDP 假设行动一步以后环境的观察不确定性就不复存在，QMDP 算法使用底层 MDP 模型的最优行动值函数近似计算原 POMDP 问题的行动值函数，具体地：

$$\hat{Q}(b, a) = \sum_{s \in S} Q_{MDP}(s, a)b(s), \tag{2.35}$$

其中 $Q_{MDP}(s, a)$ 是底层 MDP 问题的最优行动值函数。进一步，值函数由下式给出：

$$\hat{V}(b) = \max_{a \in A} \hat{Q}(b, a). \tag{2.36}$$

底层 MDP 问题的行动值函数是 POMDP 问题的上界，QMDP 算法忽略了观察不确定性，不能很好区分观察动作跟普通动作的区别，所以算法的实际效果可能非常差，但仍然是比较好的基准算法。

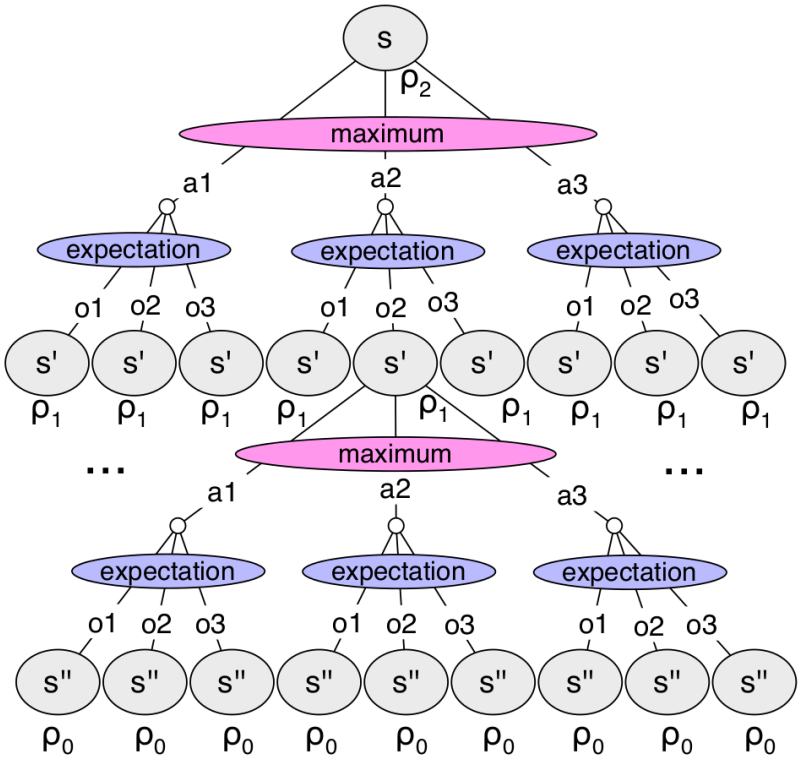


图 2.6　部分可观察 MDP 搜索树

2.5.1.3　基于信念点的算法

基于信念点的算法使用若干采样出来的下一步信念点来更新当前信念状态的值函数，并维护更新后值函数在信念点样本空间上的梯度信息[72]。基于信念点的算法通过采样出来的最多只包含一个 α-向量的若干信念点来更新值函数，克服了精确算法需要考虑所有信念点的计算复杂度问题。令 B 表示采样出来的信念点集合，规划时限为 t 时的 α-向量集合 Γ_t 可以按照下面公式获得：

$$
\begin{aligned}
&\alpha^a(s) = R(s,a),\\
&\Gamma_t^{a,o} = \left\{ \alpha_i^{a,o} \middle| \alpha_i^{a,o}(s) = \gamma \sum_{s' \in S} T(s' \mid s,a)\Omega(o \mid s',a)\alpha_i'(s'), \alpha_i' \in \Gamma_{t-1} \right\},\\
&\Gamma_t^b = \left\{ \alpha_b^a \middle| \alpha_b^a = \alpha_b^a + \sum_{o \in O} \operatorname*{argmax}_{\alpha \in \Gamma_t^{a,o}} \sum_{s \in S} \alpha(s)b(s), a \in A \right\},\\
&\Gamma_t = \left\{ \alpha_b \middle| \alpha_b = \operatorname*{argmax}_{\alpha \in \Gamma_t^b} \sum_{s \in S} b(s)\alpha(s), b \in B \right\}.
\end{aligned}
\tag{2.37}
$$

为保证能正确获得最优值函数的下界，Γ_0 初始化成只有一个 α-向量：$\alpha_0(s) = \frac{1}{1-\gamma} \min_{s' \in S, a \in A} R(s',a)$。因为 $\Gamma_{t-1} \leqslant |B|$，每次迭代的计算复杂度是 $\mathcal{O}(|A||O||S||B|(|S|+|B|))$，为多项式时间。注意到，精确求解算法的每步迭代的计算复杂度是指数时间的。常见的基于信念点的算法比如 PBVI[73]、Perseus[74]、

```
Input: Current belief b_0, AND-OR search tree T, planning horizon H, lower
       bound on L, upper bound U
Let b ← b_0
Initialize T to contain only b at the root
while not ExectionTerminated() do
    while not PlanningTermiated() do
        Let b* ← ChooseNextNodeToExpand()
        Expand(b*, H)
        UpdateAncestors(b*)
    end
    Execute best action α* for b
    Perceive a new observation o
    Update b ← ζ(b, α*, o)
    Update tree T so that b is the new root
end
```

算法 2.6: POMDP 在线规划算法基本框架

HSVI[75]、SARSOP[76] 等。这些算法的主要区别在于如何采样信念点和如何更新这些采样出的信念的的值函数。基于信念点的算法的一个优势是可以通过控制采样信念点的数目来权衡计算复杂性和值函数的精度。

2.5.2　在线求解算法

离线规划算法为所有可能的信念状态计算出最优行动，只能适用于中等规模的问题，因为完整策略的计算复杂度非常大。另外，基于信念点的算法求解大规模 POMDP 问题得到的解的质量也非常有限。跟 MDP 问题类似，在线规划算法更适用于大规模 POMDP 问题。在线规划算法的主要优势是只需要考虑当前信念状态出发的可达信念空间，并且由于只需要计算出当前信念状态下的最优动作，所以实际并不需要计算出完整的 α-向量。本节主要介绍启发式搜索、信念空间实时动态规划和蒙特卡洛树搜索等几种常见在线规划算法。

2.5.2.1　启发式搜索

跟 MDP 问题类似，POMDP 问题也可以转化成与或树上的搜索问题。作为例子，图 2.6展示了典型的 POMDP 问题搜索树[2]。图中，根节点 s 表示智能体当前的信念状态，执行动作 a 并获得观察 o 后，智能体根据信念更新公式公式 2.29更新到新的信念状态 s'，同理，继续更新得到下一个信念状态 s''，搜索过程一直持续到终端节点，或搜索深度达到事先设计的规划时限。搜索结果向根节点反馈时，观察节点需要综合考虑所有可能的情况（Expectation），动作节点则需要返回可选情况中的最好情况（Maximization）。算法 2.6给出了 POMDP 在线规划算法的基本框架[77]。算法首先使用初始信念状态初始化搜索树，然后迭代地扩展搜索树并更新树上节点的值函数。规划阶段结束时，根据树上值函

Input: Current belief b_0, initial approximate value function V_0, hashtable of beliefs and approximate values V, discretization resolution k

Initialize b to the b_0 and V to an empty hashtable
while not ExectionTerminated() **do**
  foreach $a \in A$ **do**
    Evaluate
    $Q(b,a) \leftarrow r(b,a) + \gamma \sum_{o \in O} \Pr(o \mid b,a) V(\text{Discretize}(\zeta(b,a,o),k))$
  end
  Select $a^* \leftarrow \text{argmax}_{a \in A} Q(b,a)$
  Execute best action a^* for b
  $V(\text{Discretize}(b,k)) \leftarrow Q(b,a^*)$
  Perceive a new observation o
  Let $b \leftarrow \zeta(b,a^*,o)$
end

算法 2.7: POMDP 问题实时动态规划 RTDP-Bel 算法

数选择执行一个最优动作，获得新观察，更新得到新的信念状态，并进入新的决策周期。

启发式搜索算法通过使用启发式信息关注于最相关的可达信念空间内的搜索。最相关的信念状态是指，可以让智能体尽快作出足够好的决策的信念空间比如展开的新节点数目要尽量少。常见的启发式技术包括 Satia-Lave[78]、BI-POMDP[79]、AEMS[80] 等。这些算法的主要区别在于扩展搜索树时选用的启发函数。

2.5.2.2　信念空间实时动态规划

信念空间实时动态规划算法（RTDP-Bel）把 MDP 问题的动态规划算法扩展到 POMDP 问题[58]。RTDP-Bel 从当前信念状态出发进行一系列的试验搜索，并更新遇到的信念状态的值函数。在遇到的每一个信念状态，RTDP-Bel 根据下式所示的近似行动值函数选择动作以展开该节点：

$$\hat{Q}(b,a) = r(b,a) + \gamma \sum_{o \in O} \Omega(o \mid b,a) V(\zeta(b,a,o)), \tag{2.38}$$

其中 $V(b)$ 是当前树上维护的信念状态 b 的值函数，对于新遇到的信念状态也可以是启发函数。为了保存所有遇到的信念状态，也为了泛化已知信念状态到未知信念状态，RTDP-Bel 使用近似函数对信念状态进行离散化处理。算法 2.7展示了 RTDP-Bel 的一个试验搜索[77]，其中 $\text{Discretize}(b,k)$ 函数返回信念状态 b 离散后的信念状态 b'，使得 $b'(s) = \text{round}(kb(s))/k$。值得注意的是，离散后的信念空间大小为 $\mathcal{O}((k+1)^{|S|})$。

Input: An MDP simulator sim, current history h_0, search tree T initially empty, rollout policy π, termination condition ϵ, exploration constant C
Output: An action a

Rollout($s : state,\ h : history,\ d : depth$)
if $\gamma^d < \epsilon$ **then**
  return 0
end
Select $a \leftarrow \pi(h, \cdot)$
Sample $(s', o, r) \sim sim(s, a)$
return $r + \gamma$Rollout$(s', hao, d + 1)$

Simulate($s : state,\ h : history,\ d : depth$)
if $\gamma^d < \epsilon$ **then**
  return 0
end
if $h \notin T$ **then**
  foreach $a \in A$ **do**
    $T(ha) \leftarrow (N_{init}(ha), V_{init}(hs), \varnothing)$
  end
  return Rollout(s, h, d)
end
Select $a^* \leftarrow \text{argmax}_{a \in A} \left\{ V(ha) + C\sqrt{\log N(h)/N(ha)} \right\}$
Sample $(s', o, r) \sim sim(s, a^*)$
$R \leftarrow r + \gamma$Simulate$(s', ha^*o, d + 1)$
$B(h) \leftarrow B(h) \cup \{s\}$
Increment $N(h)$ and $N(ha^*)$
Update $V(ha^*) \leftarrow V(ha^*) + \frac{R - V(ha^*)}{N(ha^*)}$
return R

repeat
  Sample $s \sim B(h)$
  Simulate$(s, h, 0)$
until *resource budgets reached*

return $a^* \leftarrow \text{argmax}_{a \in A} V(ha)$

算法 2.8: 有限规划时限 POMDP 问题的 POMCP 算法

2.5.2.3　蒙特卡洛树搜索

跟 MDP 问题类似，POMDP 蒙特卡洛树搜索算法使用蒙特卡洛方仿真来评估搜索树上的节点。POMCP 算法把 UCT 算法扩展到 POMDP 问题[81]。POMCP 在每个节点保存该节点对应的历史序列信息 h，而不是显示保存信念状态。每次迭代过程中，POMCP 根据根结点的信念状态 $b(h)$ 采样出一个状态，并从该状态开始组织搜索——即所谓的*根采样*（Root Sampling）技术。传统的 MCTS 是需要一个基于信息状态的模拟器的，在 MDP 问题里面就是需要一个基于状态的

模拟器，在 POMDP 问题里面就是需要一个基于信念状态的模拟器，POMCP 的根采样技术使得仅依赖于一个基于状态的模拟器成为可能。在每一个决策节点，POMCP 根据 UCB 启发函数来选择动作：

$$\mathrm{UCB}(h,a)=\bar{Q}(h,a)+c\sqrt{\frac{\log N(h)}{N(h,a)}}, \tag{2.39}$$

其中 $\bar{Q}(h,a)$ 是过去所有仿真中在历史节点 h 执行 a 的平均回报，$N(h,a)$ 是在节点 h 执行动作 a 的次数，$N(h)=\sum_{a\in\mathcal{A}}N(h,a)$ 是节点 h 访问过的总次数，c 是利用—探索平衡因子。POMCP 使用粒子滤波[82, 83] 根据采样得到的观察、回报和状态转移来为某个历史节点维护了一组可能的真实状态引来表示其信念状态。可以证明，对合适的 c 取值，POMCP 构造的值函数可以以概率 1 收敛与最优值函数。POMCP 在很多问题里面都取得了很成功的应用[84–86]。算法 2.8给出了 POMCP 算法的主要流程[81]。

2.6　本章小结

本章详细介绍了马尔科夫决策相关基础理论和常用求解算法。

1. MDP 为行动不确定性环境下的规划问题提供了基本理论模型。精确求解 MDP 问题的计算复杂度关于状态空间维度是呈指数增加的，无法直接应用于大规模实际问题。在线规划方法关注于只为当前状态计算出最优行动，避免了计算整个状态空间的完整策略，使 MDP 应用到大规模问题成为可能。常见的在线规划技术包括与或树搜索、实时动态规划和蒙特卡洛仿真等。

2. 分层分解是另一项广泛使用的用于扩展 MDP 算法到大规模问题的技术。分层分解利用问题存在的层次结构把原问题分解成一系列可以被更容易解决的子问题，解决这些子问题也就求解了原问题。分层分解主要得益于时序扩展动作、状态抽象和子任务共享，理论基础是半马尔科夫决策过程，常见的基本技术包括 Option 理论、层次抽象机和 MAXQ 分层分解。

3. POMDP 模型通过引入观察和观察函数，把 MDP 扩展到了部分可观察环境。POMDP 可以看成是定义在信念状态空间上的连续 MDP。精确求解 POMDP 每一步迭代的计算复杂度关于状态空间维度是都双指数的，无法直接应用于一般中等规模的问题。基于信念点的算法试图通过仅采样若干信念点来更新当前信念状态的值函数，一定程度上可以克服精确算法计算复杂度的问题。在线规划算法通过只计算可达信念空间内的值函数来避免计算整个信念空间上的完整策略，进一步使得 POMDP 应用于大规模问题成为可能。常见的 POMDP 在线规划技术包括启发式搜索、信念空间实时动态规划和蒙特卡洛仿真等。

4. 蒙特卡洛树搜索方法面临的一个基本问题是如何进行利用和探索的权衡，目前常用的方法是最大化 UCB 启发函数。UCB 启发值在多臂赌博机（MAB）问题里面可以取得渐进最优的累积剩余值，但累积剩余值并不直接适用于序列化的蒙特卡洛树搜索问题，因为蒙特卡洛树搜索并不直接关注于前向搜索过程中每一步获得的回报值。

基于 MAXQ 分层分解的在线规划算法——MAXQ-OP，基于后验动作采样的的蒙特卡洛树搜索算法——DNG-MCTS 和 D²NG-POMCP，以及基于贝叶斯信念更新模型的多目标跟踪算法——PFS，是本文接下来要介绍的三个主要工作。

第三章　基于 MAXQ 分层分解的在线规划算法

内容提要

马尔科夫决策过程基本理论为不确定性环境下的规划提供了丰富的数学框架。然而精确求解一个 MDP 问题往往非常困难，受制于所谓“维度诅咒”的问题——即状态空间大小随状态变量的数目呈指数级增加。在线算法通过避免为整个状态空间计算完整策略来克服这个问题。但由于在线算法必须实时计算出当前状态的最优动作，所以规划时允许的计算时间往往非常有限。MAXQ 是一个 MDP 值函数分解方法，试图利用问题本身具备的分层结构来把原 MDP 问题分解成一组层次树上的更小的子问题。针对以上挑战，本章主要提出 MAXQ-OP 算法——一个新颖的结合了基于 MAXQ 分层分解技术的在线规划算法。特别地，MAXQ-OP 在大规模在线规划算法框架里面引入 MAXQ 分层分解，为复杂环境里面的自主智能体设计和利用提供了更高效的解决方案。MAXQ-OP 算法里面，分层分解使得在线规划可以更高效地搜索到搜索树上更深的节点，从而为根节点提供了更精确的值函数估计，使得根节点在动态规划框架内可以实现更优良的动作选择策略。MDP 标准测试问题——出租车问题——上的实验结果证明了 MAXQ-OP 算法的有效性。MAXQ-OP 算法同样成功应用到了科大“蓝鹰”仿真 2D 机器人足球队，为球队从 2009 年至今（2014 年）在 RoboCup 机器人世界杯比赛里面获得 4 项世界冠军和 2 项世界亚军做出了重要贡献。RoboCup 仿真 2D 机器人足球比赛里面的长期案例研究表明 MAXQ-OP 算法可以适用于规模特别巨大的实际问题。本章剩余内容主要安排如下：第3.1节主要介绍了 MAXQ-OP 算法的背景知识和主要研究动机；第3.2节介绍了一些相关工作；第3.3节详细给出了 MAXQ-OP 算法；第3.4节展示了出租车问题里面的实验结果；第3.5节介绍了 RoboCup 2D 领域里面的主要研究成果；第3.6给出了本章小结。

3.1　基本介绍

马尔科夫决策过程为不确定环境下的规划问题提供了有用的数学模型[23]。目前流行的精确求解算法，比如线性规划、值迭代、策略迭代等，都关注于离线求解 MDP。也就是说，这些算法在智能体跟环境实际交互前完整计算出整个状态空间上的策略。不幸的是，应用到大规模问题时，这些算法都受制于所谓“维度诅咒”问题——即状态空间大小随状态变量的数目呈指数级增加[31]。以本章的目标实验平台之一——RoboCup 仿真 2D 机器人足球——为例，如果只考虑位置、速度、身体角度等主要状态变量，状态空间的维度是 136。如果只把每个状态变量离散成各自取值范围内的 1000 个值，那么最后得到的状态空间大小是 10^{408}。对于如此巨大的状态空间，离线求解出完整的控制策略是不可能的。

更糟糕的是，很多实际问题的环境模型经常会随时间而变化。比如 RoboCup 2D 中，对手的策略可能随时都会变化。离线求解出的策略无法灵活应对这些变化。

幸运的是，在线规划算法尝试只在当前状态的可达状态空间内计算出局部策略来克服以上难点。在线规划算法的主要理论依据是，智能体在环境中交互时只会遇到很有限的状态。比如，完整的 RoboCup 2D 比赛包含大概 6000 个周期，所以智能体只需要在这些遇到的状态下做出决策，其数目远远小于完整的状态空间大小。具体来说，在线规划算法递归地在可达状态空间内进行前向搜索以评估当前状态下所有可行的动作，并返回能找到的“最好”动作。搜索时常常可以应用各种启发式技术以减少计算时间和内存的消耗，比如实时动态规划算法[55]，LAO*[87]，UCT[56]，DNG-MCTS[43] 等。更进一步，因为在线规划算法实时在当前状态下做出决策，所以在线规划算法可以很好的处理环境模型不可预知的变化。这使得在线规划算法成为很多实际问题的首选算法，包括本章主要考虑的实验平台之一——RoboCup 仿真 2D 机器人足球。实现在线规划算法的关键是，必须为每个当前状态在很短的时间内给出最优行动以满足实时性需求，比如 RoboCup 2D 问题中，每个决策周期只有 100 毫秒。

分层分解是另一项广泛使用的用于扩展 MDP 算法到大规模问题的技术。给定问题的分层结构，分层分解技术把原问题分解成一系列可以被更容易解决的子问题[64]。具体来说，本章使用 MAXQ 值函数分解策略进行分层分解，把原 MDP 问题的值函数分解成若干子问题值函数之和，分布在一棵树状结构之上[37]。MAXQ 分层分解受益于*时序抽象*（Temporal Abstraction）、*状态抽象*（State Abstraction）和*子任务共享*（Subtask Sharing）。时序抽象是指 MAXQ 分层结构中的高层任务把底层任务看成宏动作，在高层 MDP 中像原子动作一样处理；状态抽象是指每一个子任务只需要关注于跟当前子目标相关的状态变量，等价于只需要处理原始状态空间的一个边缘空间；子任务共享是指不同的高层子任务可以共享若干底层子任务，所以一个底层子任务的策略可以在不同地方多次重用。比如在 RoboCup 2D 问题里面，通常的进攻行为有传球、带球、射门、截球、跑位等。传球、带球、射门共享相同的踢球行为；截球和跑位共享相同的移动行为。虽然把 RoboCup 2D 中的智能体决策分解成一系列子行为的决策是比较直接的，但如何自动进行高层行为和底层行为的规划和权衡仍然是需要进一步研究的重点，也是本章主要关注的问题。

本章主要介绍基于 MAXQ 分层分解的在线规划算法——MAXQ-OP。MAXQ-OP 结合了求解大规模 MDP 在线规划和分层分解的优势。特别地，MAXQ-OP 利用底层问题分层结构的同时，进行在线规划以找到当前状态最后的行动。目前领先的在线规划算法都试图通过逐渐增加搜索深度来为当前状态找到最优动作，然而在大规模问题里面合理保持搜索的动作分支数目，并且尽量搜索到足够的深度是非常困难的。举例来说，RoboCup 仿真 2D 机器人足球比赛里面，在比赛的初始阶段，智能体常常要搜索几千步才能搜索到进球的

状态。分层分解通过使用时序扩展的动作——整体需要执行很多步的一些列原子动作的集合，使得搜索过程可以搜索到更深的节点。举例来说，如果有一个 moving-to-target 的宏动作负责智能体从当前位置移动到目标位置的中间规划过程，那么智能体就可以直接从已经到达目标点的状态开始继续搜索，而不需要考虑如何到到目标点的规划细节——这些细节假设已经由宏动作内部处理好。分层分解减少了搜索巨大不必要状态空间的计算负担，使大规模剪枝成为可能。直观上，结合了分层分解的在线规划可以覆盖搜索树中更深的部分，更有可能搜索到目标状态，从而可以提高根节点的动作选择策略。MAXQ-OP 的另一个优势是不需要为每个子任务手工编写完整的策略。事实上，只需要通过定义好活动状态、目标状态和可选的伪回报函数的方式给出层次任务树（或任务图），MAXQ-OP 自动同时在任务树上和状态空间里面进行搜索，并找到近似最优策略。MAXQ 算法框架里面，完成函数给出执行一个分层策略时，在子任务结束后，任务本身结束前，可以获得的期望累积回报。传统上，把动态规划算法里面应用 MAXQ 层次结构时，需要事先知道最优策略对应的完成函数。而这实际等价于事先求解整个任务，而这时不可能的，否则就没有再进行动态规划的需要。这是在动态规划算法里面应用 MAXQ 的主要挑战。

对一般的 MDP 模型 $\langle S, A, T, R\rangle$，本章假设存在一组目标状态的子集 $G \subseteq S$，使得对所有的目标状态 $g \in G$ 和动作 $a \in A$，有 $\Pr(g \mid g, a) = 1$，以及 $R(g, a) = 0$。可以假设任意 MDP 都可以被转化成等价的*无折扣的负回报目标导向 MDP*（Undiscounted Negative-Reward Goal-directed MDP），其中非目标状态的回报值严格为负[88]。所以无折扣目标导向的 MDP（也就是*随机最短路径*（Stochastic Shortest Path）问题[89]）实际上是一个通用的模型，也是本章主要关注的基本模型。然而本章提出的算法是可以方便的应用到其他模型的。

本章的主要贡献有两个方面：1. 在线应用分层结构的整体框架；2. 完成函数的近似计算方法。这个工作主要扩展了作者之前关于 MAXQ 分层分解在线规划的研究成果[39, 41]，为每个子任务引入*终止分布*（Terminating Distribution）——即子任务结束时可能目标状态的概率分布，并且提出了在线和离线估计这些分布的方法。在 MDP 标准测试问题——出租车问题——上的实验结果证明了 MAXQ-OP 算法的有效性和高效性。MAXQ-OP 方法被成功应用于科大“蓝鹰”（WrightEagle）仿真机器人足球队，为球队多次获得 RoboCup 机器人世界杯冠亚军做出了重要贡献。在 RoboCup 2D 仿真机器人足球上的长期案例研究表明 MAXQ-OP 有很强的适用于通常算法无法涉足的大规模问题，并同时求解出高质量行动策略的潜力。

3.2　相关工作

在在线 MDP 规划领域，实时动态规划（RTDP）是第一个通过一系列基于贪心策略和启发函数的试验搜索来为当前状态寻找最优动作的算法[55, 90–92]。相反，

AO* 算法[57, 87, 93] 通过贪心地选择展开当前最优子图的叶子节点来建立 MDP 与或树上的最优策略。蒙特卡洛树搜索（MCTS）结合使用最优优先的搜索和蒙特卡洛仿真技术来寻找近似最优策略[43, 56, 94–97]。算法的关键是通过一系列的仿真结果来评估最优优先搜索树上的状态节点。最近，*基于试验的启发式树搜索*（Trial-based Heuristic Tree Search, THTS）算法试图统一以上所有算法。THTS 把在线算法分层以下几个主要组成部分：启发函数、备份操作函数、动作选择策略、结果选择策略和试验搜索深度。在线算法关注于为当前状态计算出最优动作，但很少有在线算法利用了分层分解策略——这正是本章提出的 MAXQ-OP 算法主要关注的问题。

在强化学习研究领域，*分层强化学习*（Hierarchical Reinforcement Learning, HRL）使用分层分解技术以更快的学习得到 MDP 问题的策略[64]。常见的*状态抽象*（State Abstraction）技术为子任务删减无关状态变量，实际上把状态空间分割成一组子空间[98–104]。特别地，Sutton 等通过引入时序扩展动作，即 Option，把 HRL 建模成半马尔科夫决策过程（SMDP）[65]。每个 Option 都被通过自动学习或手工编写的方式附以一个单独的内部策略。本章介绍的工作主要基于 Dietterich 等提出的 MAXQ 值函数分层分解技术[37]。基于 MAXQ 的 HRL 方法把原始 MDP 问题分解成一组具有层次结构的 SMDP，并同时为每个子任务学习策略[105, 106]。

跟强化学习领域类似，MDP 规划领域也存在一些离线算法利用问题的层次结构来加速规划过程。举例来说，Hauskrecht 等提出*抽象 MDP 模型*（Abstract MDP）。抽象 MDP 引入宏动作和宏状态的概念，把宏状态看成是在某些状态空间区域里面行动的局部策略，并限制状态空间区域交界处的状态。*变量影响结构分析*（Variable Influence Structure Analysis, VISA）假设已知底层 MDP 的因子化表示，建立行动的动态贝叶斯网络（DBN），并构造蕴含状态变量因果关系的*因果图*（Causal Graph）来进行分层分解[107]。Barry 等提出一个 DetH* 离线规划算法来分层求解大规模 MDP 问题。DetH* 假设确定性的宏状态转移。

虽然分层分解技术在强化学习和 MDP 离线规划领域已经比较普遍的应用，但如何在在线规划算法里面应用仍然是不平凡的。主要挑战是，高层任务在规划时需要假设底层任务已经被给出了。举例来说，在机器人足球领域，智能体需要搜索射门决策（高层任务）的时候，必须要知道如何调整身体位置和角度以更好地踢球（底层任务）。然而，关于底层任务的信息在在线规划时并不清楚。就如前面已经提到的，本章通过为每个子任务引入终止概率的分布来处理这个问题。更多的细节在下一节给出。

3.3　基于 MAXQ 分层分解的在线规划

一般来说，在线规划算法交替进行规划和执行，并且仅需要为当前状态选择最优动作。给定问题的 MAXQ 分层结构，比如 $\mathrm{M} = \{\mathrm{M}_0, \mathrm{M}_1, \ldots, \mathrm{M}_n\}$，MAXQ-

OP 通过前向搜索来评估每一个子任务，并在线计算出递归值函数 $V^*(i,s)$ 和行动值函数 $Q^*(i,s,a)$。这就要求，在 MAXQ 分层结构之上，从根任务 M_0 开始一直到某个叶子节点对应的原子任务结束，进行完全的搜索。搜索结束以后，根据当前树上的行动值函数选择根任务的某个最优动作 $a \in A_0$。相应地，应该自己执行的原子动作 $a_p \in A$ 也就被决定了。执行完成 a_p 以后，环境转移到新的状态，进入新的决策周期，规划过程重复选择最优动作。MAXQ-OP 算法的主要思想是在线近似计算公式 2.27，算法的主要挑战是如何完成函数的近似计算。下一节主要给出 MAXQ-OP 算法的主要思想。

3.3.1　MAXQ-OP 算法简介

直观上，完成函数给出执行底层子任务 M_a 后继续回到执行子任务 M_i 知道结束所能获得的期望累积回报。根据公式 2.25，给定最优策略 π^* 的完成函数展开后为：

$$C^*(i,s,a) = \sum_{s',N} \gamma^N \Pr(s',N \mid s,a) V^*(i,s'), \tag{3.1}$$

其中

$$\begin{aligned} \Pr(s',N \mid s,a) = \sum\nolimits_{\langle s,s_1,\ldots,s_{N-1}\rangle} \quad & \Pr(s_1 \mid s,\pi_a^*(s)) \cdot \Pr(s_2 \mid s_1,\pi_a^*(s_1)) \\ & \ldots \Pr(s' \mid s_{N-1},\pi_a^*(s_{N-1}))\Pr(N \mid s,a). \end{aligned} \tag{3.2}$$

这里，$\langle s,s_1,\ldots,s_{N-1}\rangle$ 是服从最优策略 $\pi_a^* \in \pi^*$ 从状态 s 到终止状态 s' 长度为 N 的路径。不幸的是，计算最优策略 π^* 等价于求解整个问题。理论上，可以通过完整展开搜索树并枚举所有可能的从状态—动作对 s,a 出发终止于 s' 的状态、动作序列来确定最优路径。但这在在线规划算法框架里面，特别是针对大规模问题，是不可能实现的。

为了精确计算完成函数 $C^*(i,s,a)$，智能体必须知道最优策略 π^*。正如前文所述，这等价于求解整个问题。另一方面，由于计算时间的问题，无法在在线规划的同时找到这个最优策略。当在在线规划算法框架里面应用 MAXQ 分层分解时，必须要能够为每个子任务近似计算完成函数。一个可能的方法时离线计算好，然后在在线规划的时和使用离线计算的结果。但问题规模非常巨大的时候，离线计算也是不可能的。

给定一个最优策略，子任务最终将会以概率 1 在任意目标状态终止。注意到在本章使用的设定里面 $\gamma = 1$，所以公式 2.25的 γ^N 因子总是等于 1 的。从而，完成函数可以重新写成：

$$C^*(i,s,a) = \sum_{s'} \Pr(s' \mid s,a) V^*(i,s'), \tag{3.3}$$

其中，$\Pr(s' \mid s,a) = \sum_N \Pr(s',N \mid s,a)$ 是定义在子任务 M_i 终止状态上的边缘分布。所以，为了估计完成函数，只要估计出终止分布——$\Pr(s' \mid s,a)$——就可

Input: an MDP model with its MAXQ hierarchical structure
Output: the accumulated reward r after reaching a goal

```
r ← 0
s ← GetInitState()
while s ∉ G_0 do
    ⟨v, a_p⟩ ← EvaluateState(0, s, [0, 0, ..., 0])
    r ← r+ ExecuteAction(a_p, s)
    s ← GetNextState()
end
return r
```

算法 3.1: MAXQ-OP 分层在线规划 OnlinePlanning 算法

以。于是，对于非原子子任务，公式 2.27近似展开为：

$$V^*(i,s) \approx \max_{a \in A_i} \left\{ V^*(a,s) + \sum_{s'} \Pr(s' \mid s,a) V^*(i,s') \right\}. \tag{3.4}$$

公式 3.4虽然已经隐含了完成函数的计算，使得在线求解成为可能，但公式 3.4递归依赖于其自身，实际仍然无法在线计算。为此，MAXQ-OP 引入深度数组 d 和最大搜索深度数组 D，其中 d[i] 是当前前向搜索的深度，D[i] 是子任务 M_i 说允许的最大搜索深度。同时，在子任务 M_i 搜索深度超过最大深度时，调用启发函数 H 给出最终值函数的估计值。公式 3.4进一步展开为：

$$V(i,s,d) \approx \begin{cases} H(i,s) & \text{if } d[i] \geqslant D[i] \\ \max_{a \in A_i} \{ V(a,s,d) + & \\ \sum_{s'} \Pr(s' \mid s,a) V(i,s', d[i] \leftarrow d[i]+1) \} & \text{其他情况} \end{cases} \tag{3.5}$$

公式 3.5就给出了 MAXQ-OP 算法的主要框架，使 MAXQ 分层结构的在线计算成为可能。算法实现时，调用 $V(0,s,[0,0,\ldots,0])$ 即可以返回状态 s 在子任务 M_0 下的值函数，同时算法返回应该执行的原子动作。

实际实现 MAXQ-OP 算法时，不一定需要完整评估所有的终止状态，只采样若干终止状态就可以了。为此，假设已经估计出状态 s 和宏动作 a 对应的终止分布 $\Pr(s' \mid s,a)$，令 $G_{s,a} = \{s' \mid s' \sim \Pr(s' \mid s,a)\}$ 为终止分布中采样出来的终止状态集合，那么完成函数 $C^*(i,s,a)$ 可以估计为：

$$C^*(i,s,a) \approx \frac{1}{|G_{s,a}|} \sum_{s' \in G_{s,a}} V^*(i,s'). \tag{3.6}$$

这种情况下，公式 3.5重写为：

$$V(i,s,d) \approx \begin{cases} H(i,s) & \text{if } d[i] \geqslant D[i] \\ \max_{a \in A_i} \{ V(a,s,d) + & \\ \sum_{s' \in G_{s,a}} \frac{1}{|G_{s,a}|} V(i,s', d[i] \leftarrow d[i]+1) \} & \text{其他情况} \end{cases} \tag{3.7}$$

Input: subtask M_i, state s and depth array d
Output: $\langle V^*(i,s), \text{a primitive action } a_p^* \rangle$

```
if M_i is primitive then return ⟨R(s, M_i), M_i⟩
else if s ∉ S_i and s ∉ G_i then return ⟨−∞, nil⟩
else if s ∈ G_i then return ⟨0, nil⟩
else if d[i] ⩾ D[i] then return ⟨HeuristicValue(i, s), nil⟩
else
    ⟨v*, a_p*⟩ ← ⟨−∞, nil⟩
    for M_k ∈ Subtasks(M_i) do
        if M_k is primitive or s ∉ G_k then
            ⟨v, a_p⟩ ← EvaluateState(k, s, d)
            v ← v + EvaluateCompletion(i, s, k, d)
            if v > v* then
                ⟨v*, a_p*⟩ ← ⟨v, a_p⟩;
            end
        end
    end
    return ⟨v*, a_p*⟩
end
```

算法 3.2: MAXQ-OP 分层在线规划 EvaluateState(i, s, d) 算法

3.3.2　MAXQ-OP 算法的主要流程

算法 3.1给出了 MAXQ-OP 算法 OnlinePlanning 函数的主要流程，其中状态 s 由函数 GetInitState 初始化，函数 GetNextState 在通过函数 ExecuteAction 执行动作以后获得环境的下一个状态。整个过程一直循环，知道环境到达某一目标状态 $g \in G_0$。注意到 MAXQ-OP 的主要算法由函数 EvaluateState 实现。函数 EvaluateState 使用深度优先搜索的方式在任务树上进行搜索，返回当前状态应该执行的最优动作，将在下一节被详细介绍。

3.3.3　分层任务评估

为了能够选择最好的动作，智能体必须计算当前状态 s 下每一个可行的动作的行动值函数。这个过程就建立了一棵以当前状态 s 为根节点的搜索树，也就是与或树[54, 87]。树上的搜索一般按照最优优先的方式进行，直到遇到目标状态或达到最大搜索深度。当达到最大搜索深度时，一般使用一个启发函数来返回剩下过程的期望累积回报的估计值。

算法 3.2概括了 MAXQ-OP 利用任务层次结构搜索的主要过程。更详细地说，MAXQ-OP 通过递归评估子任务 M_i 来评估当前状态 s，估计相应的完成函数，并最终选择执行返回值最高的子任务对应的原子动作。递归在以下情况终止：1. 子任务是一个原子任务；2. 当前待展开的状态时目标状态或对当前子任务而言不是活动状态；3. 或者，已经到达某一指定的搜索深度，也就是 $d[i] \geqslant D[i]$，其

Input: subtask M_i, state s, action M_a and depth array d
Output: estimated $C^*(i, s, a)$

$G_{s,a} \leftarrow \{s' \mid s' \sim Pr(s' \mid s, a)\}$
$v \leftarrow 0$

for $s' \in G_{s,a}$ **do**
    $d' \leftarrow d$
    $d'[i] \leftarrow d'[i] + 1$
    $v \leftarrow v +$ EvaluateState(i, s', d')
end
$v \leftarrow \frac{v}{|G_{s,a}|}$

return v

算法 3.3: MAXQ-OP 分层在线规划 EvaluateCompletion(i, s, a, d) 算法

中 $d[i]$ 是当前前向搜索的深度，$D[i]$ 是子任务 M_i 说允许的最大搜索深度。注意到不同的子任务允许的最大搜索深度可以不同，比如高层子任务允许的最大搜索深度自觉上应该比底层子任务允许的搜索深度更小。如果一个子任务是原子任务，那么该原子动作和其对应的立即回报会被一起返回。搜索过程超过指定最大深度时，使用启发函数类估计未来的期望累积回报。这种情况下，会返回一个 $\mathtt{nil}$ 动作（算法实现时，已经 $\mathtt{nil}$ 动作不会被最终执行)。如果以上条件都不满足，那么函数 EvaluateState 继续循环，并且递归地评估 MAXQ 任务树上所有可选子任务。

3.3.4　完成函数近似计算

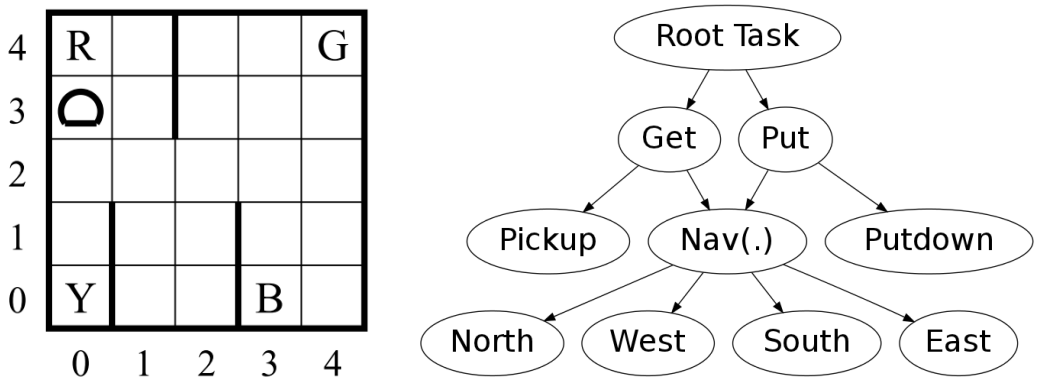
算法 3.3展示根据公式公式 3.7递归估计完成函数的主要过程。这里，首先需要为每个子任务提供终止分布。给定子任务的领域知识，可以在线或离线来估计终止分布。对具有稳定动态过程的子任务，比如机器人导航或手臂操作，离线估计是可能的——这些子任务只在到达终止状态时终止；对其他子任务，比如机器人足球中的传球、带球或射门，则在假定系统状态转移模型后，在线估计更为可取。

3.3.5　动作空间中的启发式搜索

对于状态空间很大的问题，枚举所有动作（包括宏动作和原子动作）可能会非常耗时。所以有必要引入启发式技术（包括剪枝技术）来加速搜索过程。直觉上，没有必要评估不太可能比当前已经评估过最好的动作更好的动作。在 MAXQ-OP 中，这是通过实现一个使用 NextAction 函数动态选择下一个将被评估动作的迭代版本的 Subtasks 函数完成的。注意到，NextAction 函数需要兼顾利用—探索的平衡。不同的子任务可以使用不同的启发式技术，比如 A* 搜索、爬山法、梯度法等。

Input: subtask index i and state s
Output: selected action a^*
if SearchStopped(i, s) **then**
    return nil
end
else
    $a^* \leftarrow \text{argmax}_{a \in A_i} H_i[s,a] + c\sqrt{\frac{\ln N_i[s]}{N_i[s,a]}}$
    $N_i[s] \leftarrow N_i[s] + 1$
    $N_i[s,a^*] \leftarrow N_i[s,a^*] + 1$
    return a^*
end

算法 3.4: MAXQ-OP 分层在线规划 NextAction(i, s) 算法



(a) 出租车问题　　(b) 出租车问题 MAXQ 任务图

图 3.1　出租车问题及其 MAXQ 任务图

作为例子，算法 3.4给出了一个 UCB[63] 版本的 NextAction 函数。该算法中，$N_i[s]$ 和 $N_i[s,a]$ 分别是状态 s 和状态—动作对 (s,a) 在子任务 M_i 里的访问次数，c 是利用—探索平衡因子。函数 SearchStopped 根据剪枝条件（比如最大已评估动作数目等）动态决定当前子任务的搜索过程何时应该终止。$H_i[s,a]$ 在子任务 M_i 中，状态 s 上执行动作 a 的启发值，根据领域知识初始化。

3.4　标准测试：出租车问题

出租车问题是 MDP 标准测试问题之一[37]。如图 3.1(a)所示，出租车问题由一个 5×5 的格子世界（包含墙）和 4 个出租车站台：R，G，Y 和 B 组成。与环境交互的智能体就是出租车，出租车问题的目标是接送一个乘客。系统有 4 个状态变量：出租车的坐标 x 和 y，乘客的位置 *pl*，以及乘客的目标地点 *dl*。变量 *pl* 可以四个车站中的任何一个，或者赋值成 *taxi* 表明乘客在出租车里面。变量 *dl* 只能是 4 个车站中的一个。实验中，不允许 *pl* 和 *dl* 相等。所以，除掉 *pl* 和 *dl* 相等的情况后，问题一共有 404 个状态：25 个出租车位置，5 个乘客位

表 3.1　出租车问题非原子子任务的完整定义

子任务	活动状态	终止状态	动作	最大深度
Root	所有状态	$pl = dl$	Get 和 Put	2
Get	$pl \neq taxi$	$pl = taxi$	Nav(t) 和 Pickup	2
Put	$pl = taxi$	$pl = dl$	Nav(t) 和 Putdown	2
Nav(t)	所有状态	$(x, y) = t$	North, South, East 和 West	7

表 3.2　出租车问题详细实验结果

算法	实验次数	平均回报 *	离线计算时间	平均在线计算时间
MAXQ-OP	1000	3.93 ± 0.16	-	0.20 ± 0.16 ms
LRTDP	1000	3.71 ± 0.15	-	64.88 ± 3.71 ms
AOT	1000	3.80 ± 0.16	-	41.26 ± 2.37 ms
UCT	1000	-23.10 ± 0.84	-	102.20 ± 4.24 ms
DNG-MCTS	1000	-3.13 ± 0.29	-	213.85 ± 4.75 ms
R-MAXQ	100	3.25 ± 0.50	1200 ± 50 次试验	-
MAXQ-Q	100	0.0 ± 0.50	1600 次试验	-

* 值迭代算法得到的平均回报的上界是 4.01 ± 0.15。

置，4 个目标车站位置。这个实验设置跟文献[106] 是相同的。每次测试开始的时候，出租车的位置、乘客的位置以及乘客的目标车站都是随即取值的。出租车问题在乘客顺利送达目标车站后终止。出租车一共有 6 个可选原子动作：1. 4 个导航动作，每次移动一个格子：North，South，East 和 West；2. Pickup 动作；3. Putdown 动作。每个导航动作有 0.8 的概率可以成功把智能体移动到指定的格子，0.1 的概率把智能体移动到根需要方向垂直的任意一个格子内。每个动作都有回报 -1；成功送达时，智能体有回报 20；另外非法的 Pickup 和 Putdown 动作有回报 -10。当在出租车问题里面应用 MAXQ-OP 算法时，本章使用了跟文献[37] 相同的 MAXQ 分层结构，如图图 3.1(b)所示。注意到，Nav(t) 子任务是带有参数 t 的，t 可以取值 R，G，Y 和 B 中的任何一个，表明需要前往的目标地点。非原子子任务的完整定义见表 3.1。表中，“最大深度”栏给出了每个子任务允许搜索的最大深度。

函数 EvaluateCompletion 的具体实现方法如下。对高层子任务，比如 Root，Get，Put 以及 Nav(t)，本章假设它们会以概率 1 在各自的终止状态结束；对原子动作，比如 North，South，East 和 West，本章使用底层的状态转移模型 $T(s' \mid s, a)$ 来采样一个动作作为可能的终止状态。对每一个非原子子任务，函数 HeuristicValue 被设计成返回出租车当前位置到子任务终止状态的位置之间的 Manhattan 距离的负数和该过程中可能立即回报之和。比如，Get 子

任务的启发值为：$-\text{Manhattan}((x,y),pl)-1$，其中 $\text{Manhattan}((x_1,y_1),(x_2,y_2))$ 给出两个位置 (x_1,y_1) 和 (x_2,y_2) 之间的 Manhattan 距离：$|x_1-x_2|+|y_1-y_2|$。

本章同时实现了一个基于缓存的剪枝策略以更高效的进行子任务共享。更精确地说，如果状态 s 在子任务 M_i 里被评估过，并且评估的时候搜索深度为 0，即 $d[i]=0$，假设评估的结果是 $\langle v,a_p\rangle$，那么就把这个结果存储到一个缓存表里面：

$$cache[i,hash(i,s)] \leftarrow \langle v,a_p\rangle,$$

其中 $cache$ 是缓存表，并且 $hash(i,s)$ 是关于状态 s 和子任务 M_i 的哈希函数。等到下次搜索过程中，需要在相同的条件下评估状态 s 的时候，直接以 0.9 的概率返回已经缓存的值。这个策略可以导致搜索树上的大量剪枝。主要观察是，如果一个子任务被完整评估过，那么在搜索过程中就没有必要重复评估，除非能找到更好的在线策略。

实验中，随机选择初始状态，多次运行 MAXQ-OP 算法和相关比较算法，并报告所有运行过程中的评价累积回报值和在线规划时间，如表 3.2所示。表中，离线时间是离线算法计算收敛所需要的时间，在线时间是在线算法从初始状态开始直到问题结束所有的在线规划时间。LRTDP[90]，AOT[57]，UCT[56] 和 DNG-MCTS[43] 都被实现成了基于试验搜索的 Anytime 在线规划算法。每次动作搜索时的迭代次数设定为 100；最大搜索深度也为 100。LRTDP 和 AOT 使用基于 *min-min*[90] 策略的启发函数来初始化节点；类似地，UCT 和 DNG-MCTS 使用基于 *min-min* 启发函数的贪心策略来作为基本 Rollout 策略。注意到，UCT 和 DNG-MCTS 是蒙特卡洛树搜索算法，仅依赖于出租车问题的仿真器，而不需要完整的 MDP 模型。R-MAXQ 和 MAXQ-Q 是分层强化学习算法，他们的实验结果是从文献[106] 摘录的。所有的实验都运行在一台 Linux 3.8 电脑，双核 CPU，主频 2.90 GHz，内存 8GB。可以从实验结果看出，MAXQ-OP 可以在出租车问题里面找到平均值函数为 3.93 ± 0.16 的近似最优策略，非常接近于该问题的最优解 4.01 ± 0.15。特别地，MAXQ-OP 算法需要的在线规划时间跟其他在线规划算法相比非常少。这些比较实验验证了 MAXQ-OP 算法利用问题分层结构的同时进行在线规划的有效性。

3.5　案例研究：RoboCup 仿真 2D 机器人足球

作为 RoboCup 机器人世界杯中最早的一个比赛项目，仿真 2D 机器人足球比赛已经取得了很大的成功，并且启发了很多研究学者从事相关研究，并每年参加世界杯比赛或各个国家的公开赛[108, 109]。这个研究领域，已经发表了几百篇研究论文。* 跟其他 RoboCup 中其他比赛项目相比，RoboCup 仿真 2D 机器人足球的主要特点是 RoboCup 2D 中比赛服务器（Simulator）抽象了很多决策无关

*由 Peter Stone 收集的论文列表见：http://www.cs.utexas.edu/~pstone/tmp/sim-league-research.pdf。

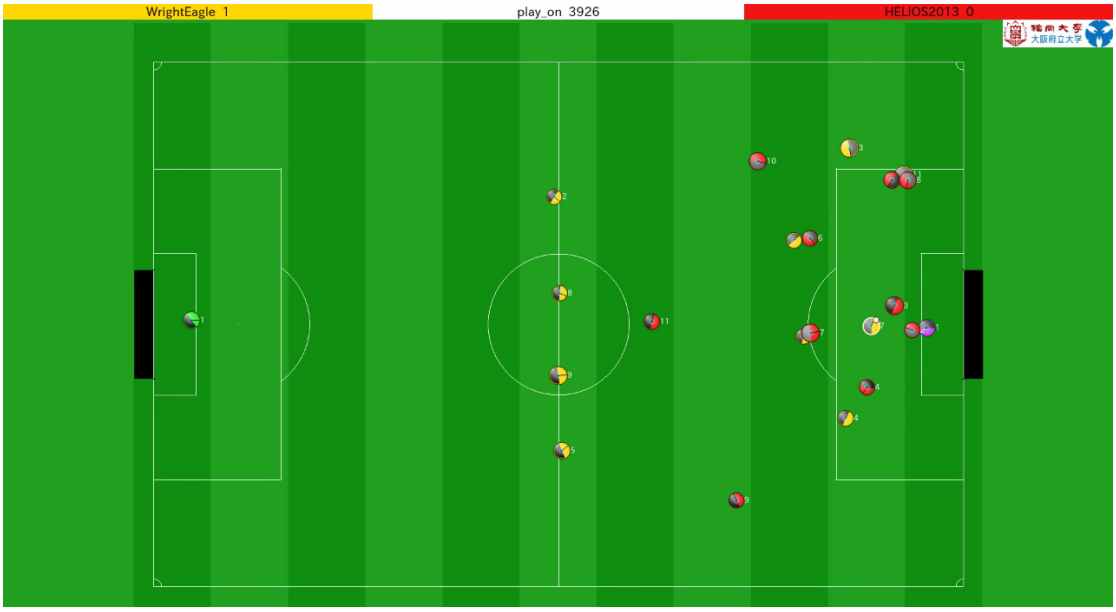


图 3.2　RoboCup 2013 仿真 2D 组决赛截图——WrightEagle 对 Helios

的细节，使得研究者可以关注于决策问题（包括规划、学习和合作）本身，而不用过多考虑机器人底层功能，比如物体识别、通信实现、硬件细节等[110]。作为例子，RoboCup 2013 仿真 2D 组的决赛——科大“蓝鹰”（WrightEagle）对日本 Helios——的截图见图 3.2。

举例来说，文献[111] 做了很多在 RoboCup 仿真 2D 中使用强化学习方法的工作。他们的方法使用基于线性 Tile-Coding 函数近似技术的半马尔科夫决策过程 Sarsa(λ) 算法在 Keepaway 子任务中学习高层策略。具体来说，他们的智能体独立学习何时应该控球，何时应该把球传给其他队友。最近，他们把这个工作扩展到了更通用的半场进攻子任务[112]。同样基于强化学习，文献[113] 提出了一系列高效学习底层策略（如截球、抢断等）的学习算法。

本节主要介绍将 MAXQ-OP 应用到 RoboCup 仿真机器人足球队中的长期研究成果。MAXQ-OP 算法在科大“蓝鹰”仿真 2D 机器人足球队中得到了实现，并取得了成功应用，为球队从 2009 年至今在 RoboCup 机器人世界杯比赛里面获得 4 项世界冠军和 2 项世界亚军做出了重要贡献。下面几小节详细介绍应用 MAXQ-OP 到 RoboCup 2D 的技术细节。

3.5.1　RoboCup 仿真 2D 机器人足球简介

RoboCup 仿真 2D 机器人足球中，一个服务器（Server）负责实时模拟 2 维足球场的所有数据，包括当前比赛进程，球的状态，以及每个球员的状态。两个队各 11 名球员通过网络接口连接到服务器上进行一场大约 6000 周期的比赛。每个球员独立跟服务器进行交互，每个周期的交互过程包括：1. 从服务器接受观察集合；2. 做出决策；3. 发送动作指令给服务器。每个球员获得的观察仅包含其视野范围内的带噪音的局部几何信息，比如球、其他球员或地标的相对距离

或角度。可以发送的动作指令即位系统的原子动作，包括把身体或脖子转到某个角度，向某个方向施加一个加速度，把球按照某个速度提到耨个方向等。这些动作也都是带有不确定性噪音的。RoboCup 2D 的主要挑战在于其是带有一个连续状态、观察和动作空间的完全分布式的多智能体系统。

3.5.2 RoboCup 仿真 2D 机器人足球建模成 MDP

通常来讲，RoboCup 仿真 2D 机器人足球实际上是一个连续状态、动作和观察空间的部分可观察随机多智能体系统。本章将 RoboCup 2D 建模成一个 MDP 问题，其中状态、动作、转移函数、回报函数的定义如下：

状态空间　RoboCup 仿真 2D 机器人足球中的联合状态使用一个高维向量来表示：$\mathbf{s} = (s_0, s_2, \ldots, s_{22})$，包含 23 个物体的信息，包括智能体自己（$s_u, u \in [1, 11]$），十个队友球员（$\{s_1, \ldots, s_{11}\} - s_u$），十一个个对手球员（$\{s_{12}, \ldots, s_{22}\}$），以及球的状态（$s_0$）。对每个球员，状态变量包括 $s = (x, y, \dot{x}, \dot{y}, \alpha, \beta)$，其中 (x, y)，$(\dot{x}, \dot{y})$，α 和 β 分别为位置、速度、身体角度、脖子角度。；对于球，状态变量包括 $s = (x, y, \dot{x}, \dot{y})$。为了简单起见，对于球员而言，其他的次要信息，比如体力值、视觉宽度等没有表示在状态里面。如前文所述，最终的状态向量有 136 维，如果把每个状态变量离散成各自取值范围内的 1000 个值，状态空间的大小是 10^{408}，远远大于通常文献里面研究的问题。

动作空间　原子动作由 RoboCup 2D 服务器定义并实现，包括 `dash`，`kick`，`tackle`，`turn` 和 `turn_neck`。这些原子动作都带有连续参数，导致了连续的动作空间。`dash` 动作可以把智能体往某一指定的方向移动特定的距离；`kick` 动作在球上施加某一指定的加速度；`tackle` 动作跟 `kick` 类似，但作用的范围更广，并且智能体铲球以后有十个周期不能移动；`turn` 和 `turn_neck` 动作分别改变智能体的身体角度和脖子角度。

转移函数　原子动作的转移模型完全由服务器给定。以 `dash` 动作为例，`dash` 动作有两个参数：$power \in [0, 1]$ 和 $angle \in [0, 2\pi)$。假设 $power = p$ 并且 $angle = \theta$，则加速度为：$(\ddot{x}, \ddot{y}) = (pA\cos\theta, pA\sin\theta) + r_a$，其中 $A = 1.0\text{m/s}^2$ 是最大加速度，r_a 是一个噪音项。物理状态按如下公式更新：$(x, y) \leftarrow (x, y) + (\dot{x}, \dot{y}) + (\ddot{x}, \ddot{y})$，以及 $(\dot{x}, \dot{y}) \leftarrow (\dot{x}, \dot{y})\omega + (\ddot{x}, \ddot{y})\omega$，其中 $\omega = 0.4$ 是速度衰减因子。本章通过把其他球员看成是环境的一部分，并假设他们的先验行为模型：如果球可踢就随机踢球，否则就随机游走，把不可预知的联合状态转移函数转化成可建模的转移函数，并把多智能体环境建模成单个智能体的 MDP 模型。

回报函数　根据 RoboCup 2D 的原始定义，智能体所在球队进球时回报为 +1；丢球时回报为 -1；其他情况回报都为 0。这个回报函数非常“稀疏”：智能体可能在进球或丢球前的几千个周期都没有任何回报。为了克服这个难点以获得富含更多信息量的在线搜索，在实现 MAXQ-OP 算法的时候，为每一个子任务引入了一系列伪回报函数和启发函数。举例来说，对导航子任务，`dash` 和 `turn` 动作的回报被定义为 -1，以保证导航子任务可以规划处最快把智能体移动到目标点的决策；对传球和带球子任务，使用基于推进速度的启发函数来估计终端节点，使得可以更快速的进攻。

3.5.3　状态估计

为把 RoboCup 建模成假设状态完全客观察的 MDP，智能体必须要克服只能获得带噪音的局部观察这个难点，估计出足够精确的当前环境状态。在 WrightEagle 中，这是通过估计出信念状态来实现的[24]。信念状态 b 就是状态空间上的概率分布，$b(\mathbf{s})$ 表示当前环境状态为 $\mathbf{s}$ 的概率。假设场上对象（包括球和球员）是条件独立的，那么 $b(\mathbf{s})$ 可以展开表示为：

$$b(\mathbf{s}) = \prod_{0 \leqslant i \leqslant 22} b_i(\mathbf{s}[i]), \tag{3.8}$$

其中 $\mathbf{s}$ 是完整的状态向量，$\mathbf{s}[i]$ 是状态向量中对象 i 的状态分量，$b_i(\mathbf{s}[i])$ 是对象 $\mathbf{s}[i]$ 的边缘分布。分别使用一个包含 m_i 加权的状态样本（或*粒子*）来近似表达边缘分布 b_i，如下：

$$b_i(\mathbf{s}[i]) \approx \{\mathbf{x}_{ij}, w_{ij}\}_{j=1\ldots m_i}, \tag{3.9}$$

其中，$\mathbf{x}_{ij}$ 是对象 i 采样出来的一个状态样本，w_{ij} 表示该样本的权重（即概率），并且有 $\sum_{1 \leqslant j \leqslant m_i} w_{ij} = 1$。

在每一个周期，每个对象的状态样本通过 RoboCup 2D 问题的*运动模型*（Motion Model）和*感知模型*（Sensor Model）进行蒙特卡洛更新[114, 115]。运动模型完全由前文所述的联合状态转移函数决定。感知模型有服务器给出。最终，环境的当前状态 $\mathbf{s}$ 可以被估计为：

$$\mathbf{s}[i] = \sum_{1 \leqslant j \leqslant m_i} w_{ij} \mathbf{x}_{ij}. \tag{3.10}$$

根据实际比赛中获得的实验结果，以上方法估计出来的状态是精确并且健壮的，特别是对智能体自身和附近对象的状态估计。表 3.3 报告了智能体自身状态估计的平均误差。注意到，`neck_dir` 动作的平均误差几乎为零，这是因为 RoboCup 2D 服务器设计 `turn_neck` 这个动作时没有加入误差。图 3.3显示了某次比赛的某个周期，智能体信念状态中其他球员的位置分布。图中，WrightEagle 在左半场，智能体自身由一个白色的小圆表示。当时，智能体自己正在带球，并且观察对方禁区情况。所以对方禁区里面球员的位置分布比较准

表 3.3　智能体自身估计状态的评价误差

x (m)	y (m)	$\dot{x}$ (m/s)	$\dot{y}$ (m/s)	α (Deg)	β (Deg)
0.047	0.046	0.0014	0.0013	0.44	1.5e-6

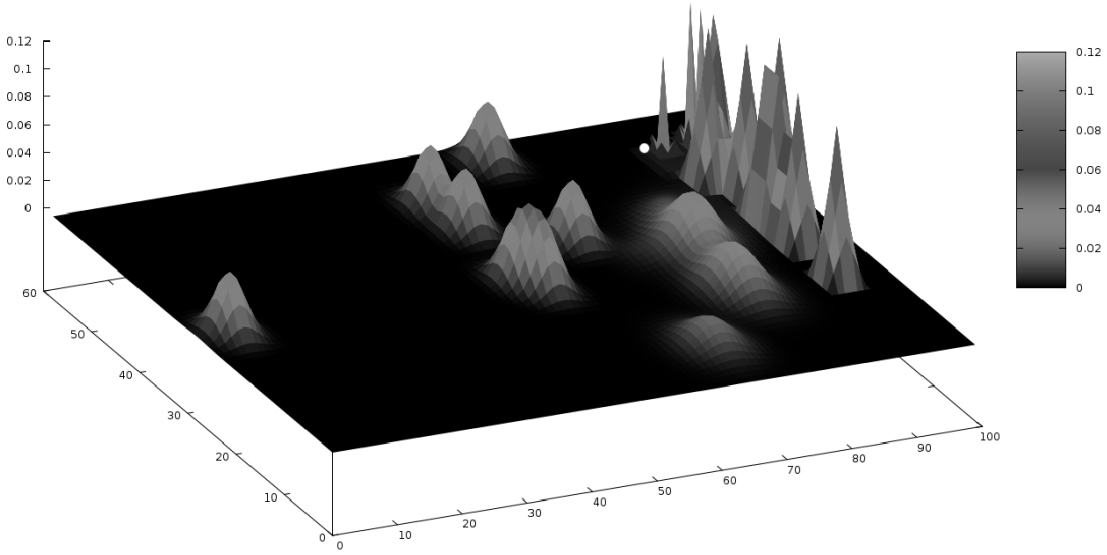


图 3.3　智能体信念状态中其他球员的位置分布

确，而后方场地上球员位置分布则比较模糊。另外，球和智能体自身的位置分布没有在图中画出，因为这两个分布相比其他分布而言更加准确，非常高并且窄，不适合跟其他球员分布画在一个图中。

3.5.4　MAXQ-OP 解决方案

本节主要介绍如何把 MAXQ-OP 应用到 RoboCup 仿真机器人足球中。首先定义了一系列分层子任务作为 MAXQ 分层结构的基本组成部分，列举如下：

- `kick`，`turn`，`dash` 和 `tackle`：这些都是由服务器定义好的原子动作。每个原子动作执行的时候都有回报 -1，以保证最优的策略可以按照最快的方式到到目标状态。

- `KickTo`，`TackleTo` 和 `NavTo`：`KickTo` 和 `TackleTo` 子任务的目标以一定的速度是把踢或铲到给定的方向；`NavTo` 子任务的目标是把智能体从当前位置移动到给定的目标地点。`KickTo` 会规划出 `kick` 动作踢球，以及 `turn` 动作调整智能体的身体角度以保证能以最轻松的方式把球踢出去；`NavTo` 动作通过规划出 `dash` 和 `turn` 动作来移动智能体。注意到为了把球按照要求的方式踢出去，可能有必要规划出多步的 `turn` 和/或 `kick` 动作。如果球不再可踢或可铲，子任务 `KickTo` 和 `TackleTo` 终止；如果智能体已经到到目标地点，子任务 `NavTo` 终止。

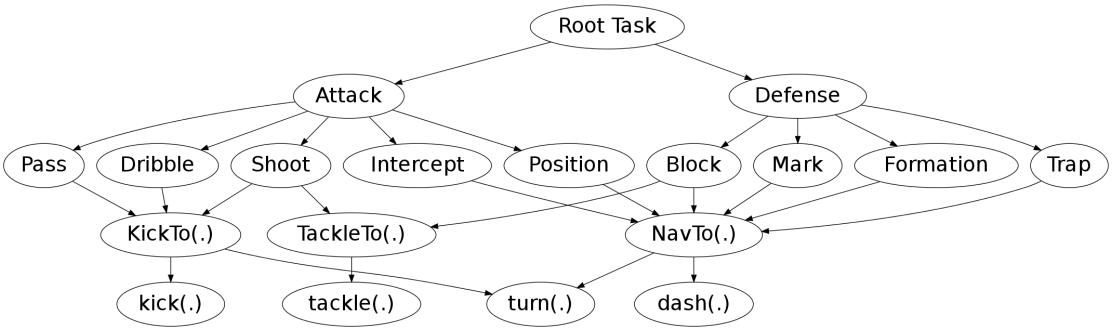


图 3.4　科大“蓝鹰”MAXQ 分层分解任务图

- `Shoot`, `Dribble`, `Pass`, `Position`, `Intercept`, `Block`, `Trap`, `Mark` 和 `Formation`：这些子任务在球队中又被称为（高层）行为，其中：1. `Shoot` 的目标是把球踢进对方球门；2. `Dribble` 的目标是在某一方向上带球前进；3. `Pass` 的目标是把球传给合适的队友；4. `Position` 的目标是保持队伍阵形，同时负责配合己方控球队友以创作出更多进攻机会；5. `Intercept` 的目标是尽快地截到球；6. `Block` 的目标是封堵对方控球球员；7. `Trap` 是尽量从对方球员脚下把球强夺过来；8. `Mark` 的目标是盯防相关对方球员；9. `Formation` 的目标是维护好防守阵形。`Shoot`, `Dribble` 和 `Pass` 的活动状态要求球对智能体而言球可踢或可铲。`Shoot`, `Dribble` 和 `Pass` 在球不再可踢或可铲的时候终止；`Intercept` 在球可踢或可铲的时候终止；`Position` 在球可踢，或被对方控球的时候终止；其他的防守行为，在球被队友截到时终止。

- `Attack` 和 `Defense`:`Attack` 子任务的目标是进攻并最终进球；`Defense` 子任务的目标是防守，以阻止对方进球。`Attack` 在球被对方截到时终止，`Defense` 在球被队友截到时终止（包括智能体自己）。

- `Root`：这个是根任务。`Root` 首先会评估 `Attack` 子任务，如果 `Attack` 无解，会继续规划 `Defense` 子任务。

MAXQ 分层结构的任务图见图 3.4，其中子任务名后面的括号表明这个子任务是带有参数的。以任务 `Attack`，`Pass` 和 `Intercept` 作为例子，假设智能体踢球的时候身体角度已经调整好了，所以 `KickTo` 子任务仅需要规划 `kick` 动作，令 s 表示当前的联合状态，根据公式 2.26，公式 2.27 和 公式 3.3，有：

$$\begin{aligned} Q^*(\texttt{Root}, s, \texttt{Attack}) = V^*(\texttt{Attack}, s) + \\ \sum_{s'} \Pr(s' \mid s, \texttt{Attack}) V^*(\texttt{Root}, s'), \end{aligned} \tag{3.11}$$

$$V^*(\texttt{Root}, s) = \max\{Q^*(\texttt{Root}, s, \texttt{Attack}), Q^*(\texttt{Root}, s, \texttt{Defense})\}, \tag{3.12}$$

$$V^*(\texttt{Attack}, s) = \max\{Q^*(\texttt{Attack}, s, \texttt{Pass}), Q^*(\texttt{Attack}, s, \texttt{Dribble}), Q^*(\texttt{Attack}, s, \texttt{Shoot}), Q^*(\texttt{Attack}, s, \texttt{Intercept}), Q^*(\texttt{Attack}, s, \texttt{Position})\}, \tag{3.13}$$

$$Q^*(\texttt{Attack}, s, \texttt{Pass}) = V^*(\texttt{Pass}, s) + \sum_{s'} \Pr(s' \mid s, \texttt{Pass}) V^*(\texttt{Attack}, s'), \tag{3.14}$$

$$Q^*(\texttt{Attack}, s, \texttt{Intercept}) = V^*(\texttt{Intercept}, s) + \sum_{s'} \Pr(s' \mid s, \texttt{Intercept}) V^*(\texttt{Attack}, s'), \tag{3.15}$$

$$V^*(\texttt{Pass}, s) = \max_{\text{position } p} Q^*(\texttt{Pass}, s, \texttt{KickTo}(p)), \tag{3.16}$$

$$V^*(\texttt{Intercept}, s) = \max_{\text{position } p} Q^*(\texttt{Intercept}, s, \texttt{NavTo}(p)), \tag{3.17}$$

$$Q^*(\texttt{Pass}, s, \texttt{KickTo}(p)) = V^*(\texttt{KickTo}(p), s) + \sum_{s'} \Pr(s' \mid s, \texttt{KickTo}(p)) V^*(\texttt{Pass}, s'), \tag{3.18}$$

$$Q^*(\texttt{Intercept}, s, \texttt{NavTo}(p)) = V^*(\texttt{NavTo}(p), s) + \sum_{s'} \Pr(s' \mid s, \texttt{NavTo}(p)) V^*(\texttt{Intercept}, s'), \tag{3.19}$$

$$V^*(\texttt{KickTo}(p), s) = \max_{\text{power } a,\ \text{angle } \theta} Q^*(\texttt{KickTo}(p), s, \texttt{kick}(a, \theta)), \tag{3.20}$$

$$V^*(\texttt{NavTo}(p), s) = \max_{\text{power } a,\ \text{angle } \theta} Q^*(\texttt{NavTo}(p), s, \texttt{dash}(a, \theta)), \tag{3.21}$$

$$Q^*(\texttt{KickTo}(p), s, \texttt{kick}(a, \theta)) = R(s, \texttt{kick}(a, \theta)) + \sum_{s'} \Pr(s' \mid s, \texttt{kick}(a, \theta)) V^*(\texttt{KickTo}(p), s'), \tag{3.22}$$

$$Q^*(\texttt{NavTo}(p), s, \texttt{dash}(a, \theta)) = R(s, \texttt{dash}(a, \theta)) + \sum_{s'} \Pr(s' \mid s, \texttt{dash}(a, \theta)) V^*(\texttt{NavTo}(p), s'). \tag{3.23}$$

如前文所述，$R(s, \texttt{kick}(a, \theta)) = -1$，并且 $\Pr(s' \mid s, \texttt{kick}(a, \theta))$ 完全由服务器给定。子任务 $\texttt{KickTo}(p)$ 在球大致向位置 p 运动时成功终止。所以 公式 3.20 给出了球运动到位置 p 所需要期望周期数的负数。子任务 `Pass` 在智能体不再

能踢球时终止，终止后，控制权返回给 Attack 任务。Attack 任务随后会规划智能体是否应该 Intercept 以防球时不小心滑出去的或 Position 以更好的维护进攻阵形。子任务 NavTo(p) 在智能体到大致达目标位置 p 时终止。类似地，$R(s, \texttt{dash}(a, \theta)) = -1$，并且 $\Pr(s' \mid s, \texttt{dash}(a, \theta))$ 是由服务器定义的。公式 3.21 给出智能体移动到目标位置 p 所需期望周期数的负数。公式 3.17 给出智能体截球需要的期望周期数的负数。Attack 行为在球被队友截到时终止。终止后，控制权返回到 Root 任务。Root 任务会继续规划 Defense 行为。Defense 行为在球被智能体或任意队友截到后终止。

为了估计出行为的终止概率，需要估计出运行中的球被任意一个球员 p（自己队或对方球员）截到的概率——这是所有子任务终止概率的基本组成部分。令 $b = (b_x, b_y, b_{\dot{x}}, b_{\dot{y}})$ 表示球的状态，并且令 $p = (p_x, p_y, p_{\dot{x}}, p_{\dot{y}}, p_\alpha, p_\beta)$ 表示球员 p 的状态。令 $\Pr(p \leftarrow b \mid b, p)$ 为球员 p 成功截到球的概率。形式上，$\Pr(p \leftarrow b \mid b, p) = \max\{\Pr(p \leftarrow b, t \mid b, p)\}$，其中 $\Pr(p \leftarrow b, t \mid b, p)$ 是球员 p 在现在开始算的周期 t 能截到球的概率，估计成：$\Pr(p \leftarrow b, t \mid b, p) = g(t - f(p, b_t))$，其中 b_t 是球在周期 t 的大致位置，$f(p, b_t)$ 函数返回球员从当前位置 (p_x, p_y) 移动到球在周期 t 时的位置所需要的期望周期数，并且 $g(\delta)$ 给出球和球员分别到达位置 b_t 具有周期差为 δ 时，球员能截到球的概率。图 3.5展示了截球概率函数 $g(\delta)$ 的大致形状。给定了截球概率的计算方法，就可以进一步估计其他子任务的终止概率，举例来说：

$$\Pr(s' \mid s, \texttt{Attack}) = 1 - \prod_{\text{opponent } o} (1 - \Pr(o \leftarrow b \mid b, o)), \tag{3.24}$$

$$\Pr(s' \mid s, \texttt{Defense}) = 1 - \prod_{\text{teammate } t} (1 - \Pr(t \leftarrow b \mid b, t)), \tag{3.25}$$

$$\Pr(s' \mid s, \texttt{Intercept}) =$$
$$\mathbf{1}[\exists \text{player } i : i \leftarrow b] \Pr(i \leftarrow b \mid b, i) \prod_{\text{player } p \neq i} (1 - \Pr(p \leftarrow b \mid b, p)), \tag{3.26}$$

$$\Pr(s' \mid s, \texttt{Position}) =$$
$$\mathbf{1}[\exists \text{non-teammate } i : i \leftarrow b] \Pr(i \leftarrow b \mid b, i) \prod_{\text{player } p \neq i} (1 - \Pr(p \leftarrow b \mid b, p)), \tag{3.27}$$

其中 $b = s[0]$ 是球的状态。

MAXQ 任务层次隐含了状态抽象。举例来说，只有智能体自身的状态和球的状态跟公式 3.20和公式 3.17是相关的。当枚举 kick 和 dash 动作的角度和加速度参数时，仅考虑了离散化以后的值。这导致了最终策略精度的下降，然而考虑到实时性要求，同时为了处理连续动作空间，离散化是必须的。为了进一步处理大动作空间的问题，实现 MAXQ-OP 时还引入了一系列启发式方法。依赖于不同子任务的特性，启发式方法的选择时比较多的。比如，Pass 行为使用爬山法来搜索 KickTo 动作空间；NavTo 子任务使用 A* 方法来搜索 dash 和

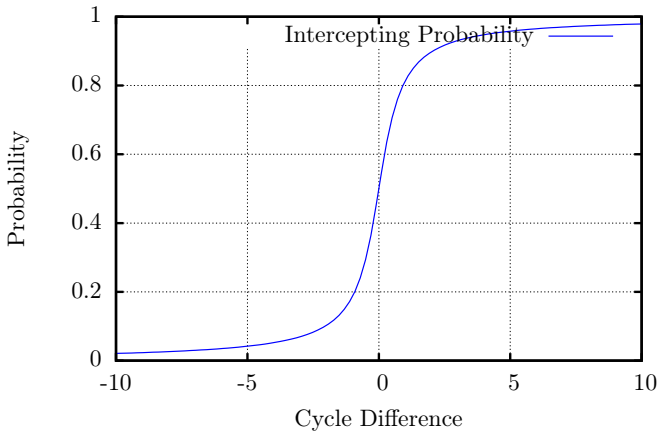


图 3.5　截球概率估计

`turn` 动作空间。

应用 MAXQ-OP 的另一个重要组成部分是在搜索超过预设的深度时使用启发函数来估计子任务值函数。以 `Attack` 任务为例，引入*推进速度*（Impelling Speed）来估计值函数 $V^*(\texttt{Attack}, s_t)$，其中 s_t 是需要评估的 t 周期以后的环境状态。假设当前联合状态为 s，需要评估的联合状态为 s'，推进速度定义如下：

$$\mathrm{impelling\_speed}(s, s', \alpha) = \frac{\mathrm{dist}(s, s', \alpha) + \mathrm{pre\_dist}(s', \alpha)}{\mathrm{step}(s, s') + \mathrm{pre\_step}(s')}, \tag{3.28}$$

其中，α 是一个全局进攻方向（又称 Aim Angle），$\mathrm{dist}(s, s', \alpha)$ 是球在进攻方向 α 上从状态 s 到 s' 移动的投影距离，$\mathrm{step}(s, s')$ 是估计的球运行的周期数，$\mathrm{pre\_dist}(s')$ 给出球可以在状态 s' 开始继续运行直到被对手截到之前，在 α 方向上可以运行的投影距离，$\mathrm{pre\_step}(s')$ 是对应的期望周期数。状态 s 上的进攻角度 α 由函数 $\mathrm{aim\_angle}(s)$ 决定。进一步，$V^*(\texttt{Attack}, s)$ 按下式进行估计：

$$V^*(\texttt{Attack}, s_t) = \mathrm{impelling\_speed}(s_0, s_t, \mathrm{aim\_angle}(s_0)), \tag{3.29}$$

其中，s_0 上当前状态，也就是搜索树根节点对应的状态。按照如上定义，推进速度的值 $\mathrm{impelling\_speed}(s_0, s_t, \mathrm{aim\_angle}(s_0))$ 隐含了球在给定进攻方向上推进越快，进攻机会可能越多，状态 s_t 也就越有价值的事实。

3.5.5　实验评估

为了测试 MAXQ-OP 算法框架在球队中的最终效果，本章将其与三个不同版本的球队进行了比较：

- Full: 这就是球队的完整版本，其中完整实现了 MAXQ-OP 在线分层规划算法框架。
- Random: 这个版本几乎等同于 Full 版本，除了球可踢但不能射门时，`Attack` 行为随机选择 `Pass` 和 `Dribble` 行为之外。

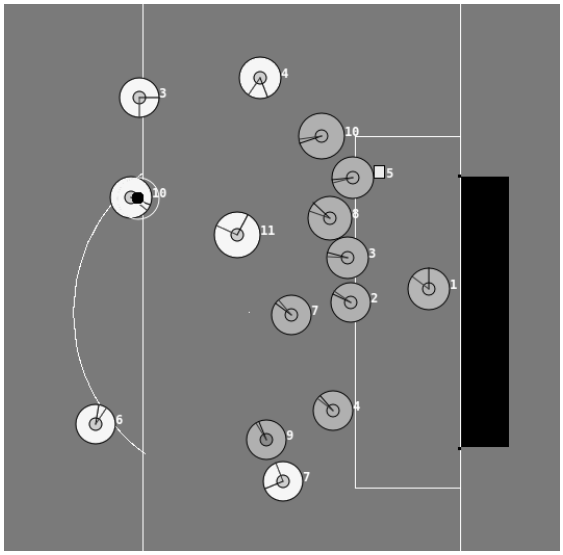


图 3.6　RoboCup 2011 仿真 2D 组决赛中选取的一个测试场景

- Hand-coded: 这个版本跟 Random 类似，但不是随机选择 `Pass` 和 `Dribble` 行为，而是使用了一个手工编写的策略。该策略下，如果智能体 3 米以内没有对手就带球；否则就传球。

Full，Random 和 Hand-coded 版本的唯一区别就在于 `Attack` 行为选择 `Pass` 和 `Dribble` 的局部策略不同。Full 版本的策略选择是 MAXQ-OP 分层规划自动完成的，也就是由函数 EvaluateState(`Pass`, ·, ·) 和 EvaluateState(`Dribble`, ·, ·) 的比较结果得到的。虽然 Random 和 Hand-coded 版本由不同的 `Pass-Dribble` 选择策略，其他的子任务，包括 `Shoot`，`Pass`，`Dribble`，`Intercept` 和所有的防守行为跟 Full 版本是完全一致的。

对每一个版本，离线教练（也就是 RoboCup 2D 中的 Trainer）被用来重复运行 100 次该版本跟 Helios2011（RoboCup 2011 比赛的亚军）的试验比赛（Episode）。每个试验比赛都以 RoboCup 2011 比赛中截取的一个相同的初始场景开始。试验比赛在以下情况终止：1. WrightEagle 进球，记为 **success**；2. 球的 x 坐标小于 -10.0，即在我方半场，记为 **failure**；3. 该试验持续周期超过 200 周期，记为 **timeout**。注意到，虽然每个试验比赛都以相同初始场景开始，但由于动作和观察的不确定性，他们中没有完全相同的。

图 3.6展示了每次试验比赛开始的初始场景，对应于 RoboCup 2011 比赛中的第 #3142 周期，其中白色的圆表示我方球员，灰色的圆表示对方球员，小黑点表示球。可以看到，该周期，我方 10 号球员正在持球，对方九个球员（包括守门员）正在禁区里面封赌。RoboCup 2011 的那场比赛里面，我方 10 号把球传给了我方 11 号球员。11 号持球时，带球几周期之后，把球重新传给了 10 号。当 11 号移动到了比较好的进攻位置时，10 号又把球传给了 11 号。最终，11 号在 #3158 周期完成射门，5 周期以后进球得分。

表 3.4　WrightEagle 重复性试验测试实验结果

版本	试验次数	Success	Failure	Timeout
Full	100	28	31	41
Random	100	15	44	41
Hand-coded	100	17	38	45

表 3.5　WrightEagle 完整比赛实验结果

对方球队	比赛场数	平均进球	平均积分	获胜概率
BrainsStomers08	100	3.09 : 0.82	2.59 : 0.28	$82.0 \pm 7.5\%$
Helios10	100	4.30 : 0.88	2.84 : 0.11	$93.0 \pm 5.0\%$
Helios11	100	3.04 : 1.33	2.33 : 0.52	$72.0 \pm 8.8\%$
Oxsy11	100	4.97 : 1.33	2.79 : 0.16	$91.0 \pm 5.6\%$

表 3.4报告了实验结果。结果显示，Full 版本相比于 Random 和 Hand-coded 的成功概率分别提高了 86.7% and 64.7% 。Random 和 Hand-coded 的实验结果其实非常接近，原因是他们都共享了，除了 `Pass-Dribble` 选择策略之外，跟 Full 版本完全相同的层次子任务。实验结果表明，`Pass` 和 `Dribble` 之间的局部选择策略在 `Attack` 的决策里面起到了关键作用，显著的影响了最终的球队性能。实验结果还表明，基于 MAXQ-OP 的 `Pass` 和 `Dribble` 局部选择策略使得 `Attack` 行为获得了高水平的性能，这对于其他子任务也递归成立，比如 `Defense`，`Shoot`，`Pass` 等。作为总结，MAXQ-OP 是 WrightEagle 在重复性试验测试中取得成功的重要原因。

本章同时将 Full 版本与 RoboCup 2D 中的 4 个高质量球队进行了完整比赛的测试。这 4 个球队包括：BrainsStomers08，Helios10，Helios11 和 Oxsy11。BrainStormers08 和 Helios10 分别是 RoboCup 2008 和 RoboCup 2010 的冠军。实验中，独立运行 100 次 WrightEagle 跟其他球队的比赛，并报告比赛结果。表 3.5 概括了详细的实验结果。获胜概率定义为 $p = n/N$，其中 n 是球队获胜的次数，N 是总共的比赛场次。实验结果中，可以看到 WrightEagle 跟其他队伍比赛的获胜概率很高，分别有 82.0%，93.0%，83.0% 和 91.0% 的概率可以赢 BrainsStomers08，Helios10，Helios11 和 Oxsy11。表 3.6 报告了科大”蓝鹰“在 RoboCup 机器人世界杯—仿真 2D 机器人足球比赛中 2005 年到 2013 年的历史数据统计结果。

导致一个球队成功的因素可能有很多，本章的主要观察是 WrightEagle 很大程度上受益于 MAXQ 分层结构，以及该结构上的状态和动作抽象。应用 MAXQ-OP 在 RoboCup 仿真 2D 机器人足球中的关键优势在于提供了一个在线分层规划的原理性决绝方案。给定任务层次结构，智能体可以自动进行在线搜

表 3.6　科大“蓝鹰”在 RoboCup 2D 比赛历史数据统计（2005-2013 年）

比赛	场数	积分	进球	赢	平	输	平均得分	平均进球
RoboCup 2005	19	47	84 : 16	15	2	2	2.47	4.42 : 0.84
RoboCup 2006	14	38	57 : 6	12	2	0	2.71	4.07 : 0.43
RoboCup 2007	14	34	125 : 9	11	1	2	2.42	8.92 : 0.64
RoboCup 2008	16	40	74 : 18	13	1	2	2.50	4.63 : 1.13
RoboCup 2009	14	36	81 : 17	12	0	2	2.57	5.79 : 1.21
RoboCup 2010	13	33	123 : 7	11	0	2	2.54	9.47 : 0.54
RoboCup 2011	12	36	151 : 3	12	0	0	3.00	12.6 : 0.25
RoboCup 2012	21	58	104 : 18	19	1	1	2.76	4.95 : 0.86
RoboCup 2013	19	53	104 : 9	17	2	0	2.79	5.47 : 0.47

索。总之，这个案例研究的目的有两个：1. 演示了 MAXQ-OP 分层规划算法解决实际大规模问题的可扩展性和高效性；2. 给出了 RoboCup 仿真 2D 机器人足球基于决策理论的解决方案，同时也能适用于其他大规模不确定性自动规划问题。

3.6　本章小结

本章介绍了 MAXQ-OP 算法——一个新颖的结合了分层分解技术的在线规划算法。MAXQ-OP 算法递归地在线展开搜索树，并且在任务层次结构上进行策略搜索。MAXQ-OP 算法的高效性在于搜索过程中只需要根据 MAXQ 分层结构考虑相关的状态。MAXQ-OP 算法的另一个理论贡献是给出了完成函数的近似计算方法，使在线规划成为可能。主要观察是给定问题领域知识的情况下，终止分布是比较容易在线或离线近似计算的。实验结果显示 MAXQ-OP 可以在 MDP 标准测试问题——出租车问题——里面找到近似最优策略；同时在 RoboCup 仿真 2D 机器人足球领域里面，取得了非常优秀的性能。实验结果证实了 MAXQ-OP 求解大规模 MDP 问题的可行性和可扩展性。未来工作中，计划从理论上分析 MAXQ-OP 算法的性能，并将其应用到更多实际问题中去。

第四章　基于后验动作采样的蒙特卡洛在线规划算法

内容提要

蒙特卡洛树搜索（MCTS）在不确定性环境规划和学习领域已经引起广泛的研究兴趣。一个主要的挑战是利用和探索之间的平衡。为了克服这个难点，本章在马尔科夫决策过程和部分可观察马尔科夫决策过程领域提出一系列新颖的使用后验动作采样技术选择动作的蒙特卡洛在线规划算法。特别地，本章在这些问题里面应用 MCTS 方法，把 MCTS 搜索树上某一节点执行某一动作之后的累积回报看成是按照某一未知概率分布采样的随机变量。引入必要的隐藏变量来参数化这一未知分布，并在引入合适的共轭分布后，通过贝叶斯方法对未知分布的后验分布进行推理。进一步，使用 Thompson 采样技术根据某一动作是最优动作的后验概率来选择该动作，做到利用和探索之间的平衡。根据这一主要思想，本章分别针对 MDP 和 POMDP 提出基于 *Dirichlet-NormalGamma* 分布的蒙特卡洛树搜索（Dirichlet-NormalGamma based Monte-Carlo Tree Search，DNG-MCTS）和基于 *Dirichlet-Dirichlet-NormalGamma* 分布的部分可观察蒙特卡洛规划（Dirichlet-Dirichlet-NormalGamma based Partially Observable Monte-Carlo Planning，D²NG-POMCP）。实验结果显示提出的算法在多个标准测试问题里面领先领域前沿，可以计算出更优的策略。本章剩余部分按如下结构安排：第4.1节介绍基本问题；第4.2节介绍相关工作；第4.3节和第4.4节分别介绍提出的算法；第4.5节讨论先验分布的参数和算法的收敛性质。第4.6节主要介绍实验结果。第4.7节给出本章小节。

4.1　基本介绍

不确定环境下的序列化决策，包含很多规划和学习问题，在人工智能领域有着广泛的研究兴趣。当前，最通用和清晰的基本理论框架是马尔科夫决策过程（MDP）理论。MDP 为完全可观察不确定性环境下的决策提供了丰富的数学框架[23]。部分可观察马尔科夫决策过程（POMDP）把 MDP 扩展到了部分可观察环境[24]。本章主要关注于 MDP 和 POMDP 的蒙特卡洛树搜索算法。MCTS 在实现不知道环境具体模型参数的情况下，通过蒙特卡洛仿真的方式建立一棵最优优先搜索树，并进一步近似计算出当前状态下的最优动作[60]。MCTS 的主要思想是在搜索树上通过采样出来的仿真状态—动作历史评估树节点。MCTS 是无模型、高效和可高度并行化的。

MCTS 的一个基本挑战是著名的利用—探索悖论：智能体不仅需要通过执行目前看起来最好的动作以最大限度地利用已经获得的信息，还要保持探索没有访问过的状态/动作空间以希望能获得未来跟高的回报[13, 14]。UCB 可能是目前最成功和使用最广泛的处理这个问题的算法[63, 116]。UCB 算法最初是

在多臂赌博机（MAB）问题里面引入的[117]。MAB 问题的算法必须在每个周期根据过去的历史做出决策，选择执行一个动作（也就是操作一个赌博机机械臂）。UCB 算法选择可以最大化 UCB 启发函数的动作。UCB 启发函数定义了行动值函数的可能上界。Auer 等证明了 UCB 算法对 MAB 问题是渐进最优的[63]。另一方面，Thompson 采样算法可以说是处理 MAB 问题最早的启发式方法之一。Thompson 采样的主要思路是贝叶斯方式的**随机概率匹配**（Randomized Probability Matching）[42]，即根据一个动作可能成为最优动作的后验概率来随机选择这个动作。跟 UCB 相比，Thompson 采样的一大优势是其可以在很大范围内，处理带先验和后验知识的信息模型，比仅仅使用回报值的期望值要好[118]。

MAB 问题里面，通常存在两种评价标准：**累积剩余值**（Cumulative Regret）和**简单剩余值**（Simple Regret）。累积剩余值定义为因执行次优动作导致的目前为止的最优期望回报与实际期望回报之差的和；简单剩余值定义为最优期望回报与当前实验上最好的动作的期望回报之差。累积剩余值适合用来评价试图通过权衡利用—探索来最优化长期回报的算法[119]。简单剩余值对纯探索性的策略，即只有最后一个动作获得实际的回报值，更有意义[120]。就累积回报而言，Thompson 采样在实验上比 UCB 方法收敛更快[121]，最近已经证明，Thompson 采样对 MAB 问题也是渐进最优的[122–125]。在蒙特卡洛规划问题里面，规划阶段，事实上只有最后一个直接作用于环境的动作才会实际获得回报。所以在规划阶段，最优化简单剩余值比最优化累积剩余值更合理[97]。然而，简单剩余值和累积剩余值无法被同时最优化[119]；特别地，Bubeck 等证明很多情况下累积剩余值越小，简单剩余值就会越大。最近一个日益显著的共识是，在 MCTS 里面最好权衡简单剩余值和累积剩余值[126]，因为虽然规划阶段算法不需要真实收集回报，但最优化累积剩余值仍然对更好的进行最优优先搜索，建立非对称搜索树更有好处。Thompson 采样就简单剩余值而言的收敛率仍然是一个未解的问题，然而，本章观察到使用简单剩余值评价时，Thompson 采样在实验上比其他方法拥有更快的收敛速度，特别是针对大动作空间的问题。Thompson 采用似乎可以很好的同时处理累积和简单剩余值，这是在 MDP 和 POMDP 问题里面，把 Thompson 采样应用到 MCTS 的主要动机之一。

本章根据 Thompson 采样的主要思想，在 MDP 和 POMDP 问题里面，提出基于贝叶斯后验动作采样的蒙特卡洛在线规划算法。特别地，基于以前的研究成果[43, 44]，本章分别针对 MDP 和 POMDP 提出**基于** *Dirichlet-NormalGamma* **分布的蒙特卡洛树搜索**（Dirichlet-NormalGamma based Monte-Carlo Tree Search，DNG-MCTS）和**基于** *Dirichlet-Dirichlet-NormalGamma* **分布的部分可观察蒙特卡洛规划**（Dirichlet-Dirichlet-NormalGamma based Partially Observable Monte-Carlo Planning，D²NG-POMCP）算法。

DNG-MCTS 算法使用正态分布的混合分布建模 MCTS 搜索树上执行一个动作的累积回报的未知分布。在 MDP 动态规划情况下，该分布存在共轭分布为

Dirichlet 分布和 NormalGamma 分布的组合形式。选择共轭分布的前提下，在搜索树上观察到每一个回报，就比较容易更新后验分布，进而使用 Thompson 采样在每个决策节点选择需要被执行的动作。DNG-MCTS 的基本假设源自，给定策略后，MDP 退化成状态空间上的马尔科夫链。然而在 POMDP 情况下，这并不正确，因为 POMDP 问题的马尔科夫链必须被定义在环境状态空间和智能体信念空间的联合空间上。所以，在 POMDP 情况下，需要使用不同的假设。相应地，在 D²NG-POMCP 中，信念状态下执行一个动作立即回报被建模成多项分布，执行一个动作的长期回报被建模成正态分布的混合分布的凸组合。进一步，选择两个 Dirichlet 分布和一个 NormalGamma 分布的组合形式作为后验分布概率推理的共轭分布。同样地，在每一个决策节点，使用 Thompson 采样选择动作。本章在多个标准测试问题里面测试了以上算法。实验结果显示本章提出的算法在对应的问题里面比目前领先的算法效果更好。特别地，本章提出的算法可以保证收敛。

4.2　相关工作

DNG-MCTS 的基本假设是把某状态下执行某一动作后服从某一策略的长期回报建模成正态分布的混合分布。文献[127] 提出了一个类似的假设，把一个状态下获得的长期回报建模成正态分布。与之相比，正如在第4.3节显示的，根据马尔科夫链上的中心极限定理，本章关于正态分布混合分布的假设更真实。文献[128] 根据高斯近似提出了一个 MCTS 的贝叶斯 UCT 算法。特别地，贝叶斯 UCT 算法通过 MAX/MIN 分布操作把叶子节点的回报值分布传播到根节点，并使用 UCB 进行动作选择。然而，MAX/MIN 分布操作非常耗时，因为必须考虑所有的子节点。相反，我们把搜索树上的每一个决策节点看成是一个 MAB，对每一个动作分别维护其累积回报的后验分布，并使用 Thompson 采样进行动作选择。

POMDP 在线规划算法试图通过只为当前信念状态计算最优行动来缓解计算完整策略的复杂性[77]。根据展开当前信念状态的不同方式，动态规划算法可以被大致分为 3 类：分支定界剪枝（Branch-and-Bound Pruning）、启发式搜索和蒙特卡洛采样。分支定界剪枝方法通过剪枝去除已知的次优节点，来避免搜索树上的不必要计算[129–132]。启发式搜索算法根据启发函数关注于最相关的可达信念空间以搜索最具有潜力的信念节点[75, 76, 79, 80, 133, 134]。蒙特卡洛技术通过采样一个或多个观察来减少搜索的分支数，从而允许在给定的规划时间内搜索到更深的节点[81, 83, 135–137]。

4.3　基于后验动作采样的 MDP 蒙特卡洛规划

本节提出基于贝叶斯后验动作采样的 MDP 蒙特卡洛在线规划算法，即 DNG-MCTS。本节首先介绍关于 MDP 在线规划的基本假设，之后提出贝叶斯建模后推理方法。

4.3.1　基本假设

本章的基本假设来源于马尔科夫链上的中心极限定理[138, 139]。

定理 4.3.1 (马尔科夫链中心极限定理). 令 $X = \{x_0, x_1, \ldots\}$ 为一个定义在可数状态空间 $\mathcal{X}$ 上的遍历马尔科夫链。假设，该马尔科夫链的稳态分布为 w。令 f 为任何定义在 $\mathcal{X}$ 上的有界函数，并定义 $\mu = E_w[f] = \int_{\mathcal{X}} w(x)f(x)\,dx$，及 $\sigma = \mathrm{Var}_w(f(x_0)) + 2\sum_{i=1}^{\infty} \mathrm{Cov}_w(f(x_0), f(x_i))$。令 $\mathcal{N}(0, \sigma^2)$ 为一正态分布，则对任何初始分布 x_0，当 $n \to \infty$ 时，有：

$$\sqrt{n}\left(\frac{1}{n}\sum_{t=0}^{n} f(x_t) - \mu\right) \to \mathcal{N}(0, \sigma^2). \tag{4.1}$$

推论 4.3.1. 定理 *4.3.1* 表明，当 n 趋向于无穷大时，样本平均值 $\frac{1}{n}\sum_{t=0}^{n} f(s_t)$ 服从正态分布 $\mathcal{N}(\mu, \sigma^2/n)$。所以，如果 n 充分大，可以合理假设 $\frac{1}{n}\sum_{t=0}^{n} f(s_t)$ 服从正态分布。在次此假设下，$\sum_{t=0}^{n} f(s_t)$ 也是正态分布，因为 n 是一个常数。

对于一个给定的 MDP 策略 π，令随机变量 $X_{s,\pi}$ 表示状态 s 下服从策略 π 获得的累积回报，并令 $X_{s,a,\pi}$ 表示状态 s 下首先执行动作 a，再服从策略 π 获得的累积回报。基本假设为：

假设 4.3.1. $X_{s,\pi}$ 服从正态分布；

假设 4.3.2. $X_{s,a,\pi}$ 服从正态分布的混合分布。

以上假设基于以下原因是合理性的假设。给定策略 π，一个 MDP 退化成一个定义在有限状态空间 S 上的马尔科夫链 $\{s_t\}$。该马尔科夫链的转移函数为：$\Pr(s' \mid s) = T(s' \mid s, \pi(s))$。假设得到的马尔科夫链 $\{s_t\}$ 是遍历的，即从任何一个状态都有可能转移到任何其他状态。对有限规划时限 MDP 问题，假设规划实现为 H，如果 $\gamma = 1$，则 $X_{s_0,\pi} = \sum_{t=0}^{H} R(s_t, \pi(s_t))$ 为 $f(s_t) = R(s_t, \pi(s_t))$ 的和。根据推论4.3.1有，如果 H 足够大，$X_{s_0,\pi}$ 对所有状态 $s_0 \in S$ 近似服从正态分布。在 $\gamma \neq 1$ 的情况下，如果 H 足够大，并且 γ 接近于 1，任何可以合理假设 $X_{s_0,\pi}$ 服从正态分布。

如果策略 π 是不固定的，并且随着时间改变（比如在线规划算法在算法收敛前的策略），$X_{s,\pi}$ 的真实分布是未知的，并且可能非常复杂。然而，如果算法最终可以保证收敛（正如在第 4.5.2节讨论的，这就是 DNG-MCTS 算法的情况），那么假设 $X_{s,\pi}$ 服从正态分布仍然是合理和方便的。

现在考虑在状态 s 下执行动作 a 后，再服从策略 π 的累积回报。根据定义：

$$X_{s,a,\pi} = R(s,a) + \gamma X_{s',\pi}, \tag{4.2}$$

其中，s' 是下一个可能的状态，按照 $T(s' \mid s,a)$ 分布。定义随机变量 $Y_{s,a,\pi}$ 如下：

$$Y_{s,a,\pi} = \frac{1}{\gamma}\left(X_{s,a,\pi} - R(s,a)\right). \tag{4.3}$$

事实上，$Y_{s,a,\pi}$ 的概率密度函数是对所有状态 s' 对应的 $X_{s',\pi}$ 的概率密度函数的凸组合。形式上，有：

$$f_{Y_{s,a,\pi}}(x) = \sum_{s' \in S} T(s' \mid s,a) f_{X_{s',\pi}}(x). \tag{4.4}$$

所以，如果对所有状态 s'，$X_{s',\pi}$ 都被假设服从正态分布，那么就可以合理地将 $Y_{s,a,\pi}$ 的分布建模成正态分布的混合分布。又因为 $X_{s,a,\pi}$ 是 $Y_{s,a,\pi}$ 的线性函数，所以 $X_{s,a,\pi}$ 的分布在该假设下也是正态分布的混合分布。

4.3.2　贝叶斯建模和推理

定理 4.3.2 (贝叶斯推理)**.** *假设分布未知的随机变量 X 被建模成依赖于参数 θ 的似然函数 $L(x \mid \theta)$。令 θ 的先验分布为 $\Pr(\theta)$，则在观察到一组独立同分布的 X 样本 $Z = \{x_1, x_2, \ldots\}$ 后，θ 的后验分布可以由贝叶斯定理得到：*

$$\Pr(\theta \mid Z) = \eta \Pr(Z \mid \theta) \Pr(\theta) = \eta \prod_i L(x_i \mid \theta) \Pr(\theta), \tag{4.5}$$

其中 $\eta = 1/\Pr(Z)$ 是一个正则化因子。

定理 4.3.2 和 假设 4.3.1 表明，在贝叶斯设定下，用正态分布 $\mathcal{N}(\mu_s, 1/\tau_s)$ 建模 $X_{s,\pi}$ 的分布是合理的，其中 μ_s 是未知的期望值，τ_s 是未知的精度。正态分布的精度定义为其方差的倒数，即 $\tau = 1/\sigma^2$。引入精度是为了下文方便引入 NormalGamma 分布作为共轭分布[140]。

定义 4.3.1 (NormalGamma 分布)**.** *一个 NormalGamma 分布由一组超参数 (Hyper Parameters) 完全决定：$\langle \mu_0, \lambda, \alpha, \beta \rangle$，其中 $\lambda > 0$，$\alpha \geqslant 1$，并且 $\beta \geqslant 0$。令 $\Gamma(\cdot)$ 为 Gamma 函数，则 (μ, τ) 服从 NormalGamma 分布 $\mathrm{NormalGamma}(\mu_0, \lambda, \alpha, \beta)$，如果 (μ, τ) 的概率密度函数有如下形式：*

$$f(\mu, \tau \mid \mu_0, \lambda, \alpha, \beta) = \frac{\beta^\alpha \sqrt{\lambda}}{\Gamma(\alpha)\sqrt{2\pi}} \tau^{\alpha - \frac{1}{2}} e^{-\beta\tau} e^{-\frac{\lambda\tau(\mu-\mu_0)^2}{2}}. \tag{4.6}$$

根据定义，τ 的边缘分布是 Gamma 分布，表示为 $\tau \sim \mathrm{Gamma}(\alpha, \beta)$，并且给定 τ 后 μ 的条件分布是一个正态分布，表示为 $\mu \sim \mathcal{N}(\mu_0, 1/(\lambda\tau))$。

定理 4.3.3 (NormalGamma 参数的后验分布). 令 X 服从未知均值 μ 和精度 τ 的正态分布：$X \sim \mathcal{N}(\mu, 1/\tau)$，并且 (μ, τ) 的先验分布服从 *NormalGamma* 分布：$(\mu, \tau) \sim \mathrm{NormalGamma}(\mu_0, \lambda_0, \alpha_0, \beta_0)$。在观察到 n 个独立同分布的 X 样本 $\{x_1, x_2, \ldots, x_n\}$ 后，令 $\bar{x} = \sum_{i=1}^{n} x_i/n$ 和 $s = \sum_{i=1}^{n} (x_i - \bar{x})^2 / n$ 分别为均值和方差，则根据贝叶斯定理，(μ, τ) 的后验分布也是 *NormalGamma* 分布，即 $(\mu, \tau) \sim \mathrm{NormalGamma}(\mu_n, \lambda_n, \alpha_n, \beta_n)$，其中：

$$\mu_n = \frac{\lambda_0 \mu_0 + n\bar{x}}{\lambda_0 + n}, \tag{4.7}$$

$$\lambda_n = \lambda_0 + n, \tag{4.8}$$

$$\alpha_n = \alpha_0 + \frac{n}{2}, \tag{4.9}$$

$$\beta_n = \beta_0 + \frac{1}{2}\left(ns + \frac{\lambda_0 n(\bar{x} - \mu_0)^2}{\lambda_0 + n}\right). \tag{4.10}$$

根据 假设 4.3.2，$Y_{s,a,\pi}$ 的分布可以被建模成正态分布的混合分布：

$$Y_{s,a,\pi} \sim \sum_{s' \in S} w_{s,a,s'} \mathcal{N}(\mu_{s'}, \frac{1}{\tau_{s'}}), \tag{4.11}$$

其中 $w_{s,a,s'} = T(s' \mid s, a)$ 是混合权重，满足 $w_{s,a,s'} \geqslant 0$ 和 $\sum_{s' \in S} w_{s,a,s'} = 1$。注意到 $w_{s,a,s'}$ 在蒙特卡洛规划问题里面是事先不知道的。一个自然的表达这些未知权重的方法是引入 Dirichlet 分布，因为 Dirichlet 分布是通常离散概率分布的共轭分布[140]。对状态 s 和动作 a，Dirichlet 分布表示为 $\mathrm{Dirichlet}(\rho_{s,a})$，其中 $\rho_{s,a} = (\rho_{s,a,s_1}, \rho_{s,a,s_2}, \ldots)$ 是超参数向量。该 Dirichlet 分布给出了，状态转移 $(s, a) \to s'$ 被观察到 $\rho_{s,a,s'} - 1$ 次数以后，$T(s' \mid s, a)$ 的后验分布。事实上，在观察到一次状态转移 $(s, a) \to s'$ 后，$T(s' \mid s, a)$ 的后验分布仍然是 Dirichlet 分布，更新规则如下：

$$\rho_{s,a,s'} \leftarrow \rho_{s,a,s'} + 1. \tag{4.12}$$

所以，为了建模 $X_{s,\pi}$ 和 $X_{s,a,\pi}$ 的分布，仅需要对 MCTS 搜索树上遇到的所有状态 s 和动作 a 维护一组超参数：$\langle \mu_{s,0}, \lambda_s, \alpha_s, \beta_s \rangle$ 和 $\rho_{s,a}$，并且根据贝叶斯规则进行更新。

4.3.3　基于 Thompson 采样的动作选择策略

定义 4.3.2 (Thompson 采样). *Thompson* 采样根据一个动作成为最优动作的后验概率来随机选择动作。形式上，在贝叶斯设定下，动作 a 被选中的概率为：

$$\Pr(a) = \int \mathbf{1}\left[a = \operatorname*{argmax}_{a'} \mathrm{E}\left[X_{a'} \mid \theta_{a'}\right]\right] \prod_{a'} P_{a'}(\theta_{a'} \mid Z)\, d\theta, \tag{4.13}$$

其中 θ_a 给出动作 a 回报值分布的隐含参数，$\theta = (\theta_{a_1}, \theta_{a_2}, \ldots)$ 是给出所有动作回报值分布的参数向量，$\mathrm{E}[X_a \mid \theta_a] = \int x L_a(x \mid \theta_a)\, dx$ 是，给定参数 θ_a 后，动作 a 的期望回报。

```
OnlinePlanning(s : state, T : tree)
Initialize H ← maximal planning horizon
repeat
    DNG-MCTS(s, T, 0)
until resource budgets reached
return ThompsonSampling(s, 0, False)

DNG-MCTS(s : state, T : tree, d : depth)
if d ⩾ H or s is terminal then
    return 0
end
else if node ⟨s, d⟩ is not in tree T then
    Initialize (μ_{s,0}, λ_s, α_s, β_s), and ρ_{s,a} for a ∈ A
    Add node ⟨s, d⟩ to T
    Play rollout policy by simulation for H − d steps
    Get the cumulative reward r
    return r
end
else
    a ← ThompsonSampling(s, d, True)
    Execute a by simulation
    Observe next state s′ and reward R(s, a)
    r ← R(s, a) + γDNG-MCTS(s′, T, d + 1)
    α_s ← α_s + 0.5
    β_s ← β_s + (λ_s(r − μ_{s,0})^2/(λ_s + 1))/2
    μ_{s,0} ← (λ_s μ_{s,0} + r)/(λ_s + 1)
    λ_s ← λ_s + 1
    ρ_{s,a,s′} ← ρ_{s,a,s′} + 1
    return r
end
```

算法 4.1: 基于 Dirichlet-NormalGamma 的蒙特卡洛树搜索算法

Thompson 采样可以高效地使用采样方法实现。为此，首先对每一个动作 $a \in A$，根据后验分布 $P_a(\theta_a \mid Z)$ 采样一组参数 θ_a，再选择具有期望回报最大的动作，即：

$$a^* = \operatorname*{argmax}_a \mathrm{E}[X_a \mid \theta_a]. \tag{4.14}$$

在 DNG-MCTS 中，每个决策节点上的动作选择策略是通过 Thompson 采样实现的。对每个决策节点 s，分别根据后验分布 $\mathrm{NormalGamma}(\mu_{s',0}, \lambda_{s'}, \alpha_{s'}, \beta_{s'})$ 和 $\mathrm{Dirichlet}(\rho_{s,a})$，为每一个可能的下一个状态 s'，采样出均值 $\mu_{s'}$ 和混合权重 $w_{s,a,s'}$。采样出的行动值函数 $\tilde{Q}(s,a)$ 计算如下；

$$\tilde{Q}(s,a) = R(s,a) + \gamma \sum_{s' \in S} w_{s,a,s'} \mu_{s'}. \tag{4.15}$$

具有最大行动值函数值的动作即被选为通过仿真执行的动作。

```
1  ThompsonSampling(s : state, d : depth, sampling : boolean)
2  foreach a ∈ A do
3  |  q_a ← QValue(s, a, d, sampling)
4  end
5  return argmax_a q_a

6  QValue(s : state, a : action, d : depth, sampling : boolean)
7  r ← 0
8  foreach s′ ∈ S do
9  |  if sampling = True then
10 |  |  Sample w_s′ ~ Dirichlet(ρ_s,a)
11 |  end
12 |  else
13 |  |  w_s′ ← ρ_s,a,s′ / Σ_s″∈S ρ_s,a,s″
14 |  end
15 |  r ← r + w_s′ Value(s′, d + 1, sampling)
16 end
17 r ← R(s, a) + γr
18 return r

19 Value(s : state, d : depth, sampling : boolean)
20 if d ⩾ H or s is terminal then
21 |  return 0
22 end
23 else
24 |  if sampling = True then
25 |  |  Sample (μ, τ) ~ NormalGamma(μ_s,0, λ_s, α_s, β_s)
26 |  |  return μ
27 |  end
28 |  else
29 |  |  return μ_s,0
30 |  end
31 end
```

算法 4.2: DNG-MCTS 算法中的 Thompson 采样算法

4.3.4　DNG-MCTS 算法

DNG-MCTS 算法的主要流程见算法 4.1和算法 4.2。值得注意的是函数 ThompsonSampling 由一个布尔类型参数 sampling。如果 sampling 为真，那么就使用 Thompson 采样方法来选择动作；否则，根据当前的平均行动值函数 $\bar{Q}(s,a)$ 来选择动作，其中：

$$\bar{Q}(s,a) = R(s,a) + \gamma \sum_{s' \in S} \frac{\rho_{s,a,s'}}{\sum_{s'' \in S} \rho_{s,a,s''}} \mu_{s',0}. \tag{4.16}$$

在每一个迭代周期内，DNG-MCTS 函数采用 Thompson 采样在当前搜索树 T 上，从根节点一直到叶子节点，递归地选择需要被仿真执行的动作。搜索过程

中，把新访问的节点插入到树上，从该节点出发运行默认 Rollout 策略，并使用仿真结果来更新已经访问过节点的状态和动作值函数分布。OnlinePlanning 函数是跟环境交互的主题函数。该函数被调用时，需要提供当前状态 s，和一棵初始为空的搜索树 T。OnlinePlanning 函数递归地调用 DNG-MCTS 函数直到达到计算资源的预算，比如搜索超时，或者已经达到最大迭代次数。最后，根据当前树上的值函数返回一个贪心的最优动作，供智能体实际在环境中执行。注意到，Rollout 策略对每一个新加入的节点，只运行一次，也就是说，贝叶斯设定里面，过去的观察 Z 大小为 $n = 1$。

4.4　基于后验动作采样的 POMDP 蒙特卡洛规划

本节介绍基于后验动作采样的 POMDP 蒙特卡洛在线规划算法 D²NG-POMCP。本节首先提出 POMDP 问题中的基本假设，之后给出贝叶斯建模和推理方法。

4.4.1　基本假设

假设一个服从策略 π 的 POMDP 智能体在跟环境交互，本节把 $\langle s, b\rangle$ 看成是一个联合状态，其中 s 是环境的真实状态（包括智能体自身的状态），b 是智能体内部维护的环境的信念状态。令 $\mathcal{J} = S \times \mathcal{B}$ 表示状态空间 S 和信念状态空间 $\mathcal{B}$ 的联合状态空间，则联合状态的随机过程退化成一个定义在联合状态空间 $\mathcal{J}$ 上的马尔科夫链 $\{\langle s_t, b_t\rangle\}$，其转移函数为：

$$P\left(\langle s', b'\rangle \mid \langle s, b\rangle\right) = T\left(s' \mid s, \pi(b)\right) T^{+}\left(b' \mid b, \pi(b)\right). \tag{4.17}$$

令随机变量 $X_{b,a}$ 表示信念状态 b 上执行动作 a 后智能体能获得的立即回报，随机变量 $X_{s,b,\pi}$ 表示从联合状态 $\langle s, b\rangle$ 开始服从策略 π 所能获得的累积回报，以及随机变量 $X_{b,\pi}$ 为从信念状态 b 开始服从策略 π 所能获得的累积回报。本节的基本假设为：

假设 4.4.1. $X_{b,a}$ 服从多项分布；

假设 4.4.2. $X_{s,b,\pi}$ 服从正态分布；

假设 4.4.3. $X_{b,\pi}$ 服从正态分布的混合分布。

本节假设 POMDP 的立即回报值集合 $\mathcal{I}$ 是有限的，即 $\mathcal{I} = \{r_1, r_2, \ldots, r_k\}$，其中 $r_i = R(s, a)$ 表示某个状态 s 下执行动作 a 的立即回报值。容易看出，$X_{b,a}$ 服从多项分布（Multinomial Distribution），表示为 $\mathrm{Multinomial}(p_1, p_2, \ldots, p_k)$，使得 $\sum_{i=1}^{k} p_i = 1$，同时 $p_i = \sum_{s \in S} \mathbf{1}[R(s, a) = r_i] b(s)$ 是 $X_{b,a} = r_i$ 的概率[141].

在 POMDP 问题里面，从初始信念状态 b_0 出发的可达信念状态空间是可数的，因为每一个动作 -观察的历史序列组合唯一确定一个信念状态，而历史空

间是按照字典序自然可数的。所以，联合状态 $\langle s_0, b_0\rangle$ 的可达联合状态空间也是可数的，因为根据定义状态空间是可数的。对规划时限为 H 的有限规划时限 POMDP，如果 $\gamma = 1$，那么 $X_{s_0,b_0,\pi} = \sum_{t=0}^{H} R(s_t, \pi(b_t))$ 可以被看成是函数 $f(s_t, b_t) = R(s_t, \pi(b_t))$ 的和。假设得到的马尔科夫链是遍历的，根据推论4.3.1，如果 H 足够大，那么对每个联合状态 $\langle s_0, b_0\rangle \in \mathcal{J}$，$X_{s_0,b_0,\pi}$ 是近似正态分布的。另一方面，如果 $\gamma \neq 1$，但接近于 1，如果 H 足够大，仍然可以合理假设 $X_{s_0,b_0,\pi}$ 是正态分布的。

从信念状态 b 开始服从策略 π 能获得的累积回报由最终的马尔科夫链完全决定，即 $X_{b,\pi} = X_{s,b,\pi}$，其中 s 根据 b(s) 分布。所以，$X_{b,\pi}$ 的概率密度函数可以被表达成 $X_{s,b,\pi}$ 概率密度函数的凸组合：

$$f_{X_{b,\pi}}(x) = \sum_{s\in S} b(s) f_{X_{s,b,\pi}}(x). \tag{4.18}$$

进一步，如果对于所有的 $\langle s, b\rangle \in \mathcal{J}$，$X_{s,b,\pi}$ 被假设成服从正态分布，可以直接把 $X_{b,\pi}$ 的分布建模成正态分布的混合分布。

在策略 π 不固定，并且随着时间改变的情况下，比如在线规划算法在算法收敛前的策略，$X_{b,\pi}$ 的真实分布是未知的，并且可能很复杂。然而，如果算法可以保证收敛，那么把 $X_{b,\pi}$ 近似成正态分布仍然是方便和合理的。

4.4.2　贝叶斯建模和推理

假设 4.4.1显示可以把 $X_{b,a}$ 的分布建模成多项分布，形式上表示为：$X_{b,a} \sim \mathrm{Multinomial}(p_1, p_2, \ldots, p_k)$。由于多项分布的共轭分布为 Dirichlet 分布，对信念状态 b 和动作 a，p_i 的先验分布可以建模为 Dirichlet 分布，即 $\mathrm{Dirichlet}(\psi_{b,a})$，其中 $\psi_{b,a} = (\psi_{b,a,r_1}, \psi_{b,a,r_2}, \ldots, \psi_{b,a,r_k})$ 是一组超参数。在观察到立即回报 r 后，后验分布也是 Dirichlet 分布，更新规则如下：

$$\psi_{b,a,r} \leftarrow \psi_{b,a,r} + 1. \tag{4.19}$$

根据假设 4.4.2，$X_{s,b,\pi}$ 的分布可以建模成正态分布 $\mathcal{N}(\mu_{s,b}, 1/\tau_{s,b})$，其中 $\mu_{s,b}$ 和 $\tau_{s,b}$ 分别为未知的均值和精度。选择 NormalGamma 分布作为共轭分布，$(\mu_{s,b}, \tau_{s,b})$ 的后验分布也是 NormalGamma 分布，也就是表示为 $(\mu_{s,b}, \tau_{s,b}) \sim \mathrm{NormalGamma}(\mu_{s,b,0}, \lambda_{s,b}, \alpha_{s,b}, \beta_{s,b})$，其中 $\mu_{s,b,0}$，$\lambda_{s,b}$，$\alpha_{s,b}$ 以及 $\beta_{s,b}$ 是超参数。

正如假设 4.4.3中所解释的，$X_{b,\pi}$ 服从正态分布的混合分布，可以被建模成 b(s) and $X_{s,b,\pi}$ 关于所有状态 s 的凸组合。现在考虑在信念状态 b 下首先执行动作 a，再服从策略 π 能获得的累积回报，记为 $X_{b,a,\pi}$。根据定义：

$$X_{b,a,\pi} = X_{b,a} + \gamma X_{b',\pi}, \tag{4.20}$$

其中 b' 是按照 $T^+(b' \mid b, a)$ 分布的下一个信念状态。显示表达 $X_{b,a,\pi}$ 的分布是困难的。然而，其期望值可以简单计算如下：

$$\begin{aligned} E[X_{b,a,\pi}] &= E[X_{b,a}] + \gamma \sum_{b' \in \mathcal{B}} E[X_{b',\pi}] T^+(b' \mid b, a) \\ &= E[X_{b,a}] + \gamma \sum_{o \in O} \mathbf{1}[b' = \zeta(b, a, o)] \Omega(o \mid b, a) E[X_{b',\pi}]. \end{aligned} \tag{4.21}$$

事实上，如果底层的转移函数和观察函数是已知的，那么 $E[X_{b,a,\pi}]$ 就是通常的行动值函数，满足：

$$Q^\pi(b, a) = r(b, a) + \gamma \sum_{o \in O} \Omega(o \mid b, a) V^\pi(\zeta(b, a, o)). \tag{4.22}$$

考虑到本节的基本假设中，$X_{b,a}$ 服从多项分布，$X_{b',\pi}$ 服从正态分布的混合分布。现在的问题就是，如何建模实现不知道的观察函数——$\Omega(\cdot \mid b, a)$。幸运的是，$\Omega(\cdot \mid b, a)$ 在贝叶斯设定下，可以通过引入 Dirichlet 分布作为共轭分布来建模。形式上，共轭分布为 $\mathrm{Dirichlet}(\rho_{b,a})$，其中 $\rho_{b,a} = (\rho_{b,a,o_1}, \rho_{b,a,o_2}, \dots)$ 是超参数。在观察到一个转移 $(b, a) \to o$ 后，$\Omega(\cdot \mid b, a)$ 的后验分布按照如下规则进行更新：

$$\rho_{b,a,o} \leftarrow \rho_{b,a,o} + 1. \tag{4.23}$$

所以，为了计算 $X_{b,a,\pi}$ 的后验分布的期望值，仅需为 MCTS 搜索树上每一个遇到的信念状态 b，状态 s 和动作 a 维护一组超参数：$\langle \mu_{s,b,0}, \lambda_{s,b}, \alpha_{s,b}, \beta_{s,b} \rangle$，$\psi_{b,a}$ 以及 $\rho_{b,a}$，并按照贝叶斯定理进行更新。

4.4.3　基于 Thompson 采样的动作选择策略

在 D²NG-POMCP 算法中，决策节点的动作选择策略由 Thompson 采样给出。更精确地说，在信念状态 s 对应的决策节点，为了计算 $X_{b,a,\pi}$ 的期望值，首先根据 $\mathrm{Dirichlet}(\rho_{b,a})$ 为每个观察 $o \in O$ 采样 $w_{b,a,o}$，根据 $\mathrm{Dirichlet}(\psi_{b,a})$ 为每个回报值 $r \in \mathcal{I}$ 采样 $w_{b,a,r}$，并根据 $\mathrm{NormalGamma}(\mu_{s',b',0}, \lambda_{s',b'}, \alpha_{s',b'}, \beta_{s',b'})$ 为每个联合状态 $\langle s', b' \rangle \in \mathcal{J}$ 采样 $\mu_{s',b'}$，其中 $b' = \zeta(b, a, o)$ 为在信念状态 b 下执行动作 a 并获得观察 o 后的下一个信念状态。最终，具有最大行动值函数值 $\tilde{Q}(b, a)$ 的动作即被选为通过仿真执行的动作，即：

$$\tilde{Q}(b, a) = \sum_{r \in \mathcal{I}} w_{b,a,r} r + \gamma \sum_{o \in O} \mathbf{1}[b' = \zeta(b, a, o)] w_{b,a,o} \sum_{s' \in S} \mu_{s',b'} b'(s'). \tag{4.24}$$

4.4.4　D²NG-POMCP 算法

D²NG-POMCP 的主要流程见算法 4.3和算法 4.4。正如前文所指出的，算法是基于历史节点设计的，而不是直接使用信念状态。给定初始信念状态，历史 h

```
OnlinePlanning(h : history, T : tree)
repeat
    Sample s ~ P(h)
    D²NG-POMCP(s, h, T, 0)
until resource budgets reached
return ThompsonSampling(h, 0, False)

Agent(b_0 : initial belief)
Initialize H ← maximal planning horizon
Initialize I ← {possible immediate rewards}
Initialize h ← ∅
Initialize P(h) ← b_0
repeat
    a ← OnlinePlanning(h, ∅)
    Execute a and get observation o
    h ← hao
    P(h) ← ParticleFilter(P(h), a, o)
until terminating conditions

D²NG-POMCP(s : state, h : history, T : tree, d : depth)
if d ⩾ H or s is terminal then
    return 0
end
else if node ⟨h⟩ is not in tree T then
    Initialize (μ_{s,h,0}, λ_{s,h}, α_{s,h}, β_{s,h}) for s ∈ S, and ρ_{h,a} and ψ_{h,a} for a ∈ A
    Add node ⟨h⟩ to T
    Play rollout policy for H − d steps
    Get cumulative reward r
    return r
end
else
    a ← ThompsonSampling(h, d, True)
    Execute a by simulation
    Get state s′, observation o and reward i
    h′ ← hao
    P(h′) ← P(h′) ∪ s′
    r ← i + γ D²NG-POMCP(s′, h′, T, d + 1)
    α_{s,h} ← α_{s,h} + 0.5
    β_{s,h} ← β_{s,h} + (λ_{s,h}(r − μ_{s,h,0})²/(λ_{s,h} + 1))/2
    μ_{s,h,0} ← (λ_{s,h}μ_{s,h,0} + r)/(λ_{s,h} + 1)
    λ_{s,h} ← λ_{s,h} + 1
    ρ_{h,a,o} ← ρ_{h,a,o} + 1
    ψ_{h,a,i} ← ψ_{h,a,i} + 1
    return r
end
```

算法 4.3: 基于 Dirichlet-Dirichlet-NormalGamma 的 POMCP 算法

```
ThompsonSampling(h : history, d : depth, sampling : boolean)
foreach a ∈ A do
    q_a ← QValue(h, a, d, sampling)
end
return argmax_a q_a

QValue(h : history, a : action, d : depth, sampling : boolean)
r ← 0
foreach o ∈ O do
    if sampling = True then
        Sample w_o ~ Dirichlet(ρ_{h,a})
    end
    else
        w_o ← ρ_{h,a,o} / Σ_{o'∈O} ρ_{h,a,o'}
    end
    h' ← hao
    r ← r + w_o Value(h', d + 1, sampling)
end
r ← γr
foreach i ∈ I do
    if sampling = True then
        Sample w_i ~ Dirichlet(ψ_{h,a})
    end
    else
        w_i ← ψ_{h,a,i} / Σ_{i'∈I} ψ_{h,a,i'}
    end
    r ← r + w_i i
end
return r

Value(h : history, d : depth, sampling : boolean)
if d ⩾ H then
    return 0
end
else
    if sampling = True then
        Sample (μ_s, τ_s) ~ NormalGamma(μ_{s,h,0}, λ_{s,h}, α_{s,h}, β_{s,h})
        for s ∈ P(h)
        return 1/|P(h)| Σ_{s∈P(h)} μ_s
    end
    else
        return 1/|P(h)| Σ_{s∈P(h)} μ_{s,h,0}
    end
end
```

算法 4.4: D²NG-POMCP 算法中的 Thompson 采样算法

唯一决定了当前信念状态，算法中为每一个 MCTS 搜索树上遇到的历史节点 h，状态 s 和动作 a 维护超参数 $\langle \mu_{s,h,0}, \lambda_{s,h}, \alpha_{s,h}, \beta_{s,h} \rangle$，$\psi_{h,a}$ 和 $\rho_{h,a}$。特别地，每个历史节点对应的信念状态通过粒子进行近似，即 $\mathcal{P}(h)$。进一步，使用粒子滤波方法用来更新这些粒子[142, 143]。公式 4.24就变为：

$$\tilde{Q}(h,a) = \sum_{r \in \mathcal{I}} w_{h,a,r} r + \gamma \sum_{o \in O} w_{h,a,o} \sum_{s' \in \mathcal{P}(hao)} \mu_{s',hao}, \tag{4.25}$$

其中参数 $w_{h,a,r}$，$w_{h,a,o}$ 和 $\mu_{s',hao}$ 分别根据 $\mathrm{Dirichlet}(\psi_{h,a})$，$\mathrm{Dirichlet}(\rho_{h,a})$ 和 $\mathrm{NormalGamma}(\mu_{s',hao,0}, \lambda_{s',hao}, \alpha_{s',hao}, \beta_{s',hao})$ 采样获得。搜索过程结束时，一个当前搜索树上的拥有最大行动值函数 $\bar{Q}(h,a)$ 的贪心动作被返回，其中：

$$\bar{Q}(h,a) = \sum_{r \in \mathcal{I}} \frac{\psi_{h,a,r}}{\sum_{r' \in \mathcal{I}} \psi_{h,a,r'}} r + \gamma \sum_{o \in O} \frac{\rho_{h,a,o}}{\sum_{o' \in O} \rho_{h,a,o'}} \sum_{s' \in \mathcal{P}(hao)} \mu_{s,hao,0}. \tag{4.26}$$

在每个迭代周期，D²NG-POMCP 函数在已经存在的搜索树 T 上从根节点开始一直到叶子节点递归地选择动作，并仿真执行。算法将新访问的节点加入到搜索树上，并从该节点开始仿真运行默认 Rollout 策略，最后把仿真结果反馈到根节点，并更新搜索上的统计量。OnlinePlanning 函数被调用时，需提供当前的历史节点 h 和初始为空的搜索树 T。算法重复地从当前信念状态 $\mathcal{P}(h)$ 中采样出一个状态，并通过调用 D²NG-POMCP 函数进行前向搜索，直到达到计算资源预算。最后，根据当前树上的值函数返回一个贪心的最优动作给智能体。Agent 函数是智能体更环境交互的整体过程。该函数重复调用 OnlinePlanning 选择最好的动作，执行这个动作，获得观察，调用 ParticleFilter 函数更新粒子，进入下一决策周期，直到终止条件（比如问题获得解决，或达到最大运行时间）。

4.5　讨论

本节主要讨论如何通过初始化超参数来选择先验分布，和算法的收敛性质。

4.5.1　先验分布

虽然先验分布的影响在算法后期阶段可以忽略，但先验分布的选择在算法的早期阶段非常重要，特别是只观察到很少量数据的时候。通常来讲，先验分布应该反映隐藏模型的先验知识。

在没有任何先验知识的时候，比较偏好*无信息先验分布*（Uninformative Priors）[144]。根据*无差别原则*（Principle of Indifference），无信息先验分布对所有的可能性赋值相同的概率。以 NormalGamma 分布为例，本章希望给定 τ 后采样出来的 μ 分布：$\mathcal{N}(\mu_0, 1/(\lambda\tau))$，尽量比较平。这就需要一个无限大的方差 $1/(\lambda\tau) \to \infty$，所以 $\lambda\tau \to 0$。另一方面，τ 服从期望为 $\mathrm{E}[\tau] = \alpha/\beta$ 的 Gamma 分布 $\mathrm{Gamma}(\alpha, \beta)$，所以从期望来看，$\lambda\alpha/\beta \to 0$。考虑参数的定义范

围（$\lambda > 0, \alpha \geqslant 1, \beta \geqslant 0$），可以把 λ 设定的尽量小，$\alpha = 1$，并且 β 足够大。另一方面，希望采样出来的分布在数轴的中间位置，这就要求 $\mu_0 = 0$。值的指出的是，直觉上 β 不应该被设定的太大，否则收敛过程会比较慢。对于 Dirichlet 分布，把其参数设置为很小的正数就可以得到无信息的先验分布。

另一方面，如果确实有先验知识，*信息先验分布*（Informative Priors）更合理。以 DNG-MCTS 为例，通过利用先验知识，一个状态节点可以被初始化成反映其跟其他节点相比优先级的信息先验分布。在 DNG-MCTS 中，这是通过根据对状态节点主观估值进行初始化先验分布参数的。对于 NormalGamma 分布，就伪观察而言，其参数的解释是：如果事先通过 λ 样本得到的均值估计为 μ_0，通过 2α 样本得到的精度估计为 α/β，那么 μ 和 τ 的先验分布为 $\mathrm{NormalGamma}(\mu_0, \lambda, \alpha, \beta)$[140]。这个先验分布的解释提供了根据历史信息初始化先验分布的直接方法。如何为特定的问题初始化特定的先验部分超出本章的讨论范围。可以初始化不同先验分布为算法提供了重要的灵活性，应被认为是算法的优越性之一。

4.5.2　收敛性质

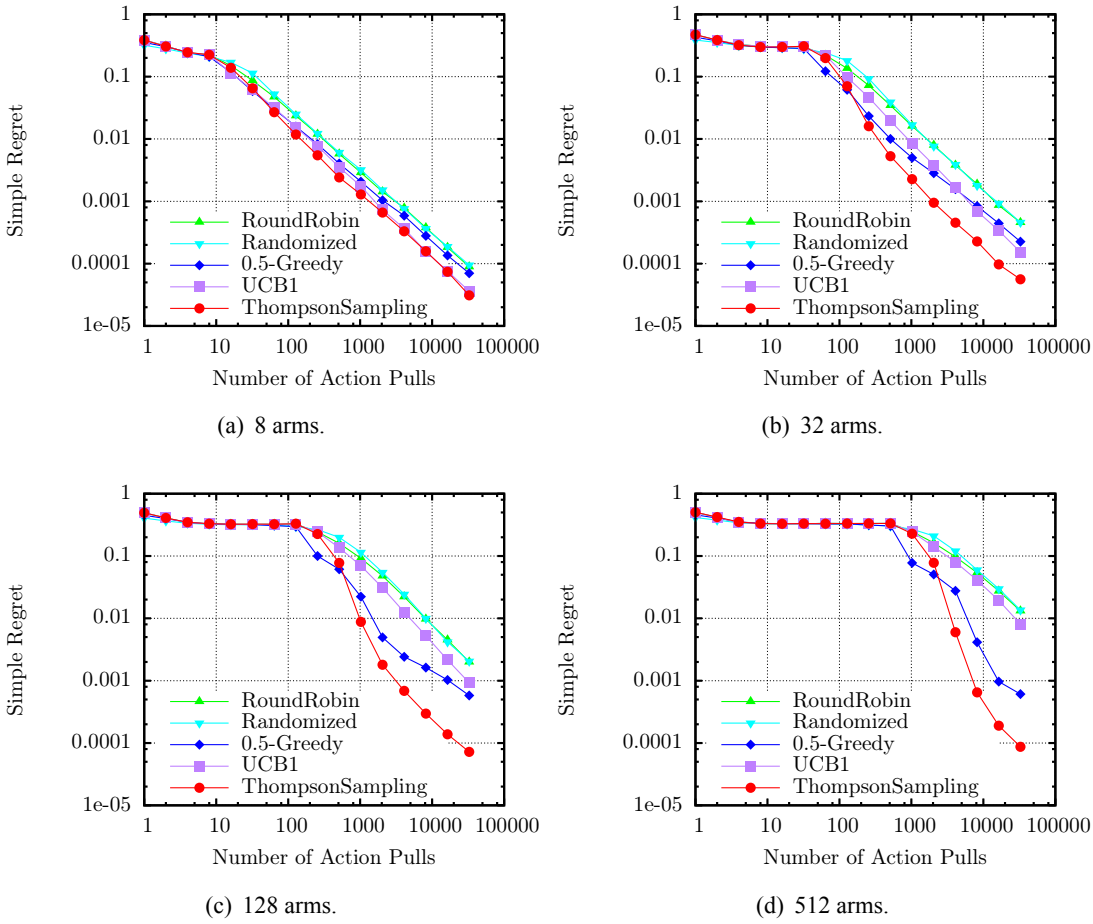
对稳定 MAB 问题（也就是底层回报分布不随时间而变化）中的 Thompson 采样，文献[124] 证明：1. 选择次优动作的概率是选择最优动作的概率的线性函数；2. 该线性函数的系数随选择最优动作次数的增加呈指数减小。所以，Thompson 采样在 MAB 问题中最终选择最优动作的概率趋于 1。

以 DNG-MCTS 为例，$X_{s,\pi}$ 的分布由转移函数和策略 π 的行动值函数 Q 完全决定。当 Q 值收敛时，$X_{s,\pi}$ 的分布也就收敛成对应的最优策略的稳定分布。对叶子节点而言（深度为 H 的节点），Thompson 采样采样以概率 1 收敛到最优动作，因为叶子节点对应的 MAB 问题就默认 Rollout 策略而言是稳定的。当所有的叶子节点收敛时，叶子节点返回值的分布也就收敛了。所以 $H-1$ 层的 MAB 问题就稳定了。进而，Thompson 采样对 $H-1$ 层节点可以收敛到最优动作。递归地，这对所有层次的节点都成立。所以，DNG-MCTS 可以在具有无限计算资源的情况下，找到跟节点对默认 Rollout 策略而言的最优动作。

对 D²NG-POMCP 算法，也有类似的结论。所以，DNG-MCTS 和 D²NG-POMCP 在给定规划时限 H 和默认 Rollout 策略的情况下，最终可以收敛到最优策略。

4.6　实验结果

本节首先给出 Thompson 采样研究动机的实验验证；之后，在不同的标准测试问题里面分布评估和比较 DNG-MCTS 和 D²NG-POMCP 的实验性能。所有的实验都在 Linux 3.8.0 双核 2.80 GHz，8G 内存电脑上运行。



(a) 8 arms.　(b) 32 arms.

(c) 128 arms.　(d) 512 arms.

图 4.1　MAB 问题中 Thompson 采样简单剩余值（Simple Regret）实验结果

4.6.1　研究动机验证实验

本节就简单剩余值在 MAB 问题里面比较 Thompson 采样和其他算法，包括 RoundRobin, Randomized, 0.5-Greedy 和 UCB。RoundRobin 算法轮流选择所有动作[97]；Randomized 算法随机选择一个动作；0.5-Greedy 算法以 0.5 的概率选择实验效果最好的动作，以 0.5 的概率随机选择一个动作[126]；UCB 算法通过最大化 UCB 启发函数选择动作。实验中，每个动作返回一个从 Bernoulli 分布采样得到的随机回报值；UCB 算法的探索因子为 $\sqrt{2}$；Thompson 采样算法选择的共扼分布为 Beta 分布，初始化为 $(\alpha = 1, \beta = 1)$。每个算法在随机运行 1000 次，每次的问题实例都是随机生成的。图 4.1报告了不同大小 MAB 问题的实验结果。可以看出，Thompson 采样算法的简单剩余值比其他算法更小，特别对大规模动作空间问题而言。就累积剩余值而言，Thompson 采样理论上可以达到最优，并且实验性能也非常好。本实验表明，Thompson 采样就简单剩余值而言的实验性能也非常好，为其在 MCTS 上的应用取得成功提供了可能。

4.6.2　MDP 实验

本节主要测试 DNG-MCTS 算法，并跟 UCT 算法做了比较。比较的问题包括*加拿大旅行者问题*（Canadian Traveler Problem, CTP），*赛车问题*（Racetrack Problem）和*帆船问题*（Sailing Problem）。

这三个问题都是基于成本的问题。也就是说，这三个问题使用的是成本函数 $c(s,a)$，而不是回报函数 $R(s,a)$，贝尔曼最优性等式中使用的是 min 操作符，而不是 max 操作符。类似地，基于成本的 MDP 问题的目标是找到一个策略可以对每个状态最小化期望累积成本。注意到只要用 min 操作符代替 max 操作符，为基于回报的 MDP 问题利用的算法可以直接转化并应用到基于成本的 MDP。相应的，转化后的 DNG-MCTS 算法在 Thompson 采样中使用 min 操作符。本节主要基于 MDP-engine*——一个收集 MDP 问题实例和基本算法的软件包——实现算法并进行实验。

在每一个标准测试问题中，本节：1. 从当前状态开始将算法运行一定的迭代次数；2. 执行算法最终返回的贪心动作；3. 重复这个过程直到遇到终止条件（比如达到目标状态，或超过最大迭代次数）；4. 报告折扣累积成本。算法的性能通过 1000 次独立运行的平均折扣累积成本来衡量。在所有的实验中，对所有的 $s \in S, a \in A$ 和 $s' \in S$，$(\mu_{s,0}, \lambda_s, \alpha_s, \beta_s)$ 初始化成 $(0, 0.01, 1, 100)$，$\rho_{s,a,s'}$ 初始化成 0.01。为了公平比较，本节使用了跟文献[57] 一样的设定：1. 对每个状态只选择可以执行的动作；2. 每个状态的可执行动作都在至少执行一次以后才会第二次执行某个动作；3. UCT 算法的探索因子设定成当前的平均行动值函数 $Q(s,a,d)$。

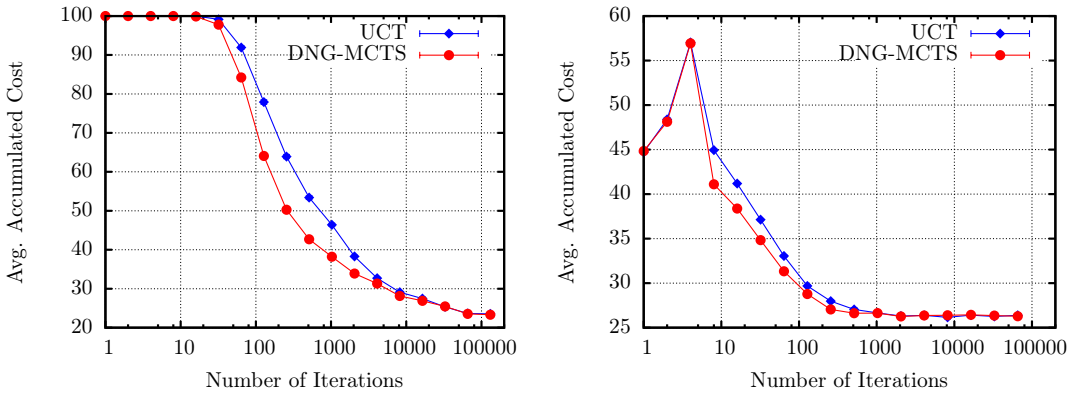
加拿大旅行者问题（CTP）是一个信息不完整的图上路径搜索问题，图上的边以一定的先验概率被堵住[145]。CTP 问题可以被建模成一个确定性 POMDP 问题，即唯一的不确定性来源于初始信念状态。当转化成 MDP 问题后，信念空间的大小为 $n \times 3^m$，其中 n 是节点数目，m 是边的数目。该问题的折扣因子 $\gamma = 1$。求解这个问题的目标是尽快移动到目标节点。该问题最近被 Anytime AO*（AOT）算法[57]，和两个专门算法 UCTB 和 UCTO[146] 探讨过。UCTB 和 UCTO 是 UCT 的特别实现，利用了 CTP 问题的特殊结构，和更高级的基本 Rollout 策略。本实验使用了跟他们论文相同的 10 个 20 节点问题实例。

在 CTP 问题里面运行 DNG-MCTS 和 UCT 算法时，每次决策时的迭代次数为 10000，跟[57] 的设定是一样的。主要使用了两类默认 Rollout 策略：*随机策略*以相同的概率选择每个动作；*乐观策略*假设未知边总是可以通过的，并按照估计的成本来选择动作。表 4.1 显示了实验结果。跟文献[57] 类似，UCTB 和 UCTO 的结果也在表中，用于参考。表中第二栏给出了每个问题转化成 MDP 后的信念空间大小。黑体字显示的是表中最好的结果；灰色背景显示了领域无关的算法中最好的结果。可以看出，使用随机 Rollout 策略时，DNG-MCTS 在一些问题实例里面比 UCT 算法表现更好；特别的，使用了乐观 Rollout 策略的时候，比 UCT

*MDP-engine 代码库见：https://code.google.com/p/mdp-engine/。

表 4.1　20 节点 CTP 实验结果

		领域相关 UCT		随机 Rollout 策略		乐观 Rollout 策略	
问题	信念空间	UCTB	UCTO	UCT	DNG	UCT	DNG
20-1	20×3^{49}	210.7±7	**169.0**±6	216.4±3	223.9±4	180.7±3	177.1±3
20-2	20×3^{49}	176.4±4	**148.9**±3	178.5±2	178.1±2	160.8±2	155.2±2
20-3	20×3^{51}	150.7±7	**132.5**±6	169.7±4	159.5±4	144.3±3	140.1±3
20-4	20×3^{49}	264.8±9	**235.2**±7	264.1±4	266.8±4	238.3±3	242.7±4
20-5	20×3^{52}	123.2±7	**111.3**±5	139.8±4	133.4±4	123.9±3	122.1±3
20-6	20×3^{49}	165.4±6	**133.1**±3	178.0±3	169.8±3	167.8±2	141.9±2
20-7	20×3^{50}	191.6±6	**148.2**±4	211.8±3	214.9±4	174.1±2	166.1±3
20-8	20×3^{51}	160.1±7	**134.5**±5	218.5±4	202.3±4	152.3±3	151.4±3
20-9	20×3^{50}	235.2±6	**173.9**±4	251.9±3	246.0±3	185.2±2	180.4±2
20-10	20×3^{49}	180.8±7	**167.0**±5	185.7±3	188.9±4	178.5±3	170.5±3
total		1858.9	**1553.6**	2014.4	1983.68	1705.9	1647.4



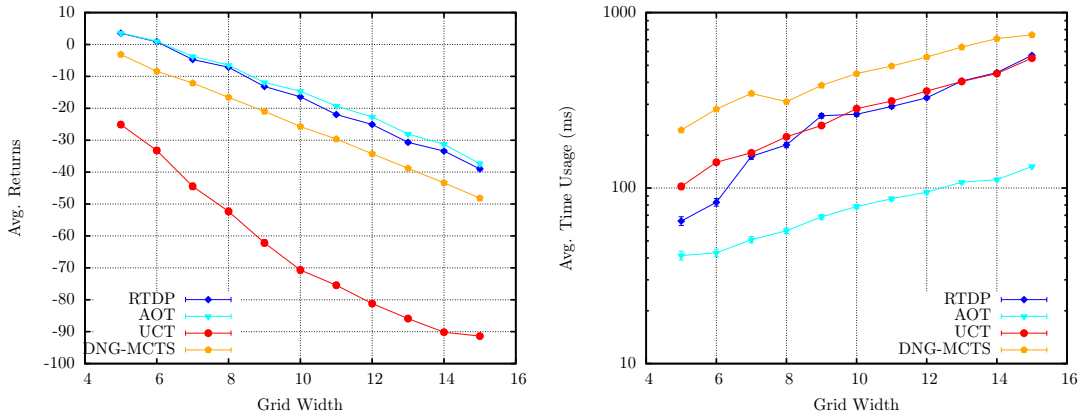
(a) Barto-Big 赛车问题实验结果　　(b) 100 × 100 帆船问题实验结果

图 4.2　赛车问题和帆船问题实验结果

算法好很多。虽然 DNG-MCTS 没有领域相关的专门算法 UCTO 表现好，但跟通用的 UCT 相比仍然是有竞争性的。

赛车问题模拟赛车比赛[55]，一个汽车在一个初始化状态集合里面随机初始化，并向终点运动。在每一个时间周期，汽车可以选择 8 个方向进行加速。运动过程中，汽车有 0.9 的概率成功加速，有 0.1 的概率不能成功加速。本节测试了 DNG-MCTS 和 UCT 算法。测试中，采用随机 Rollout 策略，规划时限为 $H = 100$。采用 *Barto-Big* 实验地图，状态空间大小为 $|s| = 22534$，折扣因子是 $\gamma = 0.95$，最优策略对应的累积成本为 21.38。图 4.2(a)报告了关于迭代次数的平均累积成本的变化情况。每个数据点是 1000 次运行的平均值，每次运行最大运行 100 布。可以看出，就样本复杂度而言，DNG-MCTS 比 UCT 收敛更快。

帆船问题来自于文献[56]。这个问题里面，一个帆船在一个格子地图表示的海面上运动。风的速度根据一定的转移概率随时间而变化。帆船的目标是尽快到达目标格子。帆船在每个周期内可以运动到一个选择的邻居格子内。问题的



(a) 平均回报值　　(b) 平均时间消耗

图 4.3　eTaxi 问题实验结果

折扣因子是 $\gamma = 0.95$，最大规划时限是 $H = 100$。实验中，在 100×100 的问题里面测试了 DNG-MCTS 和 UCT 算法。实验采用随机的 Rollout 策略。该问题具有 80000 个状态，最优累积回报是 26.08。图 4.2(b)显示了平均性能曲线。可以在这个图中看到一个相似的趋势：就样本复杂度而言，DNG-MCTS 比 UCT 算法收敛更快。

特别地，基于原始 Taxi 问题，本节提出了扩展的 Taxi 问题，以更进一步测试算法。在地图边长为 n 的扩展问题（记为 eTaxi[n]）中，格子世界的大小为 $n \times n$。车站 R，G，Y 和 B 分别在位置 $(0,0)$，$(0,n-1)$，$(n-2,0)$ 和 $(n-1,n-1)$。地图中三段墙的长度为 $\lfloor \frac{n-1}{2} \rfloor$，分别在位置 $(0,0)$ 和 $(1,0)$，$(1,n-1)$ 和 $(2,n-1)$，以及 $(n-3,0)$ 和 $(n-2,0)$ 之间。动作空间、转移模型和回报模型跟原始 Taxi 问题是一致的。所以，如果 $n = 5$，eTaxi[n] 问题和原始 Taxi 问题等价。

本节将 DNG-MCTS 算法跟 LRTDP[90]，AOT[57] 和 UCT 进行里比较。注意到，LRTDP 和 AOT 并不是蒙特卡洛算法，也就是说他们使用了问题的完整 MDP 模型。*Min-Min* 启发函数[90] 被用来给 LRTDP 和 AOT 算法初始化新的节点，基于 *Min-Min* 启发函数的贪心策略被 UCT 和 DNG-MCTS 用作 Rollout 策略。实验中，从随机初始化的初始状态运行算法，并报告累积回报和算法的时间消耗。每个动作选择允许的迭代次数为 100；最大搜索深度也为 100。实验测试了地图边长从 5 到 15 之间的 eTaxi 问题。图 4.3(a)和图 4.3(b)报告了 1000 次运行的平均回报值和时间消耗。eTaxi[5] 问题更详细的实验结果见表 4.2。可以从实验结果看出，DNG-MCTS 比 UCT 算法表现出色，同时可以提供跟 LRTDP 和 AOT 相竞争的实验结果。同时也注意到，DNG-MCTS 算法的时间消耗比其他算法更多，这是由于 DNG-MCTS 算法中采样操作比较耗时造成的。

表 4.2　eTaxi[5] 问题详细实验结果

算法	运行次数	平均回报值	平均运行时间 (ms)
LRTDP	1000	3.71 ± 0.15	64.88 ± 3.71
AOT	1000	3.80 ± 0.16	41.26 ± 2.37
UCT	1000	-23.10 ± 0.84	102.20 ± 4.24
DNG-MCTS	1000	-3.13 ± 0.29	213.85 ± 4.75

4.6.3　POMDP 实验

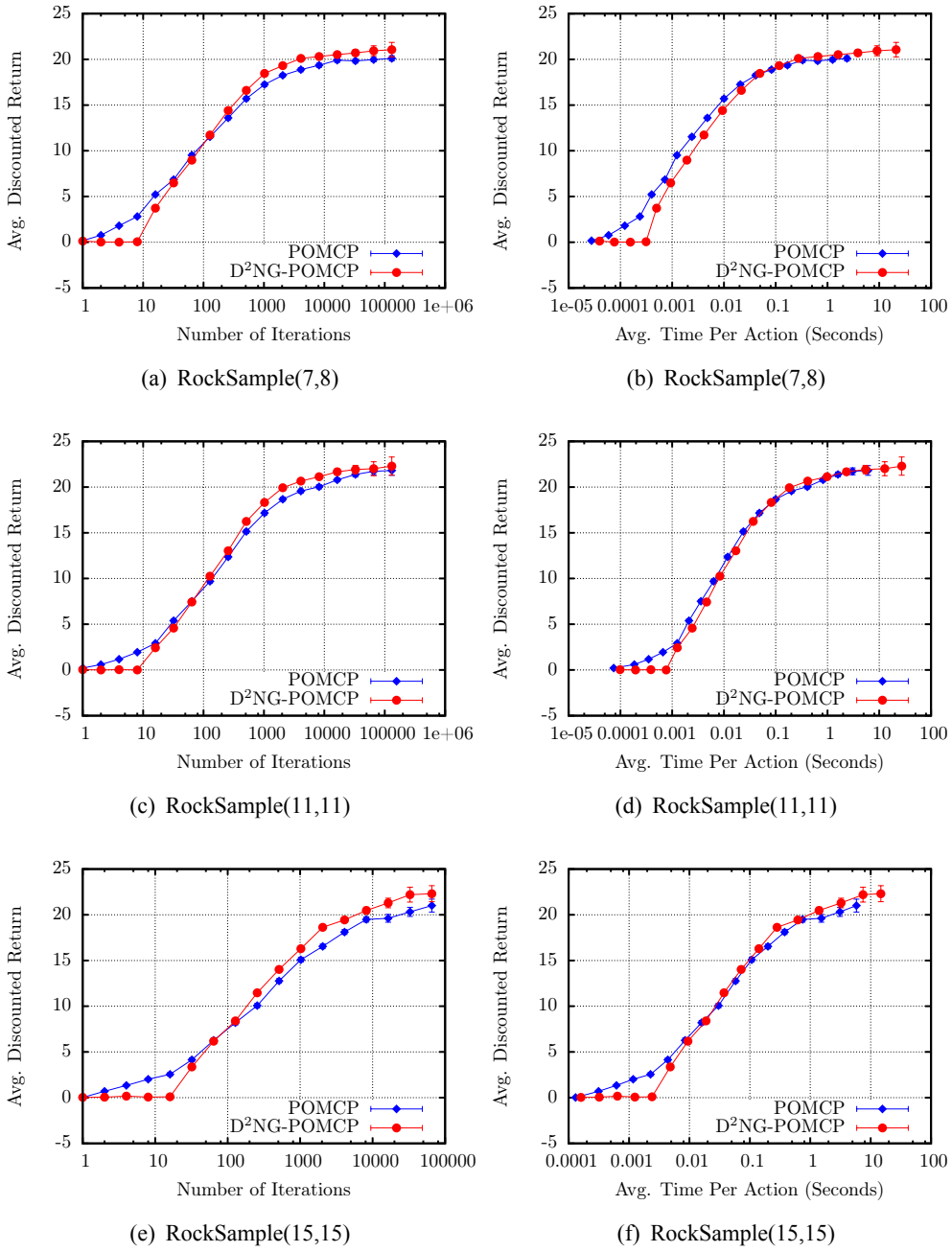
本节在*岩石采样问题*(RockSample Problem)和*吃豆人问题*(PocMan Problem)里面测试 D²NG-POMCP 算法，并跟 POMCP 作比较。D²NG-POMCP 算法的实现主要基于 POMCP* ——一个实现 POMCP 算法和若干标准测试问题的软件包。

对每一个问题实例，本节：1. 从当前历史节点出发将算法运行一定的迭代次数；2. 执行搜索树上的贪心动作；3. 重复这个过程直到遇到终止条件；4. 报告累积折扣回报和平均每个动作选择的计算时间。算法的性能通过 1000 次独立运行的平均累积折扣回报来衡量。所有实验里面，对搜索树所有遇到的 $s \in S$，$a \in A$，$r \in \mathcal{I}$，$o \in O$ 和历史 h，$(\mu_{s,h,0}, \lambda_{s,h}, \alpha_{s,h}, \beta_{s,h})$ 初始化为 $(0, 0.01, 1, 100)$，$\psi_{h,a,r}$ 初始化为 0.01，$\rho_{h,a,o}$ 初始化为 0.01。测试 D²NG-POMCP 和 POMCP 时，使用了文献[81] 里描述的相同的基于优先动作的 Rollout 策略。为了公平比较，本节使用了跟 POMCP 相同的设定，对每一个决策节点：1. 只选择可执行的动作；2. 每个可执行动作至少选择一次后才会第二次选择某个可执行动作。

RockSample(n, k) 问题中，一个机器人在一个包含了 k 块岩石的 $n \times n$ 的格子世界里面进行探索。问题的目标是确定哪些岩石是有价值的，尽可能多地收集有价值的岩石，并安全退出地图。机器人可以执行 3 种动作，即*移动*，*检查*和*采样*。*移动*动作接受方向作为参数，使机器人可以在地图里面移动；*采样*动作收集一块岩石；*检查*动作检查一块岩石是否有价值，检查结果是带有误差的。收集一块有价值的岩石的回报是 $+10$；收集一块无价值岩石的回报是 -10；最终安全退出的回报是 $+10$。其他动作都没有回报或成本。问题的折扣因子是 $\gamma = 0.95$。图 4.4展示了该问题的实验结果。每个数据点是 1000 次运行，或最多 12 小时计算时间的平均值。左边 3 个图像比较了就迭代次数而言的算法性能；右边 3 个图像比较了就每个动作选择平均计算时间而言的算法性能。迭代次数即每次动作选择时 OnlinePlanning 函数调用 D²NG-POMCP 函数的次数；每个动作选择的平均计算时间即只能体每次做决策时实际花费的 OnlinePlanning 函数计算时间的平均值。实验结果显示 D²NG-POMCP 算法就迭代次数而言比 POMCP 收敛更快，就平均每个动作选择的计算时间而言可以跟 POMCP 算法竞争。

表 4.3将 D²NG-POMCP 算法跟其他已有工作进行了比较，包括 AEMS2[80]，HSVI-BFS[75, 77]，SARSOP[76] 和 POMCP[81]。AEMS2 和 HSVI-BFS 是在线算法；

*POMCP 软件包见：http://www0.cs.ucl.ac.uk/staff/D.Silver/web/Applications.html。



(a) RockSample(7,8)　(b) RockSample(7,8)

(c) RockSample(11,11)　(d) RockSample(11,11)

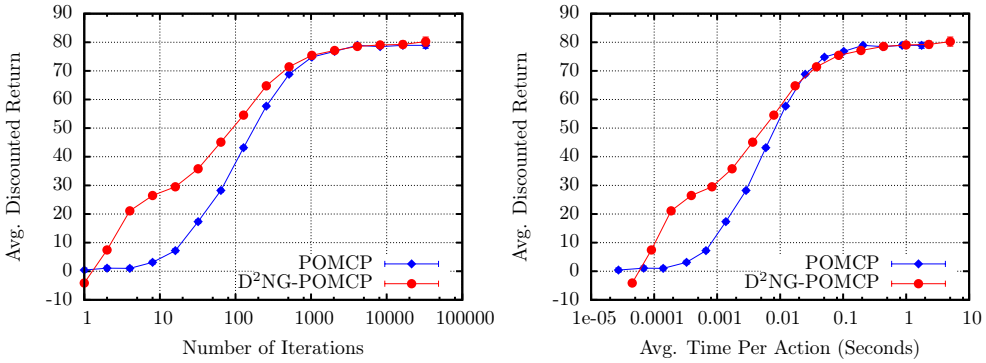
(e) RockSample(15,15)　(f) RockSample(15,15)

图 4.4　RockSample 问题 D²NG-POMCP 实验结果

SARSOP 是离线算法。他们都被提供了完整的 POMDP 模型。AEMS2 和 HSVI-BFS 使用了通过 PBVI 计算出来的值函数作为启发信息[73]；实验结果来自于文献[77]。SARSOP 算法的离线计算时间是 1000 秒；实验结果来自于文献[76]。POMCP 和 D²NG-POMCP 使用了相同的 Rollout 策略。POMCP 的实验结果来自于文献[81]。每个在线算法在动作选择时的计算时间不超过 1 秒。算法性能通过 1000 次运行，或最多 12 小时计算时间的平均折扣累积回报衡量。实验结果显示 D²NG-POMCP 在 RockSample(7, 8) 和 RockSample(11, 11) 问题里面提供了有竞争力的算法性能，在 RockSample(15, 15) 问题里面比 POMCP 好很多。

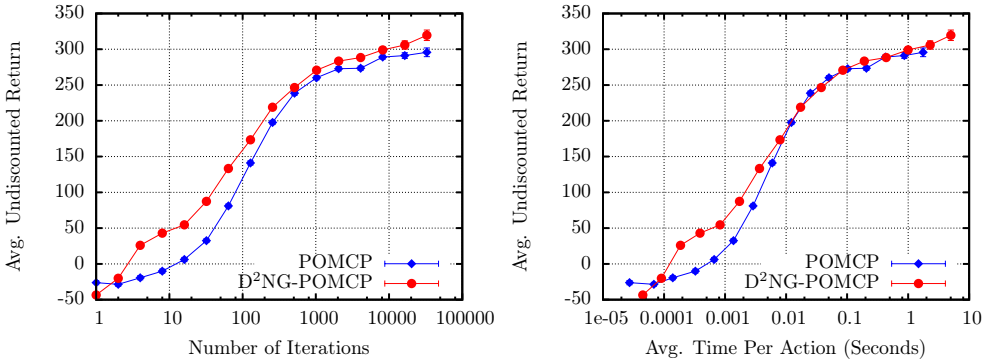
表 4.3　RockSample 问题 D²NG-POMCP 比较实验结果

RockSample	[7, 8]	[11,11]	[15,15]
状态数 |s|	12,544	247,808	7,372,800
AEMS2	21.37 ± 0.22	N/A	N/A
HSVI-BFS	21.46 ± 0.22	N/A	N/A
SARSOP	21.39 ± 0.01	21.56 ± 0.11	N/A
POMCP	20.71 ± 0.21	20.01 ± 0.23	15.32 ± 0.28
D²NG-POMCP	20.87 ± 0.20	21.44 ± 0.21	20.20 ± 0.24



(a) Pocman　(b) Pocman

图 4.5　PocMan 问题 D²NG-POMCP 实验结果——平均折扣回报



(a) Pocman　(b) Pocman

图 4.6　PocMan 问题 D²NG-POMCP 实验结果——平均非折扣回报

PocMan 问题最初由文献[81] 提出。该问题中，一个 PocMan 智能体在一个 17×19 的迷宫中导航，尝试搜寻到尽量多的随机分布的食物和能量药丸。四个幽灵智能体在迷宫里面根据预先设计的随机策略巡逻。如果遇到任何一个幽灵智能体，PocMan 智能体就会死亡，除非它此前 15 个时间周期以内吃过一个能量药丸。每一步行动有 -1 的立即回报值，吃到食物有 $+10$ 的回报值，吃到能量药丸有 $+10$ 的回报值，死亡的回报值是 -100。智能体在每个周期内接受一个 10 比特的观察，对应于智能体的视觉、听觉、触觉和嗅觉感知。PocMan 问题有

大约 10^{56} 个状态，4 个动作和 1024 个观察。问题的折扣因子 $\gamma = 0.95$。用平均累积折扣回报衡量的实验结果见图 4.5。每个数据点是 1000 次运行，或最多 12 小时计算时间的平均值。值得指出的是，算法性能是用平均累积折扣回报衡量的，而不是原始文献里面的平均累积非折扣回报[81]。使用平均累积折扣回报更合理，因为这是算法真实优化的指标。为了提供更全面的比较，使用平均累积非折扣回报衡量的实验结果见图 4.6。该问题中，就迭代次数和平均每个动作选择的计算时间而言，D²NG-POMCP 的实验性能都比 POMCP 好很多。

4.6.4　计算复杂度讨论

考虑到计算复杂度问题，虽然 DNG-MCTS 和 D²NG-POMCP 算法的总计算时间跟是采样次数的线性函数，采样次数最多为 $\mathtt{width} \times \mathtt{depth}$（其中 $\mathtt{width}$ 是迭代次数，$\mathtt{depth}$ 是规划时限），本章提出的算法相比 UCT 和 POMCP 而言确实需要更多的计算时间，这是因为从各种分布里面采样比较耗时。然而，如果仿真过程比较耗时（比如 3D 环境中的物理仿真或多智能体环境中，蒙特卡洛仿真的计算时间超出 MCTS 搜索过程中动作选择的时间），DNG-MCTS 和 D²NG-POMCP 可以获得更好的计算时间复杂度，因为它们具有期望更低的样本复杂度（正如 PocMan 实验中所显示的那样）。

4.7　本章小结

本章主要介绍了 DNG-MCTS 和 D²NG-POMCP 算法。DNG-MCTS 和 D²NG-POMCP 是新颖的基于贝叶斯建模和推理的 MDP 和 POMDP 蒙特卡洛规划 Thompson 采样方法。其基本思想是把蒙特卡洛搜索树上执行一个动作后获得的累积回报的不确定性建模成一些混合分布的组合，根据贝叶斯方法推理出后验分布，并使用 Thompson 采样进行动作选择。提出的算法可以保证在极限情况下收敛到最优策略。MDP 实验结果显示，跟 UCT 相比，DNG-MCTS 在 CTP 问题里面可以提供有竞争力的策略，在 RaceTrack 和 Sailing 问题里面就样本复杂度而言收敛更快。POMDP 实验结果显示 D²NG-POMCP 在 RockSample 和 PocMan 问题里面比目前最前沿的算（包括 AEMS2，HSVI-BFS，SARSOP 和 POMCP）法效果更好。实验结果显示了该后验动作采样技术具有重要的应用和研究潜力。

第五章　基于集合粒子滤波的多对象跟踪算法

内容提要

自主机器人在动态环境中识别、跟踪和确认潜在的多人状态的能力对社会化的人—机器人交互任务非常重要。主要的挑战在于，事先不知道有多少人的情况下，机器人需要通过序列化的无差别观察（包含不可避免的误报和漏报探测）实时估计出当前的多人状态信息。本章主要介绍一个针对该挑战的集合粒子滤波（Particle Filter over Sets，PFS）算法。该算法使用集合来定义联合状态和联合观察，并成功应用相应的运动和观察模型。粒子更新后的，目标确认（Target Identification）问题通过期望—最大化（EM）方法得到解决。基于集合的形式化使得本算法不需要进行显式的观察到目标数据关联。最终提出的 PFS 算法在 PETS2009 数据集上获得了比目前的先进算法更好的实验结果。本章还在真实机器人平台 CoBot 上验证了算法的可行性。剩余内容安排如下：第5.1节介绍基本问题；第5.2节主要介绍相关工作；第5.3节给出具体算法；第5.4节展示实验结果；第5.5节给出本章小结。

5.1　基本介绍

自主机器人在动态环境中识别、跟踪和确认潜在的多人状态的能力对成功完成社会化的人—机器人交互任务非常关键[147, 148]。以 CoBot[149, 150] 机器人试图进入一个电梯作为例子。当电梯门开启时，假设里面有多个人占据了空间，CoBot 需要在几秒钟的时间内跟踪每个人的状态，并且理解每个人到底是要留在电梯里面，还是准备离开电梯的意图。考虑到安全并且友好的跟人类交互，CoBot 只有在确信所有准备离开电梯的人都已经离开的情况下，才能做出进入电梯的决策。当然，这一切的前提是，CoBot 需要以很高的准确度成功跟踪并确认多人的实时状态。

大多数多对象跟踪（Multi-Object Tracking，MOT）方法都遵循一个通过探测跟踪（Tracking-by-Detection）的框架[151, 152]。在该设定下，一个对象探测器（Object Detector）分析每一帧数据，识别对象，把识别的结果作为输入传给一个跟踪器（Tracker）。通过探测跟踪方法可以大致被分成两类：在线和离线。在线跟踪算法在给定过去观察的情况下，按照贝叶斯的方式递归估计出当前状态；离线跟踪算法一般被形式化成一个给定所有观察序列后的，全局组合搜索问题。本章主要关注在线跟踪算法，因为在线跟踪更适合机器人问题上的应用。

在在线多对象跟踪研究领域，大多数已有算法假设一个或多个观察到目标的数据关联假设，并使用贝叶斯滤波方法（比如粒子滤波或卡尔曼滤波）分别更新每一个潜在目标[151, 153, 154]。基于全局最近邻居（Global Nearest Neighbor，GNN）的滤波器根据找到的最小化一个全局目标函数（比如距离或联合概率）的假设

来更新每一个目标的状态[155]。基于联合概率数据关联（Joint Probabilistic Data-Association，JPDA）的滤波器使用根据联合概率（Association Probability）加权后的所有观察来更新每一个状态[153]。多假设跟踪（Multiple Hypothesis Tracking，MHT）方法试图将所有可能的数据关联假设维护在一颗树状结构上，并总是选择“最优”的假设来更新目标状态[154]。考虑到这些算法假设一个数据关联，并对每个潜在目标进行单独的贝叶斯更新，如果某个周期假设发生了错误，估计的状态就会发生错误，并且以后也很难恢复。相反，本章提出的算法通过使用联合状态来编码所有的多目标状态（包括目标个数、每个目标的状态和潜在的所有可能数据关联）避免了直接假设观察到目标的数据关联。更新过程自动对联合状态和数据关联进行贝叶斯最优更新。

本章的主要贡献是最终提出的集合粒子滤波（Particle Filter over Sets，PFS）算法，和引入的一些使得 PFS 成为可能的技术，包括：1. 近似计算观察函数的基于分配和误报—漏报探测的剪枝算法；2. 基于数据关联的粒子改进算法；3. 估计运动和提议权重的贝叶斯概率密度估计方法；4. 基于期望—最大化（EM）的从联合粒子集合里面识别出单个人的目标确认算法。值得一提的是，PFS 及其相关技术是通用的，可以贝广泛用到其他多对象识别问题里面，包括自动监控、交通监控、人—计算机交互等。

为了跟现有算法进行比较，本章在流行的 PETS2009 标准测试数据集里面对 PFS 算法进行了评估。实验结果显示，PFS 算法的实验结果比目前领域里面最先进的算法更好。本章同时通过在 CoBot 上实现 PFS 算法，对算法的可行性进行了验证。

5.2　相关工作

联合多目标概率密度（Joint Multi-Target Probability Density，JMPD）算法跟本章提出的算法类似，使用了联合状态形式化[156] 。在每一帧数据里面，JMPD 假设离散后的像素值（比目标大小更大）作为观察，通过计算每个像素占有目标的个数来近似计算观察函数。JMPD 方法可以忽略底层的数据关联问题，但其跟踪精度非常有限。相反，本章提出的方法把连续的探测结果看成观察，使用一个集合来编码联合状态，并通过考虑所有的数据关联（包含剪枝）来近似计算观察函数。

文献[157] 通过使用基于马尔科夫随机场（Markov Random Field，MRF）的运动模型，利用了基于蒙特卡洛马尔科夫链（Monte-Carlo Markov Chain，MCMC）的粒子滤波算法在联合空间中跟踪多个目标，并允许相距较远的目标被单独跟踪。文献[158] 提出了一个包含多个伪独立的粒子滤波的基于学习的框架。该框架中，每个目标被一个单独的贝叶斯滤波器跟踪，但把其他所有目标过去的联合状态看成是每个滤波器的输入。这两个方法都关注于交互（Interacting）目标的跟踪，但不容易被扩展到具有频繁误报和漏报检测的复杂环境中。

文献[159]提出了一个 Rao-Blackwellized 粒子滤波方法。该方法假设环境中存在常数数量（很多）的目标，但只有不固定的有限数目的目标当前可以被观察到。该方法在每一个粒子内部编码数据关联的概率，并使用若干卡尔曼滤波来单独更新每一个目标。本章提出的方法使用集合的形式化方法在一个粒子滤波器内编码所有可能的数据关联，并通过联合空间内的观察函数的计算自动推理数据关联。

随机有限集合（Random Finite Set，RFS）[160–164]根据一个专门的有限集合统计（Finite Set Statistics，FISST）[165]方法来处理多对象跟踪问题。从数学的角度看 FISST 的主要概念，比如集合积分和集合微分，超出了标准的概率理论。本章提出的方法具有相同的优势，但更简单，并仅依赖于传统的概率概念。

5.3　集合粒子滤波方法

本节主要介绍集合粒子滤波方法（PFS）。

5.3.1　将集合看成一个随机变量

在介绍完整的 PFS 方法之前，本节首先介绍本章把集合看成一个随机变量的方法，特别地，观察到一个集合的概率或概率密度的定义。注意，本章在没有歧义的情况下，交换使用概率和概率密度。

定理 5.3.1. 令随机变量 S 为一个大小为 n 的集合：$S = \{X_i\}_{i=1:n}$，观察到包含 n 个不同元素的集合 $S = \{x_i\}_{i=1:n}$ 的联合概率是 $\Pr(S) = \sum_{\sigma\in A_n} \Pr(X_1 = x_{\sigma(1)}, X_2 = x_{\sigma(2)}, \ldots, X_n = x_{\sigma(n)})$，其中 A_n 是集合 $\{i\}_{i=1:n}$ 的所有置换的集合，$\Pr(X_1 = x_{\sigma(1)}, X_2 = x_{\sigma(2)}, \ldots, X_n = x_{\sigma(n)})$ 是观察到 $X_1 = x_{\sigma(1)} \wedge X_2 = x_{\sigma(2)} \wedge \cdots \wedge X_n = x_{\sigma(n)}$ 的联合概率。

证明. 当观察到 $S = \{x_i\}_{i=1:n}$ 时，每个元素 $x \in \{x_i\}_{i=1:n}$ 到底来自于哪个 S 中的随机变量 $X \in \{X_i\}_{i=1:n}$ 是不清楚的。观察到集合 S 的概率应该把所有的可能性计算在内，也就是所有的随机变量到观察到元素的分配。一个分配 ψ 是一个一一映射 $\psi : \{X_i\}_{i=1:n} \to \{x_i\}_{i=1:n}$，对应于 S 中所有元素的一个置换。 □

推论 5.3.1. 令 $O = \{o_i\}_{i=1:n}$ 为一个包含 n 个不同物体的集合。不重复抽样 k 次，假设结果是一个集合 $S = \{o_{(i)}\}_{i=1:k}$。观察到 S 的联合概率是 $\Pr(S) = k!\frac{1}{n(n-1)\cdots(n-k+1)} = \frac{1}{\binom{n}{k}}$，其中 $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ 是二项系数。

推论 5.3.2. 令 X 为一个服从分布 $f_X(x)$ 的随机变量，$S = \{X_i\}_{i=1:n}$ 是一个随机变量的集合。S 中的所有元素跟 X 独立同分布。观察到包含 n 个不同元素的集合 $S = \{x_i\}_{i=1:n}$ 的联合概率是 $\Pr(S) = n! \prod_{1\leqslant i\leqslant n} f_X(x_i)$。

注意到，当说到 $S = \{X_i\}_{i=1:n}$ 是一个包含 n 个元素的随机变量集合时，根据集合的定义，已经隐含了 S 中的所有元素都不相同。作为例子，假设两次投掷一枚均匀的硬币，表示成集合的可能结果有 3 个：$\{\mathrm{Head}\}$，$\{\mathrm{Head}, \mathrm{Tail}\}$ 和 $\{\mathrm{Tail}\}$。根据推论5.3.2，$\Pr(\{\mathrm{Head}, \mathrm{Tail}\}) = 1/2$——是正确的。然而，推论5.3.2并不能覆盖两次都是 Head 或两次都是 Tail 的情况。

5.3.2　隐马尔科夫模型形式化

本节介绍 PFS 算法把多对象跟踪问题建模成隐马尔科夫模型的方法。

5.3.2.1　运动模型

形式上，联合状态被定义成一个包含多人状态的集合 $S = \{s_i\}_{i=1:|S|}$。一个人的状态被表示成一个多维向量 $s = (x, y, \dot{x}, \dot{y})$，其中 (x, y) 和 $(\dot{x}, \dot{y})$ 分别是世界坐标系中的位置和速度。本章假设每个人的移动都是独立的，服从一个随机加速模型：$(x, y) \leftarrow (x, y) + (\dot{x}, \dot{y})\tau + \frac{1}{2}(\ddot{x}, \ddot{y})\tau^2$，和 $(\dot{x}, \dot{y}) \leftarrow (\dot{x}, \dot{y}) + (\ddot{x}, \ddot{y})\tau$，其中 τ 是更新的时间间隔，$(\ddot{x}, \ddot{y})$ 是随机加速度，计算为：$(\ddot{x}, \ddot{y}) = (p\cos\theta, p\sin\theta)$，其中 $p \sim \mathcal{N}(0, \sigma_p^2)$ 是加速度大小（Dash Power），$\theta \sim \mathcal{U}(0, 2\pi)$ 是加速方向（Dash Direction）。这里，$\mathcal{N}$ 和 $\mathcal{U}$ 分布表示高斯分布和均匀分布；σ_p^2 是加速度大小的方差。更进一步，本章假设人在机器人*视觉范围*（Filed of View，FOV）内按照一个*生灭过程*（Birth-Death Process）产生和消失，产生速率为每秒 λ，消失速率为每秒 $|S|\mu$。

5.3.2.2　观察模型

一个观察被定义成若干探测结果的集合，$O = \{o_i\}_{i=1:|O|}$。本章假设，一个探测结果 $o = (x, y, c)$ 包含一个世界坐标系中的位置 (x, y) 和一个置信度 $c \in [0, 1]$。置信度的值反映了*人体探测器*（Human Detector）对某一探测结果的内部分类置信度。举例来说，该置信度可能来自于探测算法内部的*支持向量机*（Support Vector Machine，SVM）的边缘距离[166]。如果底层探测器不能提供置信度，那么可以仅仅使用默认值（比如 0.5），所以本章对探测结果的假设是通用的。

下面首先考虑只有一个人和一个探测结果的情况。给定人的状态 $s = (x, y, \dot{x}, \dot{y})$，$\Pr(o \mid s)$ 表示观察到探测结果 $o = (x', y', c)$ 的概率，计算为：$\Pr(o \mid s) = \Pr(c \mid 1)\Pr(x', y' | x, y)$，其中 $\Pr(c \mid 1)$ 是实际有人的情况下，观察到置信度为 c 的概率，$\Pr(x', y' \mid x, y)$ 是人在位置 (x, y) 的情况下，观察到人在位置 (x', y') 的概率。本章使用 Beta 分布建模 $\Pr(c \mid 1) = \mathrm{Beta}(c \mid 2, 1)$，高斯分布建模 $\Pr(x', y' \mid x, y) = \mathcal{N}(x', y' \mid x, y, \Sigma)$，其中 Σ 是协方差。

在有*误报探测*（False Detection）的情况下，令 $\Pr(o \mid \varnothing)$ 表示观察到探测结果 $o = (x', y', c)$ 的概率，计算为 $\Pr(o \mid \varnothing) = \Pr(c \mid 0) f_b(x', y')$，其中 $\Pr(c \mid 0)$ 是实际没有人的情况下，探测到一个置信度为 c 的结果的概率，$f_b(x', y')$ 给出了误报探测出现在位置 (x', y') 的背景概率。本章使用 Beta 分布建模 $\Pr(c \mid 0) = \mathrm{Beta}(c \mid 1, 2)$，整个监控范围内的均匀分布建模 f_b。

在通常情况下，本章假设每个时间周期内，一个真实存在的目标最多产生一个探测结果，一个探测结果最多来源于一个目标。令 $F \subseteq O$ 和 $M \subseteq S$ 分别为误报和*漏报探测*（Missing Detection）的集合。每个可能的 F 和 M 集合的组合必

须满足：$|O-F|=|S-M|$。使用 $O\circ S=\{\langle F_i,M_i\rangle\}_{i=1:|O\circ S|}$ 表示所有可能 F-M 对的集合，则 $|O\circ S|=\sum_{0\leqslant i\leqslant\min\{|O|,|S|\}}\binom{|O|}{i}\binom{|S|}{i}=\binom{|O|+|S|}{|O|}$。

本章假设误报和漏报探测分别服从参数为 ν 和 $|S|\xi$ 的泊松分布。假设更新间隔是 τ，根据推论 5.3.1 和 5.3.2，联合观察函数为：

$$\Pr(O\mid S)=\sum_{\langle F,M\rangle\in O\circ S}\Pr(O-F\mid S-M)\cdot(\nu\tau)^{|F|}e^{-\nu\tau}\prod_{o\in F}P(o\mid\varnothing)\frac{(|S|\xi\tau)^{|M|}e^{-|S|\xi\tau}}{|M|!}\frac{1}{\binom{|S|}{|M|}},\quad(5.1)$$

其中 $\Pr(O-F\mid S-M)$ 给出真实状态和观察大小相同时的概率。为方便起见，下面定义 $f_F(F)=(\nu\tau)^{|F|}e^{-\nu\tau}\prod_{o\in F}P(o\mid\varnothing)$，和 $f_M(M)=\frac{(|S|\xi\tau)^{|M|}e^{-|S|\xi\tau}}{|M|!}\frac{1}{\binom{|S|}{|M|}}$。令 Ψ_{S-M}^{O-F} 表示集合 $S-M$ 到 $O-F$ 的所有可能分配，假设观察自己是条件独立的，则：

$$\Pr(O-F\mid S-M)=\sum_{\psi\in\Psi_{S-M}^{O-F}}\prod_{s\in S-M}\Pr(\psi(s)\mid s).\quad(5.2)$$

结合公式 5.1 和 公式 5.2，可以活动最终的完整观察函数。完整观察函数包含的项数是：$\sum_{0\leqslant i\leqslant\min\{|O|,|S|\}}\binom{|O|}{i}\binom{|S|}{i}i!=\Omega((\frac{\max\{|O|,|S|\}}{e})^{\min\{|O|,|S|\}})$。哪怕对中等规模的状态和观察数，实时完整计算这个函数是不可能的。实际实现时必须进行近似计算。

5.3.3　观察函数近似

本节主要介绍 PFS 算法中观察函数的近似计算方法。

5.3.3.1　分配剪枝

公式 5.2 总共有 $m!$（$m=|S-M|=|O-F|$）项，实际求解非常困难。一般来说，并不是所有的分配都需要被考虑，因为这些分配中的大多数跟最优分配相比，概率都非常小，特别对 $m>2$ 的情况。为此，把概率 $\Pr(o\mid s)$ 转化成成本 $c(s,o)=-\log(\Pr(o\mid s))$，并使用 Murty[167] 算法按照成本增加的顺序美剧所有分配，直到最后一个分配相比最优分配的概率比小于一个阈值。事实上，优化后的 Murty 算法在 $N\times N$ in $O(kN^3)$ 时间内找到一个大小为 $N\times N$ 的分配问题的前 k（Top k）个分配方案[168]。

5.3.3.2　误报—漏报剪枝

所有可能的 F-M 对的大小为 $\binom{|O|+|S|}{|O|}$，为完整计算观察函数带来了巨大的复杂度。误报—漏报剪枝的思想是借助于优先队列，按照概率降低的顺序找到所有的 F-M 对 $\langle F,M\rangle$，知道 $f_F(F)f_M(M)$ 的值比某一阈值更小。最终的包含了

Input: A set of detections O, and a set of humans S
Output: Probability of observing O given S

Let $Q \leftarrow$ a descending priority queue initially empty
Let $\mathcal{F} \leftarrow$ a list of all possible false detections F
Let $\mathcal{M} \leftarrow$ a list of all possible missing detections M
Sort $\mathcal{F}$ according to $f_F(\cdot)$ in descending order
Sort $\mathcal{M}$ according to $f_M(\cdot)$ in descending order
Add $(1,1)$ to Q with priority $f_F(\mathcal{F}[1])f_M(\mathcal{M}[1])$
Let $p \leftarrow 0$
repeat
    Let $(i,j) \leftarrow \mathsf{Pop}(Q)$
    Let $q \leftarrow f_F(\mathcal{F}[i])f_M(\mathcal{M}[j])$
    if $|\mathcal{F}[i]| = |\mathcal{M}[j]|$ **then**
        $p \leftarrow p + q\ \mathsf{Murty}(\mathcal{F}[i], \mathcal{M}[j])$
    end
    if $i+1 \leqslant |\mathcal{F}|$ **then**
        Add $(i+1, j)$ to Q with priority $f_F(\mathcal{F}[i+1])f_M(\mathcal{M}[j])$
    end
    if $j+1 \leqslant |\mathcal{M}|$ **then**
        Add $(i, j+1)$ to Q with priority $f_F(\mathcal{F}[i])f_M(\mathcal{M}[j+1])$
    end
until $q <$ *threshold* **or** Q *is empty*
return p

算法 5.1: 联合观察函数的近似算法

分配剪枝和误报—漏报剪枝的算法见算法 5.1，其中 Murty 函数使用上一节介绍的分配剪枝技术近似计算公式 5.2。

证明. 为简单起见，定义 $f_{FM}(i,j) = f_F(\mathcal{F}[i])f_M(\mathcal{M}[j])$。在算法主循环第 k 次 $(1 \leqslant k \leqslant |\mathcal{F}||\mathcal{M}|)$ 迭代期间，令 Q_k 为出队（Pop）操作前的优先队列，令 (i_k, j_k) 出队的元素，则 $(i_k, j_k) = \operatorname{argmax}_{(i,j)\in Q_k} f_{FM}(i,j)$，并且 $Q_{k+1} \cup (i_k, j_k) = Q_k \cup \mathbf{1}[i_k+1 \leqslant |\mathcal{F}|](i_k+1, j_k) \cup \mathbf{1}[j_k+1 \leqslant |\mathcal{M}|](i_k, j_k+1)$。因为，$f_{FM}(i_k+1, j_k) \leqslant f_{FM}(i_k, j_k)$，以及 $f_{FM}(i_k, j_k+1) \leqslant f_{FM}(i_k, j_k)$，有 $f_{FM}(i_{k+1}, j_{k+1}) \leqslant f_{FM}(i_k, j_k)$ for $1 \leqslant k \leqslant |\mathcal{F}||\mathcal{M}| - 1$。所以 算法 5.1 以概率递减的顺序找到所有的 F-M 对。 □

5.3.4 粒子滤波

通过把一个粒子定义成一个联合状态 $X = \{s_i\}_{i=1:|X|}$，给定到时间 t 为止的观察后的，联合状态的后验分布 $\Pr(S_t \mid O_1, O_2, \ldots, O_t)$ 被近似成一组粒子的集合 $\mathcal{P}_t = \{\langle X_t^{(i)}, w_t^{(i)}\rangle\}_{i=1:N}$，使得 $\sum_{i=1}^{N} w = 1$。假设新粒子是按照*提议分布*（Proposal Distribution）$\pi(\cdot \mid X_{t-1}, O_t)$ 提议得到的，一步粒子滤波的主要过程按如下方式进行：

1. 提议：对 $1 \leqslant i \leqslant N$，$\hat{X}_t^{(i)} \sim \pi(\cdot \mid X_{t-1}^{(i)}, O_t)$,

2. 更新：对 $1 \leqslant i \leqslant N$,

   (a) 运动权值（Motion Weight）：$m_t^{(i)} = \Pr(\hat{X}_t^{(i)} \mid X_{t-1}^{(i)})$,

   (b) 观察权值（Observation Weight）：$o_t^{(i)} = \Pr(O_t \mid \hat{X}_t^{(i)})$,

   (c) 提议权值（Proposal Weight）：$p_t^{(i)} = \pi(\hat{X}_t^{(i)} \mid X_{t-1}^{(i)}, O_t)$,

   (d) 非归一化的粒子权值：$\tilde{w}_t^{(i)} = w_{t-1}^{(i)} \frac{m_t^{(i)} o_t^{(i)}}{p_t^{(i)}}$.

3. 归一化：对 $1 \leqslant i \leqslant N$，$\hat{w}_t^{(i)} = \frac{\tilde{w}_t^{(i)}}{\sum_{1 \leqslant j \leqslant N} \tilde{w}_t^{(j)}}$.

4. 重采样（Resample）：对 $1 \leqslant i \leqslant N$，根据 $\sum_{1 \leqslant i \leqslant N} \hat{w}_t^{(i)} \delta_{\hat{X}_t^{(i)}}(X_t^{(i)})$，从 $\{\langle \hat{X}_t^{(i)}, \hat{w}_t^{(i)} \rangle\}_{i=1:N}$ 中，采样 $X_t^{(i)}$；返回 $\mathcal{P}_t = \{\langle X_t^{(i)}, \frac{1}{N} \rangle\}_{i=1:N}$.

在粒子滤波的通常实现中，新的粒子一帮直接根据运动模型提出，这种情况下，粒子权值的更新简化成 $w_t \leftarrow w_{t-1} o_t$。而这种简单的提议策略并不适用于多对象跟踪问题，因为根据运动模型提议出来的人的状态符合观察模型的概率极小。为了克服这个难题，本章利用了一个基于数据关联的粒子改进策略，使包含更多信息量的提议成为可能。

5.3.4.1　粒子改进策略

对于探测结果 $o = (x', y', c)$，在贝叶斯设定下，假设先验概率 $\Pr(1) = \Pr(0) = 0.5$，那么 o 不是误报探测的概率是 $\Pr(1 \mid c) = \frac{\Pr(c|1)\Pr(1)}{\Pr(c|1)\Pr(1)+\Pr(c|0)\Pr(0)} = c$。在高斯假设下，如果人在任何一个位置出现的先验分布是均匀的，给定 o 不是误报探测，那么 o 来自于状态 $s = (x, y, \cdot, \cdot)$ 的概率 $\Pr(s \mid o) = \eta \Pr(o \mid s) \Pr(s) = \mathcal{N}(x', y' \mid x, y, \Sigma)$。所以，对于新的探测 $o \in O$，理论上就可以根据概率 c 按照分布 $\Pr(s \mid o)$ 提议其源自的状态 s。这里把这个混合提议分布记为 $\pi_s(\cdot \mid o)$。如果 $o = (x', y', c)$，并且 $s = (x, y, \cdot, \cdot)$，那么 $\pi_s(\varnothing \mid o) = 1 - c$，并且 $\pi_s(s \mid o) = c\mathcal{N}(x', y' \mid x, y, \Sigma)$。注意到 s 的速度在该提议分布里面被忽略了。现在的问题是，对每一个粒子 X，如何确定每一个探测结果 $o \in O$ 是新的探测还是源自于某个 X 中已经存在的人。PFS 算法通过 X 和 O 之间的最可能的数据关联来找到 O 中所有可能的新的探测结果。

形式上，粒子 X 和观察 O 之间的一个数据关联可以被定义成一个 3 元组 $\varphi = \langle F, M, \psi \rangle$，其中 $F \subseteq X$ 是误报探测的集合，$M \subseteq O$ 是漏报探测的集合，并且 $\psi \in \Psi_{X-M}^{O-F}$ 是 $X - M$ 到 $O - F$ 之间的分配。公式 5.1 可以被重写为 $\Pr(O \mid X) = \sum_{\varphi} \Pr(O, \varphi \mid X)$。令 $\varphi^* = \operatorname{argmax}_{\varphi} \Pr(O, \varphi \mid X)$ 为就观察似然性（Observation Likelihood）而言最优的数据关联。假设 $\varphi^* = \langle F^*, M^*, \psi^* \rangle$，那么 F^* 就是给定 X 后，直观上最可能的新探测的集合。最终得到的提议分布 $\pi_r(\cdot \mid X_{t-1}, O_t)$ 如下：

1. 根据运动模型采样 $X'_t \sim \Pr(\cdot \mid X_{t-1})$,
2. 找到最优数据关联 $\varphi^* = \langle F^*, M^*, \psi^* \rangle$ given X'_t,
3. 提议新的状态集合 $X' = \{s \mid s \sim \pi_s(\cdot \mid o), o \in F^*\}$,
4. 提议一个改进的粒子 $X''_t \leftarrow X'_t \cup X'$,
5. 返回 $\hat{X}_t \leftarrow \operatorname{argmax}_{X \in \{X'_t, X''_t\}} \Pr(O_t \mid X)$.

注意到，第2步可以被通过简单运行算法 5.1来近似。第5步被用来作为一个*接受测试*（Acceptance Test）步骤，以保证就观察 O 而言返回一个更好的提议。算法 5.1运行的中间结果被缓存，并在任何可能的情况下，在第2步、第5步和以后的更新步骤中被重用。最终得到的提议分布就探测新出现的目标而言，非常高效。事实上已经不需要运动模型中提议任何新出现的状态。实验中，运动模型中的产生速率被简单设置成 0。

5.3.4.2　贝叶斯概率密度估计

使用改进后的粒子提议分布，就必须为提议出来的粒子计算运动和提议权值。然而，在粒子滤波框架中，并不必须拥有运动模型的显式表达式。为了克服这个困难，本节提出一个基于贝叶斯的概率密度估计方法。

对粒子集合 $\mathcal{P} = \{X_i\}_{i=1:N}$，令 $\mathcal{P}'$ 为直接根据运动模型提议出来的粒子集合：$\mathcal{P}' = \{X' \mid X' \sim \Pr(\cdot \mid X), X \in \mathcal{P}\}$，$\mathcal{P}''$ 为改进后的粒子集合：$\mathcal{P}'' = \{X'' \mid X'' \sim \pi_r(\cdot \mid X'), X' \in \mathcal{P}'\}$。根据文献[169] 的思路，这里把 $\mathcal{P}'$ 和 $\mathcal{P}''$ 看成数据，并在其上建立*概率密度估计*（Probability Density Estimation）以近似估计运动和提议权值，具体地为：$\Pr(X'' \mid X) \approx \Pr(X'' \mid \mathcal{P}')$，以及 $\pi_r(X'' \mid X) \approx \Pr(X'' \mid \mathcal{P}'')$，其中 X'' 是根据 X 提议并改进后的粒子集合。

假设每个人的状态独立同分布，服从某一未知分布 f_s，那么根据推论5.3.2，定义在集合上，$\Pr(X \mid \mathcal{P}) = n! \Pr(|X| = n \mid \mathcal{P}) \prod_{s \in X} f_s(s \mid \mathcal{P})$。进一步，这里假设人的数目服从泊松分布，参数为 γ。假设 γ 的先验分布为 Gamma 分布，参数为 (α_0, β_0)，那么 γ 的后验分布也是 Gamma 分布，根据贝叶斯方法更新后的参数为 $(\alpha = \alpha_0 + \sum_{X \in \mathcal{P}} |X|, \beta = \beta_0 + N)$。所以人的数目的*后验预测*（Posterior Predictive）是一个*负二项分布*（Negative Binomial Distribution）。该负二项分布的失败次数参数为 $r = \alpha$，成功率参数为 $p = \frac{1}{1+\beta}$。也就是说，$\Pr(|X| = n \mid \mathcal{P}) = \mathcal{NB}(n; r, p) = \binom{n+r-1}{n} p^n (1-p)^r$.

接下来，忽略状态 $s = (x, y, \dot{x}, \dot{y})$ 中的速度 $(\dot{x}, \dot{y})$，并假设 x 和 y 是独立的，使用*多变量核密度估计*（Multivariate Kernel Density Estimator）来估计 $f_s(s \mid \mathcal{P})$。一个多变量核密度估计方法根据从目标分布 f 中采样的样本集合来估计 f。令 $\{\mathbf{x}_i\}_{i=1:n}$ 为服从分布 f 的样本集合，f 的核密度估计为 $\hat{f}(\mathbf{x} \mid \{\mathbf{x}_i\}_{i=1:n}) = \frac{1}{n} \sum_{1 \leqslant i \leqslant n} K(\mathbf{x} - \mathbf{x}_i)$，其中 $K(\cdot)$ 是*核函数*（Kernel Function）。核函数为对称的

多变量概率密度函数。用 $\mathcal{H}(\mathcal{P}) = \{s \mid s \in X, X \in \mathcal{P}\}$ 表示粒子集合 $\mathcal{P}$ 中所有的人的状态，f_s 近似为 $f_s(s \mid \mathcal{P}) \approx \frac{1}{|\mathcal{H}(\mathcal{P})|} \sum_{s' \in \mathcal{H}(\mathcal{P})} \phi(x - x')\phi(y - y')$，其中 $s = (x, y, \cdot, \cdot)$，并且 ϕ 为标准高斯函数。实际实现时，随机在 $\mathcal{H}(\mathcal{P})$ 中抽样一个子集，而不是考虑所有的人的状态 $\mathcal{H}(\mathcal{P})$。因为，后者会非常耗时。

5.3.5　个体确认算法

虽然更新后的粒子集合 $\mathcal{P}_t$ 已经完整编码了联合状态的后验分布，一个**个体确认**（Human Identification）过程被用来从联合状态的粒子集合中识别并报告每一个潜在的个体。识别出来的若干个体可能对高层认为而言是更有用的。一个**已确认个人**（Identified Human）——简写为**个体**（Identity）——定义为一个 3 元组 $h = (s, c, \rho)$，其中 s 是期望状态，$c \in [0, 1]$ 是置信度，ρ 是一个唯一编号（ID）。个体 h 是从一个与之关联的状态子集中估计出来的 $\mathcal{H}(h) \subseteq \mathcal{H}(\mathcal{P}_t)$：$s = \frac{1}{|\mathcal{H}(h)|} \sum_{s' \in \mathcal{H}(h)} s'$，并且 $c = \frac{|\mathcal{H}(h)|}{N}$。这个状态集合在本章中又被为**状态池**（State Pool）。

令 $L_t = \{h_i\}_{i=1:|L_t|}$ 表示周期 t 时的个体集合。注意到，初始时，$L_0 = \varnothing$。对每一个探测结果 $o \in O_t$，首先提议一个新的个体 h_o，其状态池 $\mathcal{H}(h_o)$ 为空。令 $L_{O_t} = \{h_o \mid o \in O_t\}$ 所有潜在的新的个体，那么周期 t 的所有候选个体集合为 $C_t = L_{t-1} \cup L_{O_t}$。正如前文所指出的，每一个个体 $h \in C_t$ 都被关联一个状态池 $\mathcal{H}(h)$，这等价于为每个状态 s 进行**标记**（Labeling），使得 $\mathcal{H}(h) = \{s \mid l(s) = h, s \in \mathcal{H}(\mathcal{P}_t)\}$，同时满足：1. 对任意 $s \in \mathcal{H}(\mathcal{P}_t)$，$\exists! h \in C_t : l(s) = h$; 2. 对任意 $X \in \mathcal{P}_t$，$s_1 \in X$，$s_2 \in X$ 并且 $s_1 \neq s_2$，那么 $l(s_1) \neq l(s_2)$。令 f_h 为个体 $h \in C_t$ 的状态分布，$\mathbf{P} = \{f_h \mid h \in C_t\}$ 为所有个体状态分布的集合，本章把个体确认过程看成是找出最好的 $\mathbf{P}^*$ 估计，从而可以最优的解释更新后的粒子。形式上，$\mathbf{P}^* = \text{argmax}_{\mathbf{P}} \max_l \Pr(\mathcal{P}_t, l \mid \mathbf{P})$。

这里使用一个期望—最大化（EM）过程来找到**最大后验概率**（Maximum a Posterior，MAP），按照以下步骤迭代计算——基本思想跟 K-均值（K-Means）算法类似[170]：

E step: $l^{(k)} = \text{argmax}_l \Pr(\mathcal{P}_t, l \mid \mathbf{P}^{(k-1)})$,

M step: $\mathbf{P}^{(k)} = \text{argmax}_{\mathbf{P}} \Pr(\mathcal{P}_t, l^{(k-1)} \mid \mathbf{P})$.

E 步骤等价于，给定 $f_h \in \mathbf{P}^{(k-1)}$ 的条件下，找到一个标记方法 l^*，使得联合概率 $\prod_{s \in \mathcal{H}(\mathcal{P}_t)} f_{l^*(s)}(s)$ 取得最大值。可以归纳证明这个标记方法是存在的。显然，这又等价于对每个粒子 $X \in \mathcal{P}_t$，使得 $\prod_{s \in X} f_{l^*(s)}(s)$ 取最大值，进而退化成 N 个最优分配子问题。

证明. 周期 0 时，因为所有的新的人的状态都是根据改进后的提议分布提议的，对每个粒子 $X \in \mathcal{P}_0$ 和每个人的状态 $s \in X$，s 一定是根据某一个探测结果 $o \in O_0$ 提议出来的，所以 $|X| \leqslant |O_0|$。因为 $|C_0| = |L_{O_0}| = |O_0|$，所以

$\forall X \in \mathcal{P}_0 : |X| \leqslant |C_0|$。所以周期为 0 时，标记方法 l_0 存在。假设周期 $t \leqslant T$ 时，标记方法都存在，周期 T 的标记为 l_T，那么 $L_T = \{l_T(s) \mid s \in X, X \in \mathcal{P}_T\}$。所以，$\forall X \in \mathcal{P}_T : |X| \leqslant |L_T|$。周期 $T+1$ 时，根据定义 $C_{T+1} = L_T \cup L_{O_{T+1}}$，并且 $L_T \cap L_{O_{T+1}} = \varnothing$，所以 $|C_{T+1}| = |L_T| + |L_{O_{T+1}}|$。对每一个粒子 $X \in \mathcal{P}_{T+1}$，$X = (X_- - X') \cup X_{O'}$，其中 $X_- \in \mathcal{P}_T$ 是粒子 X 的源，也就是说 X 在粒子滤波过程中直接继承于 X_-，另外，$X' \subseteq X_-$ 是根据运动模型消失的人的状态集合，$X_{O'}$ 是周期 $T+1$ 根据探测结果子集 $O' \subseteq O_{T+1}$（如果有的话）新提议的新的人的状态集合。注意到 $X_- \cap X_{O'} = \varnothing$，所以 $|X| = |X_-| - |X'| + |X_{O'}|$。又因为，$|X_-| \leqslant |L_T|$，并且 $|X_{O'}| = |O'| \leqslant |O_{T+1}| = |L_{O_{T+1}}|$，所以 $|X| \leqslant |C_{T+1}|$ 对所有 $X \in \mathcal{P}_{T+1}$ 都成立。所以，l_{T+1} 存在。归纳得到，对所有周期 $t \geqslant 0$，E 步骤中的需要寻找的标记方法 l_t 都存在。 □

M 步骤通过考虑当前的观察 O_t，按照*最大似然估计*（Maximal Likelihood Estimation，MLE）的方式进行近似计算。为此，每一个探测结果 $o \in O_t$ 被关联到一个状态子集 $\mathcal{H}(o) \subseteq \mathcal{H}(\mathcal{P}_t)$。对所有的粒子 $X \in \mathcal{P}_t$，这个子集的构造过程如下：1. 给定 X，选择最有可能的数据关联 $\langle F^*, M^*, \psi^* \rangle = \varphi^* = \operatorname{argmax}_\varphi \Pr(O_t, \varphi \mid X)$；2. 对所有 $s \in X$，按照 $\mathcal{H}(\psi^*(s)) \leftarrow \mathcal{H}(\psi^*(s)) \cup s$ 更新 $\mathcal{H}(\psi^*(s))$。注意到，粒子滤波过程中缓存的计算结果在这里被重用。对每一个个体 $h \in C_t$，$f_h^{(k)}$ 按照如下方式计算：$f_h^{(k)}(s) = \sum_{o \in O_t} f_h(s, o) + f_h(s, \varnothing) = \sum_{o \in O_t} \Pr(s \mid o) f_h(o) + f_h(s, \varnothing) = \sum_{o \in O_t} \mathbf{1}[s \in \mathcal{H}(o)] f_h(o) + \mathbf{1}[\forall o : s \notin \mathcal{H}(o)] f_h(\varnothing)$，其中 $f_h(o)$ 是探测结果 o 由个体 h 产生的概率，$f_h(\varnothing)$ 是个体 h 没有产生探测结果的概率。下一步，近似计算 $f_h(o) = \Pr(o \mid h) \Pr(h) \approx \frac{1}{N} |\mathcal{H}(o) \cap \mathcal{H}(h)|$，并且 $f_h(\varnothing) = \Pr(\varnothing \mid h) \Pr(h) \approx \frac{1}{N} \left| \mathcal{H}(h) - \bigcup_{o \in O_t} \mathcal{H}(o) \cap \mathcal{H}(h) \right|$。所以，$\mathbf{P}^{(k)} = \{f_h^{(k)} \mid h \in C_t\}$。

最终的算法首先运行 M 步骤，$l^{(0)}$ 初始化成上周期收敛或最终的标记方法，同时需要考虑粒子滤波过程中的状态删除、增加和重复操作。然后，迭代地提出新的概率解释 $\mathbf{P}^{(k)}$ 和新的标记方法 $l^{(k+1)}$，知道收敛，或超过最大迭代次数。最终的标记方法被用来构造 L_t 如下：$L_t = \{l(s) \mid s \in \mathcal{H}(\mathcal{P}_t)\}$。注意到，$|L_t| \leqslant |C_t|$，这是因为最终状态池为空的候选个体不会被加到 L_t 列表里面。算法 5.2给出了个人确认算法的伪代码，其中 FindBestAssignment 函数求解给定成本矩阵后的最优分配子问题，ApproximateHuman 函数对每个个体 $h \in L_t$，根据 $\mathcal{H}(h)$ 估计其新的状态信息，包括期望状态和置信度。确认后的结果，进一步通过重采样反馈到更新后的粒子集合中，使得粒子集合中的状态更加集中。

5.4　实验验证

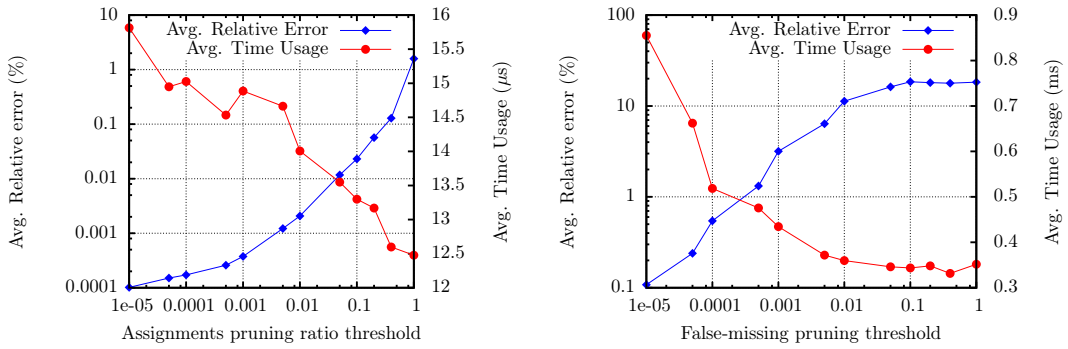
本节首先进行剪枝近似计算的误差测试，之后在 PETS2009 数据集里面比较了 PFS 和其他算法，最后在实际机器人平台 CoBot 上验证了 PFS 的有效性。

Input: Identities L_{t-1}, state pools $\mathcal{H}(h)$ for $h \in L_{t-1}$, particles $\mathcal{P}_t$, observation O_t, and maximal EM steps EM
Output: Identities L_t, and state pools $\mathcal{H}(h)$ for $h \in L_t$

Let $L_t \leftarrow \varnothing$, $L_{O_t} \leftarrow \varnothing$
foreach $o \in O_t$ **do**
    Let $\mathcal{H}_o \leftarrow \varnothing$
    Propose h_o as a potential new identity from o
    $L_{O_t} \leftarrow L_{O_t} \cup h_o$
    Let $\mathcal{H}(h_o) \leftarrow \varnothing$
end
Let $C_t \leftarrow L_{t-1} \cup L_{O_t}$
foreach $X \in \mathcal{P}_t$ **do**
    Let $\langle F^*, M^*, \psi^* \rangle = \varphi^* = \operatorname{argmax}_\varphi \Pr(O_t, \varphi \mid X)$
    $\mathcal{H}(\psi^*(s)) \leftarrow \mathcal{H}(\psi^*(s)) \cup s$ for each $s \in X$
end
foreach $h \in L_{t-1}$ **do**
    $\mathcal{H}(h) \leftarrow \mathcal{H}(h)$ taking account particle filtering from $\mathcal{P}_{t-1}$ to $\mathcal{P}_t$
end
foreach $s \in \mathcal{H}(\mathcal{P}_t)$ **do**
    if $\exists h \in C_t : s \in \mathcal{H}(h)$ **then**
        $l(s) \leftarrow h$
    end
end
Let $n \leftarrow 0$
repeat
    foreach $X \in \mathcal{P}_t$ **do**
        $n \leftarrow n + 1$
        Let $converged \leftarrow True$
        Let $c(s, h) \leftarrow -\log(f_h(s))$ for $s \in X$, $h \in C_t$
        Let $l^* \leftarrow$ FindBestAssignment(c)
        foreach $s \in X$ **do**
            if $l(s) \neq l^*(s)$ **then**
                $converged \leftarrow False$
                $l(s) \leftarrow l^*(s)$
            end
        end
        foreach $h \in C_t$ **do**
            $\mathcal{H}(h) = \{s \mid l(s) = h, s \in \mathcal{H}(\mathcal{P}_t)\}$
        end
    end
until $converged = True$ *or* $n > EM$

$L_t \leftarrow L_t \cup l(s)$ for $s \in \mathcal{H}(\mathcal{P}_t)$
$h \leftarrow$ ApproximateHuman($\mathcal{H}(h)$) for each $h \in L_t$
return $\langle L_t, \{\mathcal{H}(h) \mid h \in L_t\} \rangle$

算法 5.2: 个体确认（Human Identification）算法



(a) 分配剪枝　　(b) 误报—漏报剪枝

图 5.1　近似误差测试实验

表 5.1　近似误差测试实验详细实验结果（$T_a = 0.1$，且 $T_{fm} = 0.001$）

剪枝前	公式 5.2	公式 5.1
平均计算项数	32.66 ± 0.09	1466.52 ± 34.77
最大计算项数	5040	2.5018×10^6
剪枝前		
平均计算项数	2.11 ± 0.01	29.23 ± 0.13
最大计算项数	145	3043
剪枝率	93.50%	97.95%
相对误差	0.026%	3.30%

5.4.1　近似误差测试实验

在第一个实验里面，评估了剪枝策略带来的近似误差。通过仿真器，产生了一个 1000 周期的随机场景。该场景中，人按照参数为 $\lambda_s = 0.06/s$ 和 $\mu_s = 0.02$ 的泊松分布产生和消失；探测结果根据观察模型仿真得到，误报探测率为 $\nu_s = 0.5$，漏报探测率为 $\xi_s = 0.5$。当计算公式 5.2和公式 5.1时，假设没有剪枝的结果是 ν，有剪枝的结果是 ν'，则相对误差为 $|\frac{\nu-\nu'}{\nu}|$。令 T' 为为分配剪枝的概率比率阈值，T'' 为误报—漏报剪枝的概率阈值。为了评估分配剪枝，固定 T'' 为 0.001，在产生的随机场景上按照不同的 T' 值运行 PFS，并报告平均相对误差，和评价计算时间。实验中，分配问题大小小于 2×2 的平凡情况没有参与计算。图 5.1(a)展示了实验结果，从中可以看出，随着 T' 增大，平均相对误差几乎成比率增大，平均计算时间指数减少。然而，平均相对误差一直保持很小。即使 T' 取 1 的时候，也就是说之计算前 2 个分配方案，平均相对误差也不超过 2%。为了验证误报—漏报剪枝，固定 T' 的值为 0.1，在随机场景上按照不同的 T'' 值运行 PFS 算法。实验结果见图 5.1(b)。图中，可以看到一个类似的趋势。然而 T'' 对最终结果的影响相对较大。表 5.1展示了更详细的实验结果。

表 5.2　PFS 算法实现参数

	Parameter	PETS2009	Real Robot
λ	人的产生速率 (1/s)	0.0	0.0
μ	人的消失速率 (1/s)	0.02	0.01
σ_p	加速度大小方差 (m^2/s)	1.0	0.5
ν	误报探测率 (1/s)	6.0	1.0
ξ	漏报探测率 (1/s)	2.0	1.0
τ	更新时间间隔 (s)	0.14	0.1
T'	分配剪枝阈值	0.1	0.1
T''	误报—漏报剪枝阈值	0.001	0.001
$\mathbf{\Sigma}$	观察协方差	$0.5\mathbf{I}$	$0.3\mathbf{I}$
α_0	初始 Gamma 分布 α 参数	2.0	2.0
β_0	初始 Gamma 分布 β 参数	1.0	1.0
A'	最小限位框面积 (m^2)	0.5	0.5
A''	最大限位框面积 (m^2)	2.5	2.5
R	报告个体的最小置信度	0.4	0.0
N	粒子总数	128	256
H	最大 EM 步数	10	10

5.4.2　标准测试数据集实验

为了跟现有多对象跟踪算法进行比较，本节在 PETS2009[47] 数据集的 S2L1 系列数据上评估了 PFS 算法。该数据是从一个较高位置的摄像头拍摄获得的，大概 7fps，包含 795 帧，每帧显示最多 8 个实际的人，和 13 个探测结果。本节主要在按照文献[171] 切割后的数据上做评估（大约覆盖 $19.0 \times 15.8 m^2$ 面积的区域），但也报告了完整数据上的实验结果。切割后的数据包含最多 11 个观察，平均每帧 5.67 个探测结果。每一个探测结果包含一个置信度，和一个*限位框*（Bounding Box）。限位框包含了图像坐标系中的中心位置、高度和宽度数据。该数据集包含了摄像头校准数据，所以可以在假设图像中一点在地面上的情况下，计算出该点在世界坐标系中的位置。* 实验中，具有特别大或小的限位框的探测结果被认为置信度为 0。最终 PFS 报告的个体中只有置信度大于 0.4 的参与了最后的评估。表 5.2列出了实验用到的具体算法参数。

本节主要使用 CLEAR MOT 指标[177] 评估 PFS 算法的性能。实验室采用了跟目前文献里面一致的 1 米距离阈值。*多对象跟踪准确度*（Multiple Object Tracking Accuracy，MOTA）指标反应了算法跟踪的准确度，综合考虑了*误报*（False Positive）、*漏报*（False Negative）和 *ID 交换*（Identity Switch）等错误情况。*多对象跟踪精度*（Multiple Object Tracking Precision，MOTP）是所有真实目标和估计目标之间所有时间内的平均距离 d 折算后的一个百分比：$(1-d) \times 100\%$。具体地，令 n_t 为周期 t 时真实联合状态和算法估计的联合状态之间通过求解一

*由文献[172] 收集的完整数据见：http://www.milanton.de/data.html。

表 5.3　PETS2009 S2L1 数据集量化实验结果

算法	MOTA	MOTP	IDS	MT	FM
PFS[1] **(本章提出)**	93.1%	76.1%	3.6	18.0	16.0
PFS[1,2] **(本章提出)**	90.6%	74.5%	4.8	17.6	20.4
Milan[172]	90.6%	80.2%	11	21	6
Milan et al.[173]	90.3%	74.3%	22	18	15
Segal et al.[174]	92%	75%	4	18	18
Segal et al.[174][2]	90%	75%	6	17	21
Zamir et al.[2][175]	90.3%	69.0%	8	-	-
Andriyenko et al.[171]	81.4%	76.1%	15	19	21
Breitenstein et al.[176][2]	56.3%	79.7%	-	-	-

[1] 16 次运行的平均值
[2] 整体范围内的评估结果

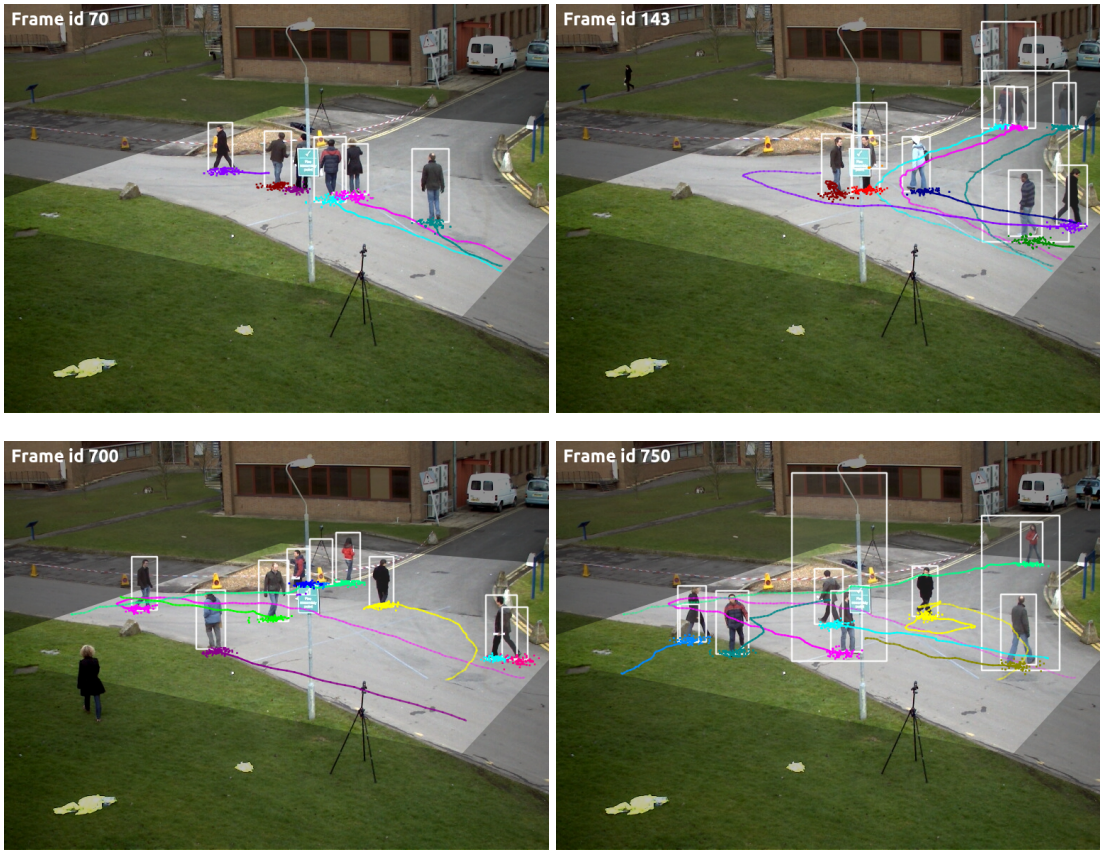
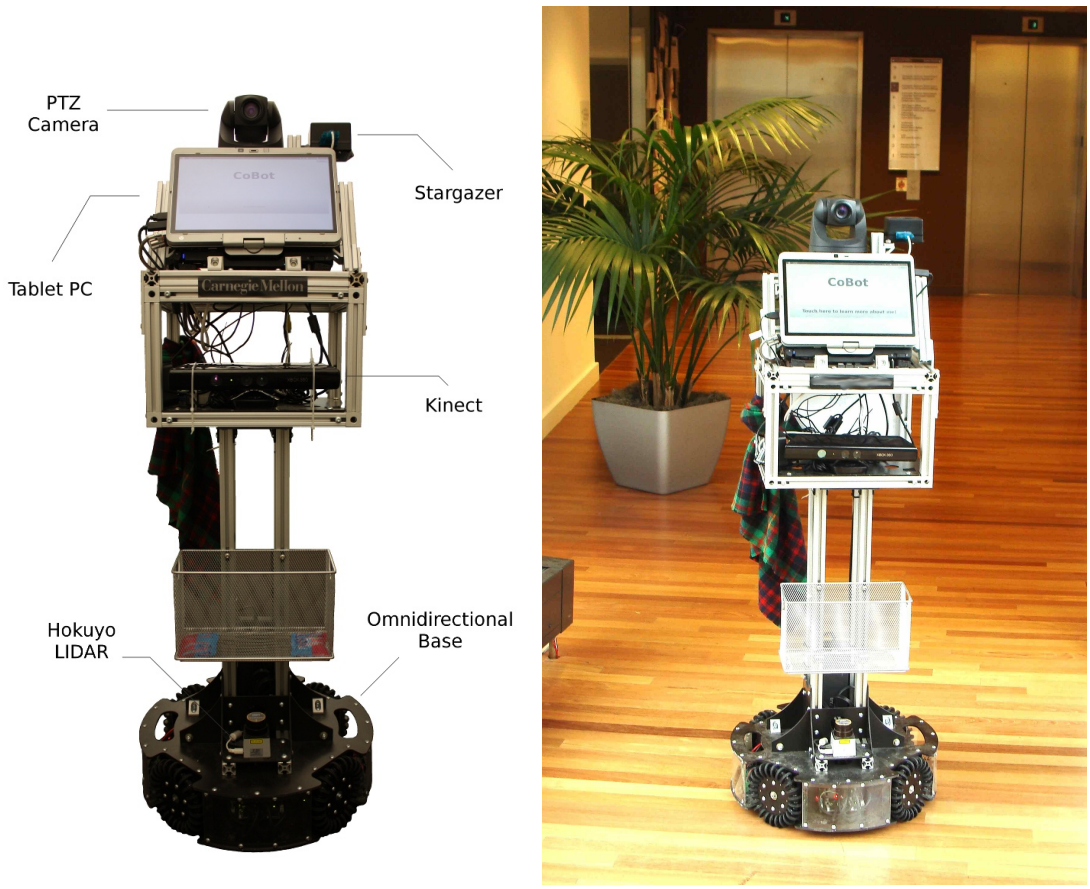


图 5.2　PFS 算法在 PETS2009 S2L1 数据集中的跟踪结果举例

个最优分配问题找到的匹配数目（如果某个真实状态和估计状态之间的距离小于 1 米，就认为这两个状态匹配），$d_t^{(i)}$ 为真实的状态和与之匹配的估计状态之间的距离，则 MOTP 计算如下：

$$\mathrm{MOTP} = \left(1 - \frac{\sum_t \sum_{1 \leqslant i \leqslant n_t} d_t^{(i)}}{\sum_t n_t}\right) \times 100\%. \tag{5.3}$$



(a) CoBot 硬件构造　　(b) 部署中的 CoBot 服务机器人

图 5.3　CoBot 实验平台（图片来自 CORAL 研究组）

令 g_t 为实际目标的真实数目，a_t 为算法报告的估计目标的数目，m_t 为 ID 交换的错误数目，MOTA 定义为：

$$\text{MOTA} = \left(1 - \frac{\sum_t (g_t + a_t - 2n_t + m_t)}{\sum_t g_t}\right) \times 100\%. \tag{5.4}$$

MOTP 反应了算法精确估计目标状态的能力，MOTA 反应了算法成功跟踪和保持目标轨迹的能力。

另外，本节也报告了文献[178] 提出的一些度量指标，包括几乎全部跟踪（Mostly Tracked，MT）数目、跟踪碎片（Track Fragmentation，FM）数目和 ID 交换数目。一个目标如果在其 80% 的时间内都被成功跟踪就称其为几乎全部跟踪。跟踪碎片数目即一个目标的真实轨迹在“被跟踪”和“没有被跟踪”之间切换的次数。表 5.3展示了主要实验结果，图 5.2显示了一些跟踪的例子。图中，白色的限位框为原始的探测结果；估计出来的目标轨迹和当前粒子状态使用不同的颜色显示，意为其属于不同的个体。*

作为比较，文献[176] 使用贪心的数据关联方法在粒子滤波框架内单独跟踪每一个目标，可以被看成是 PFS 的一个很好的基准（Baseline）方法。文献[174] 把多

*显示整个实验结果的完整视频见：http://goo.gl/4QIley。

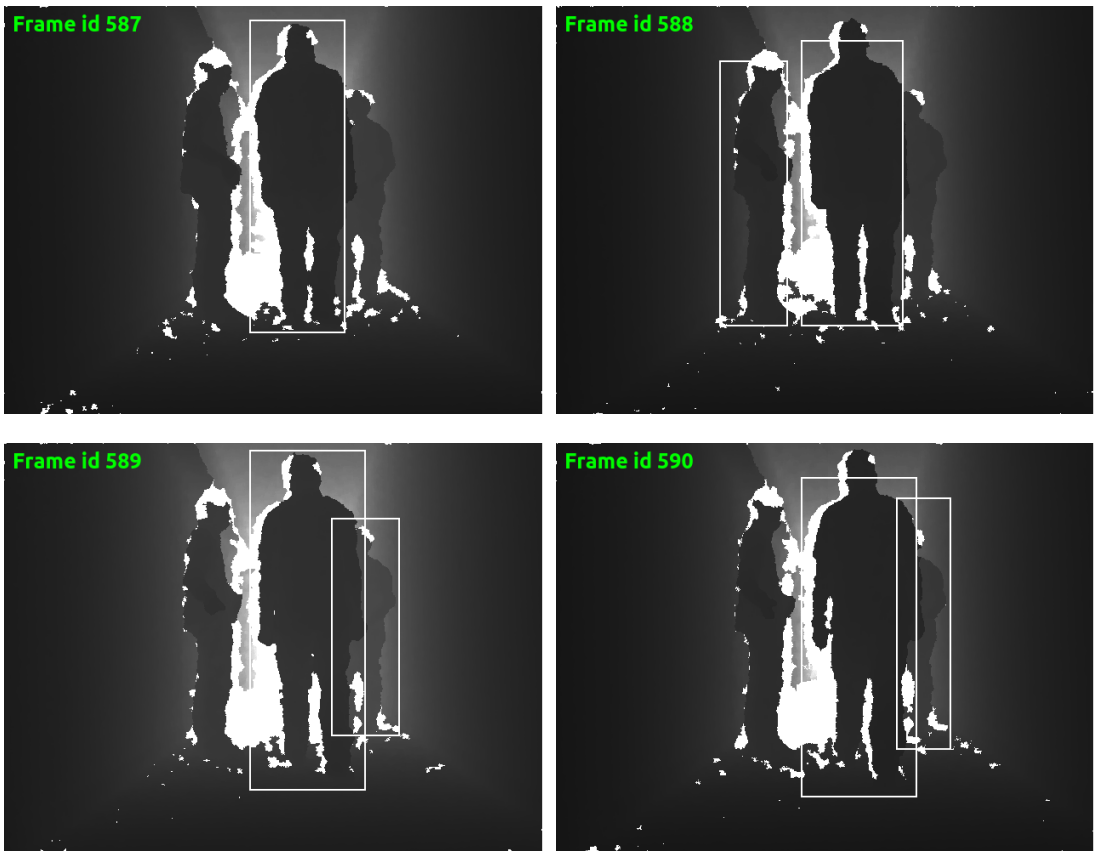


图 5.4　CoBot 实验连续原始探测数据 I

对象跟踪问题建模成一个*切换线性动态系统*（Switch Linear Dynamical System），并利用了一个目标问题里面训练得到的行人—*离群值*（Outlier）探测器。文献[175] 通过结合运动模型和外貌信息利用*通用最小集团图*（Generalized Minimum Clique Graph）求解全局数据联合问题。文献[171]、文献[172] 和文献[173] 把多对象问题建模成给定*能量函数*（Energy Function）和初始轨迹后*样条曲线*（Spline）空间上的离线优化问题，其中初始轨迹由使用贪心数据联合的若干独立*扩展卡尔曼滤波器*（Extended Kalman Filter，EKF）获得。实验结果显示，PFS 算法比目前领域内先进算法效果更好，并且 PFS 没有使用任何数据相关的信息（比如外貌信息），同时具有可以按照贝叶斯递归方式在线运行的能力。

5.4.3　真实机器人演示

本节通过把 CoBot 部署到一个大学建筑里面供用户使用来验证 PFS 算法在实际机器人上的有效性。下面首先介绍 CoBot 实验平台。

5.4.3.1　CoBot 实验平台

CoBot 是由美国卡耐基梅隆大学（Carnegie Mellon University）Manuela M. Veloso 教授的 CORAL（为 Cooperate, Observe, Reason, Act, and Learn 的缩写）

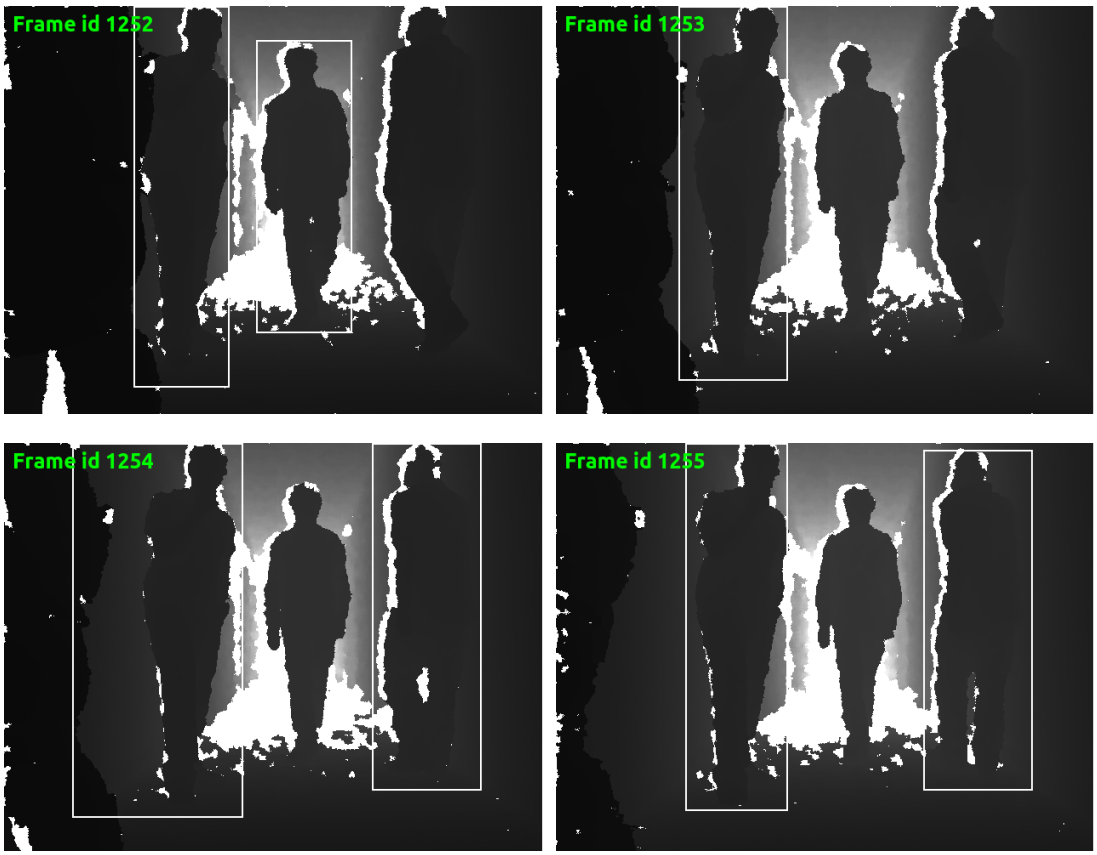


图 5.5　CoBot 实验连续原始探测数据 II

研究组研制的一系列智能自主移动机器人[149, 150]。目前已经有四个版本，包括 CoBot-1，CoBot-2，CoBot-3 和 CoBot-4。其中，CoBot-2 的照片见图 5.3(a)。CoBot 机器人拥有搭载 4 个万向轮的移动底盘，其他设备还包括：角度可控的 PTZ 摄像头，Microsoft Kinect 深度摄像头，Hokuyo 激光传感器，触屏平板电脑，麦克风以及扬声器。CoBot 使用深度摄像头和激光传感器在建筑走廊内，自主定位、导航和避障，并且可以在人的帮助下自主搭乘电梯。几个 CoBot 版本的区别主要在硬件上，比如 CoBot-4 只装备了 Kinect 传感器，而没有装备激光传感器。

5.4.3.2　CoBot 机器人演示实验

本章主要采用 CoBot-2 作为演示实验平台。实验中，把 CoBot-2 部署到一幢建筑内，见图 5.3(b)。用户可以通过触屏给机器人下达任务指令，机器人能完成的任务包括：1. 运动到某一房间；2. 到某一房间传送语音信息；3. 在房间中递送物品；4. 护送用户到电梯或某一房间。CoBot-2 在完成用户任务的同时，使用 PFS 算法自动跟踪随机遇到的人，并把结果提供给避障模块使用。CoBot-2 使用 Kinect 作为主要的人像探测传感器，在导航的时候同时试图跟踪遇到的随机人类目标。频率为 30Hz、分辨率为 640 × 480 的深度图像被用作人体探测的输入。一个结合了基于图的分段方法的面向深度的柱状图（Histogram of Oriented

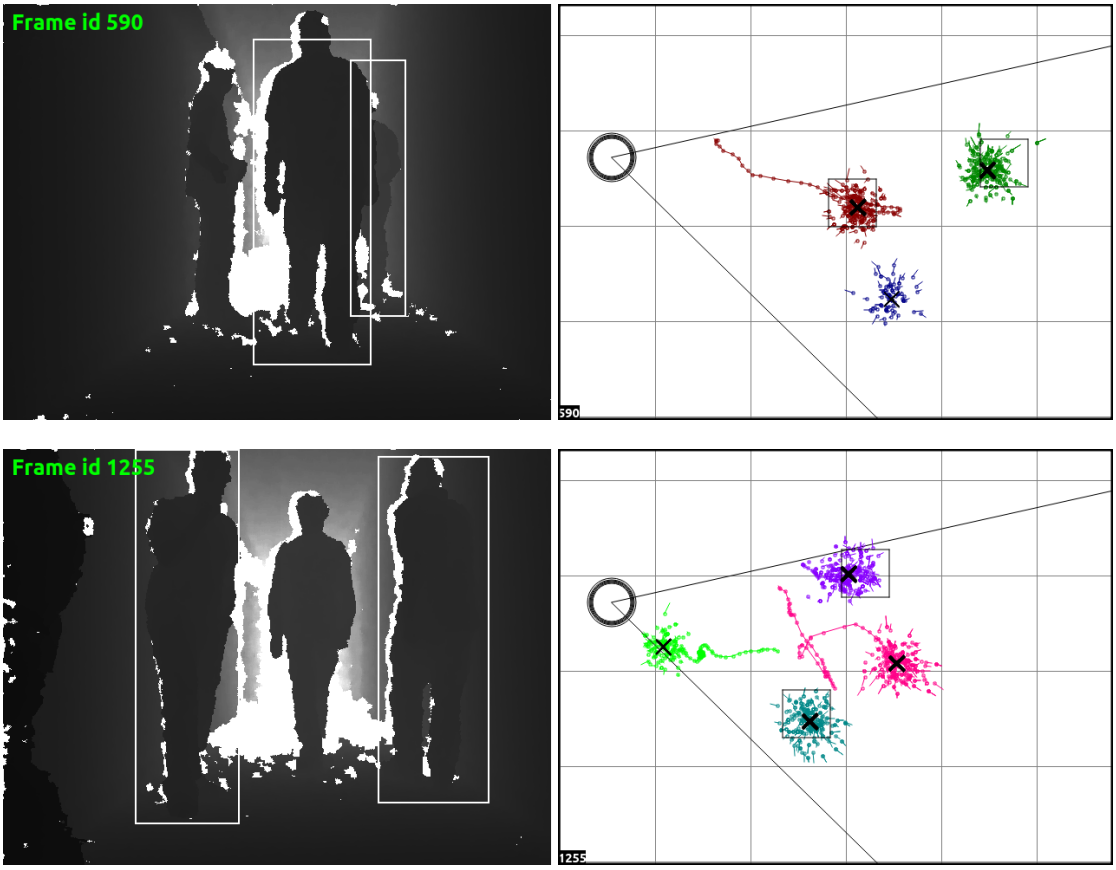


图 5.6　PFS 在真实机器人（CoBot）上的实验结果

Depth，HOD）算法独立处理每一帧深度图像，并以 10Hz 的频率报告探测到的人体图像[179]。机器人搭载一个 2.7GHz 四核 CPU、4GB 内存 Linux 3.5 笔记本电脑。PFS 算法实现的具体参数见 表 5.2。

图 5.4和图 5.5展示了连续探测结果的两个例子。图中，每个小图显示了标注原始探测结果（限位框）的深度图像。图 5.6显示对应的最后一帧跟踪结果。图中，左边栏是深度图像；右边栏显示了相应的世界坐标系中的跟踪结果：圆形代表机器人；两条射线构成的区域代表机器人的视角范围；图中的 X 形状代表估计个体的期望位置；不同个体的估计轨迹和粒子中的状态显示为不同的颜色；原始探测结果用正方形表示。图中格子大小为 1m×1m。值得指出的是，图中没有参数探测结果的人依然被成功跟踪了。*

5.5　本章小结

本章主要介绍了一个新颖的面向人—机器人交互和多对象跟踪问题的集合粒子滤波算法（PFS）。该算法不需要事先知道有多少人，并且可以处理不可避免的误报和漏报探测。从多对象跟踪的角度来说，PFS 算法通过使用基于集合的形式化方法，并且自动推理联合空间中的后验状态分布和数据关联，避免了

*显示原始探测结果和跟踪结果的完整视频见：http://goo.gl/06T77a。

直接进行观察到目标的数据关联。PFS 算法整体上在 PETS2009 数据集上比目前领域先进的算法效果更好。真实机器人 CoBot 上的测试表明使用该算法的机器人可以在真实环境中实时识别和跟踪遇到的随机目标。下一步工作，计划在更多人的环境，和更复杂的人机交互任务中进一步测试该算法。

第六章　总结和展望

本章主要总结本文的主要贡献，并讨论未来的研究前景。

6.1　工作总结

随着人工智能研究和应用技术的快速发展，智能系统已经渗透到人们生活的各个方面，并逐渐发挥着越来越重要的作用。智能体（Agent）理论为智能系统的建模、设计和实现提供了统一的框架。智能体必须能够自主地综合各种传感器提供的感知信息，做出实时决策，并与环境进行交互。一般来讲，传感器收集到的信息是不全面的，并带有各种噪音，同时执行机构的执行效果也具有不可预知性，并带有误差。诸如此类的不确定性为智能体的规划和感知任务带来了很大的挑战。基于马尔科夫过程的决策理论为这类问题提供了基本的理论框架，根据具体问题的不同一般细分为处理完全可观察问题的马尔科夫决策过程（Markov Decision Processes，MDP）和处理部分可观察问题的部分可观察马尔科夫决策过程（Partially Observable Markov Decision Processes，POMDP）。MDP 中，智能体通过跟环境进行交互，根据行动值函数选择每一步的动作，以最大化期望累积回报（或最小化期望累积成本）。MDP 假设智能体可以完全感知环境的状态，不确定性主要来自环境的状态转移。POMDP 将 MDP 模型扩展到部分可观察环境，引入状态空间上的概率分布作为信念状态，进而转化为连续空间上的 MDP 问题。不确定环境下的感知问题实际上就是 POMDP 模型下的信念更新问题。从数学的角度来看，MDP 和 POMDP 统一建模了不确定环境下的感知和规划问题。完整求解 MDP 和 POMDP 问题是非常困难的，实际应用时，往往需要采用各种在线规划、分层规划、蒙特卡洛仿真和近似计算的技术。本文以马尔科夫决策理论为主要理论基础，以大规模不确定环境下的规划和感知问题为主要研究内容，主要贡献有：

1. 基于 MAXQ 分层分解的 MDP 在线规划算法。经过多年的发展，学术界已经提出了很多高效的 MDP 求解算法，可以处理规模不断变大的问题。但特别巨大规模问题的求解，仍然有很大的挑战性。举例来说，离线算法能精确找到这些问题的最优解，但需要遍历整个状态空间或策略空间。但由于维度诅咒问题（即问题的状态数随着状态变量数呈指数式增长），离线算法在很多实际问题中并不可用。一个研究热点是利用在线算法，在线算法并不需要预先遍历整个状态空间或策略空间，而是在智能体跟环境交互过程中实时计算当期状态下的最优动作，从而可以解决规模较大的问题。其主要的理论依据在于每次执行策略的过程中，智能体实际遇到的情况只是各种可能中很小的一部分，因此在大规模问题求解上，在线算法更具有优势。但在线算法本身也有需要解决的难题。因为智能体需要实时的

对环境做出反应，因此每次可用于规划的时间非常的有限。本文设计出高效的分层在线规划算法 MAXQ-OP。MAXQ-OP 利用原始问题本身所具有的分层结构，根据 MAXQ 值函数分解理论把原始问题分解成一系列的小问题，在线规划过程中同时求解这些小问题，最终完成原始问题的求解。MAXQ-OP 得益于分层结构上的时序抽象、状态抽象和子任务共享，比直接求解原始问题高效。创新点是通过提出终止概率和完成函数的近似计算方法，把分层理论应用到在线规划算法中，从而可以求解比现有文献范围内规模更大的问题。本文主要研究平台之一的 RoboCup 仿真 2D 机器人足球是典型的多智能体合作与对抗问题，主要挑战包括：完全分布式、局部观察、受限通讯、多智能体系统、动作不确定性、观察不确定性、和问题规模巨大。自 2009 年，以 MAXQ-OP 为主要决策框架的算法成功应用到科大“蓝鹰”RoboCup 仿真 2D 机器人足球队中，为球队自 2005 年起至今（2014 年）连续多年在 RoboCup 2D 世界杯比赛中保持前两名，并获得五个世界冠军（2006，2009，2011，2013 和 2014）做出了重要贡献。

2. 基于后验动作采样的蒙特卡洛在线规划算法。对于大规模不确定性问题，完整的状态转移函数往往是无法显式给出的。但问题的仿真器则比较容易得到，蒙特卡洛树搜索为解决此类方法提供了有用的框架。其基本思想是在状态空间里面通过蒙特卡洛仿真进行前向搜索，在末端节点运行基本的 Rollout 策略，并使用仿真运行的结果更新搜索树上的统计量。蒙特卡洛树搜索是 Anytime 算法，其迭代过程中逐步获得更优的策略。蒙特卡洛树搜索的一个基本问题是利用—探索的平衡。使用 UCB 启发值的算法取得了不错的效果，但关注于累积剩余值优化的 UCB 策略其实并不适用于迭代式的在线规划算法。另一方面，UCB 方法的探索因子也没有系统的方法可以确定。Thompson 采样在 MAB 问题就累积剩余值而言可以达到渐进最优，同时累积剩余值和简单剩余值的实验性能都比同类算法更好。针对 MDP 问题，本文使用正态分布的混合分布建模蒙特卡洛搜索树上的长期回报的未知分布，使用 Dirichlet-NormalGamma 混合分布作为其共轭分布，根据贝叶斯推理理论对其进行后验更新，设计出基于 Thompson 采样蒙特卡洛在线 MDP 规划算法 DNG-MCTS，证明其收敛性，并在 MDP 标准测试问题上取得了比本领域内先进算法（UCT 相关算法）更好的实验结果。创新点是根据马尔科夫链上的中心极限定理，推导出来蒙特卡洛搜索树上的长期回报的近似分布，从而可以使用贝叶斯方法对其进行建模和推理，通过使用 Thompson 采样方法在多个 MDP 标准测试问题上取得了较好的实验结果。进一步，将提出的 DNG-MCTS 算法推广到 POMDP 问题，使用 Dirichlet-Dirichlet-NormalGamma 混合分布作为长期回报分布的共轭分布，使用贝叶斯方法对其进行建模和更新，设计出基于 Thompson 采用的蒙特卡洛在线 POMDP 规划算法 D²NG-POMCP，在 POMDP 标准测试问题上取

得了比本领域内先进算法（POMCP 算法）更好的实验结果。创新点是提出把环境状态和智能体的信念状态看成一个联合状态，从而可以把给定策略的 POMDP 问题转化成一个马尔科夫链，在这个马尔科夫链上根据中心极限定理，推导出蒙特卡洛搜索树上长期回报分布的近似分布，从而可以使用贝叶斯方法对其进行建模和推理，使用 Thompson 采样在每个决策节点选择动作，在多个 POMDP 标准测试问题上取得了相比 POMCP 更好的实验结果。

3. 基于集合粒子滤波的多对象跟踪算法。在复杂的动态社会化环境中，机器人能够自主实时识别和跟踪潜在的多人状态的能力对人—机器人的成功交互至关重要。面临的主要挑战有：底层的机器视觉算法不可能具有 100% 识别准确率，总是会有不可避免的各种误报和漏报的探测结果；人和机器人都处在不停的运动过程中，会有各种产生遮挡的情况；同时，通常情况下，探测结果是匿名的，即无法直接从探测结果知道该探测结果来自于哪个潜在目标；机器人的计算资源有限，必须做出实时的状态估计。从数学的角度考虑，在线多对象跟踪问题等价于一个复杂不确定性 POMDP 环境下的信念更新问题，一般可以建模成隐马尔科夫模型的滤波问题。很多传统的在线多对象跟踪算法，会假设一个或多个数据关联，并使用贝叶斯滤波方法单独更新每一个潜在状态。这类方法的一个共同问题是，某次假设的数据关联出错以后，估计的多目标状态更新也就出错了，并且以后很难恢复。本文提出的集合粒子滤波方法（PFS）使用集合来表示联合状态和联合观察，在集合的定义下计算完整的观察函数。该观察函数的计算隐含了所有数据关联的可能性。最后使用粒子滤波在联合空间中根据序列化的观察自动推理出后验状态分布，同时隐含了数据关联的概率推理。最后，基于期望—最大化的个体确认过程从更新后的粒子集合中辨认和报告每一个潜在的个人供高层任务使用。从多对象跟踪的角度来看，PFS 方法避免了在更新估计状态前直接进行观察到目标的数据关联，相比于传统算法，在复杂的不确定性环境中可以保持更加稳定。PFS 算法在非常具有挑战性的 PETS2009 数据集中，在 CLEAR MOT 指标上，跟很多领域内先进算法相比，取得了更好的实验效果。同时，实际机器人平台 CoBot 上的实验结果表明，PFS 可以实时识别和跟踪室内环境中遇到的随机目标，为进一步成功的人—机器人交互提供了可能。

综上所述，本文研究的重点是按照原理性的方式赋予智能体在不确定性环境自动规划和感知的能力。本文工作主要参考马尔科夫过程相关决策论规划框架展开。马尔科夫决策理论按照贝叶斯最优的方式建模智能体在不确定性环境下的规划和感知问题：不确定环境下的规划问题可以用 MDP 或 POMDP 分别在完全可观察和部分可观察环境下建模；不确定环境下的感知问题可以用 POMDP

建模成部分可观察环境下的信念更新问题。特别地，本文提出基于 MAXQ 分层分解的在线规划算法——MAXQ-OP，以解决大规模 MDP 问题在线规划的计算时间瓶颈问题；本文在 MDP 和 POMDP 在线蒙特卡洛规划算法里面应用 Thompson 采样的方法，提出 DNG-MCTS 和 D²NG-POMCP 算法，在在线搜索过程中，根据一个动作的成为最优动作的后验概率来随机选择这个动作，以尝试搜索树上的累积剩余值和简单剩余值；本文在人—机器人交互领域，针对多对象跟踪问题，利用了一个基于集合形式化的联合粒子滤波算法——PFS，以避免直接进行观察到目标的数据关联带来的算法不够稳定的问题。这三个主要工作彼此独立，又紧密相联，分别从不同的角度研究和探讨了不确定性环境下的规划和感知问题：MAXQ-OP 算法关注于大规模 MDP 问题的分层在线规划；DNG-MCTS 和 D²NG-POMCP 算法分别关注于没有完整显式模型的 MDP 和 POMDP 问题的蒙特卡洛在线规划问题；PFS 算法关注于复杂 POMDP 问题的信念更新问题。本文工作的有机结合为大规模不确定环境下的规划和感知问题带来了原理性的解决方案，为进一步研究和应用提供了重要的基础，具有很强的处理和解决大规模不确定性问题的研究和应用价值。

6.2　前景展望

本文希望未来的工作可以更多的涉及集成了感知、规划和学习能力的人工智能系统的原理性解决方案，以处理越来越复杂的大规模不确定性动态和/或随机环境下的研究和应用问题。特别地，包括以下几个方面：

1. 分层结构的自动化获取。MAXQ-OP 算法利用问题本身的分层结构在线求解大规模 MDP 问题。标准测试实验结果表明 MAXQ-OP 可以很高效地找到问题的最优解；RoboCup 2D 中的成功应用表明 MAXQ-OP 具有巨大的处理大规模现实问题的潜力。应用 MAXQ-OP 算法的前提是提供目标问题的 MAXQ 分层结构。这个前提是不平凡的，常常需要用到问题本身的先验知识。下一步工作计划通过目标问题的先验数据（如状态—行动转移历史）使用机器学习的方法自动获取完整的分层结构和终止分布的近似计算。就自动发现分层结构而言，一个可能的技术方案是通过历史数据自动构建表达状态转移函数的动态贝叶斯网络。MDP 问题的动态贝叶斯网络可以很好地简化问题，揭示状态变量之间的内部结构，并且具有很好的范化特性：即通过局部数据学习得到的动态贝叶斯网络在一般情况下同样适用于状态空间中的其他部分。分析学习到的贝叶斯动态网络的结构[180]，整理状态变量之间的因果关系，为自动创建宏状态提供了原理性的方法。进一步，通过分析不同层次宏状态之间的转移策略就完成了 MAXQ 分层结构的自动发现。

2. 交互式任务规划和运动规划。机器人的任务规划和运动规划为机器人的利

用和部署提供了自然的分层解决方案。任务规划给出完成某项具体任务所必需的基本步骤。这些步骤往往表现为若干高层宏动作的序列，而忽略具体问题的参数和几何限制。运动规划部分复杂实际完成任务规划的结果。任务规划必需考虑每一步原子行动的几何限制。实际情况下，一方面，忽略问题具体限制的任务规划模块规划出来基本步骤，并期望运动规划模块改进和完成高层规划的结果，往往是不可行的，因为高层规划出来的结果经常不能进行有效的进行运动规划；另一方面，任务规划时有很多隐藏信息，必需要等到实际执行运动规划时才能获得。这说明，将任务规划和运动规划在时间上割裂开来是不可行的。本文希望能提出一个统一的迭代式处理任务规划和运动规划的在线规划框架。跟 MAXQ-OP 算法类似，该框架中任务规划模块把底层运动规划模块看成是宏动作，任务规划的时候自动在分层任务结构和状态空间中进行搜索，搜索的结果给出实际应该执行的原子动作。机器人执行该原子动作，观察动作执行结果，更新世界状态，并进入下一个决策周期，再次进行分层规划。该在线规划和执行框架可以自动隐式处理规划和重规划，每一步即是实时规划，也是自动重规划。任务规划总是在最新的信息上进行的，并且由于底层运动规划一起参与了高层的任务规划，所以任务规划的结果可以保证是可以执行的。

3. 贝叶斯最优的蒙特卡洛规划算法。蒙特卡洛规划的基本问题是利用—探索的平衡，即智能体不仅需要充分选择当前最好的动作，同时也需要尝试没有实验过的动作。现有的方法，比如 UCB 启发函数或本文提出的后验动作采样方法（Thompson 采样）的主要思想都是基于累积剩余值或简单剩余值的优化。但单纯的累积剩余值或简单剩余值并不完全适合蒙特卡洛规划算法：蒙特卡洛规划算法实际关注的不是搜索过程中的累积或简单剩余值，而是搜索结束以后实际选择动作的最优性。使用贝叶斯方法可以给出利用—探索的最优平衡策略，即把搜索过程中的每一步仿真看成是数据收集的过程，观察仿真结果，更新模型的后验分布，并根据当前的模型分布选择下一个最佳动作，达到贝叶斯设定下的最优搜索。给定问题的模拟器，和最多的计算资源预算（比如最多采样的次数，或最多的计算时间），可以将这个问题转化成一个有限规划时限的 POMDP 问题。求解该 POMDP 就可以得到最优的搜索策略。当然完全求解这个连续模型分布空间上的 POMDP 是不可能的，所以有必要在这个特殊的 POMDP 问题中发现其特有的内部结构，并利用发现的结构利用出高效的启发式求解策略。

4. 多人跟踪和意图识别算法。成功完成社会化人—机器人交互任务，不仅需要实时识别和跟踪潜在多人的状态，还需要理解每个人的潜在行动意图。这里的行动意图可以理解为人的运动模式，比如停止，还是运动，或者更复杂也更微妙的意图，比如运动中的人是计划跟机器人交互，还是计划避

开机器人。诸如此类的意图识别对机器人跟人的成功交互是至关重要的。一个可行的方案是引入人的意图作为隐藏的状态变量之一，为不同的意图构建不同的运动模型，在 PFS 框架里面自动推理人的意图，或建立意图上的混合分布，并报告每个个体的支配意图供高层任务决策使用。另一方面，成功识别意图，获得人的实际运动模型（而不仅仅是通常的随机行走或加速模型），以进一步可以提高多人状态跟踪的精度和准确度。

5. 信念空间中的统一规划和感知框架。智能体在部分可观察环境中成功决策的关键是感知当前状态。引入信念状态，并按照贝叶斯推理的方式进行信念更新为感知问题提供了原理性的解决方案。信念空间上的规划又进一步同意了规划和感知任务。POMDP 已经为这一问题提供了理论基础，但在实际环境中应用的时候，往往由于建模后的 POMDP 过于复杂而很难求解。本文希望可以提出具有保证的近似方法，在信念空间上，统一并求解复杂环境中的感知和规划问题。一个思路是构造实际问题的模拟器，近似表达观察函数，使用粒子滤波的方法描述信念状态，并使用蒙特卡洛方法进行实时规划。问题的完整模拟器往往很难构造，不过给出一定精度的近似模拟器也就可以了，模拟器可以表达成一系列动态贝叶斯网络。网络上的采样规则给出了状态转移的产生式模型。最终提出的方法可以使用蒙特卡洛仿真的方式动态求解规划和感知问题，并且其上也可以引入诸如分层、启发式搜索等近似计算技术，以保证实时性。

参考文献

[1] Stuart Russell, Peter Norvig. Artificial Intelligence: A modern approach, volume 25. Prentice-Hall, Egnlewood Cliffs, 1995.

[2] Leslie Pack Kaelbling, Tomás Lozano-Pérez. Integrated task and motion planning in belief space. The International Journal of Robotics Research, 2013. 0278364913484072.

[3] 李响, 陈小平. 一种动态不确定性环境中的持续规划系统. 计算机学报, 2005, 28(7):1163–1170.

[4] N Xiong, P Svensson. Multi-sensor management for information fusion: issues and approaches. Information Fusion, 2002, 3(2):163–186.

[5] 王醒策, 张汝波, 顾国昌. 基于势场栅格法的机器人全局路径规划. 哈尔滨工程大学学报, 2003, 24(2):170–174.

[6] Sebastian Thrun, Wolfram Burgard, Dieter Fox. Probabilistic robotics. MIT press, 2005.

[7] 陈华平, 计永昶, 陈国良. 分布式动态负载平衡调度的一个通用模型. 软件学报, 1998, 9(1):25–29.

[8] Zhaohong Jia, Huaping Chen, Jun Tang. An improved particle swarm optimization for multi-objective flexible job-shop scheduling problem. Proceedings of Grey Systems and Intelligent Services, 2007. GSIS 2007. IEEE International Conference on. IEEE, 2007. 1587–1592.

[9] Shengchao Zhou, Huaping Chen, Rui Xu, Xueping Li. Minimising makespan on a single batch processing machine with dynamic job arrivals and non-identical job sizes. International Journal of Production Research, 2014, 52(8):2258–2274.

[10] Malik Ghallab, Dana Nau, Paolo Traverso. Automated planning: theory & practice. Elsevier, 2004.

[11] 范永, 谭民. 机器人控制器的现状及展望. 机器人, 1999, 21(1):75–80.

[12] Michael Duff. Design for an optimal probe. Proceedings of ICML, 2003. 131–138.

[13] Leslie Pack Kaelbling, Michael L Littman, Andrew W Moore. Reinforcement learning: a survey. Journal of Artificial Intelligence Research, 1996, 4:237–285.

[14] RS Sutton, AG Barto. Reinforcement learning: An introduction, volume 116. Cambridge Univ Press, 1998.

[15] 张汝波, 顾国昌. 强化学习理论, 算法及应用. 控制理论与应用, 2000, 17(5):637–642.

[16] 王醒策, 张汝波, 顾国昌, et al. 多机器人动态编队的强化学习算法研究. 计算机研究与发展, 2003, 40(10):1444–1450.

[17] 张汝波, 周宁. 基于强化学习的智能机器人避碰方法研究. 机器人, 1999, 21(3):204–209.

[18] 高阳, 陈世福, 陆鑫. 强化学习研究综述. 自动化学报, 2004, 30(1):86–100.

[19] 李春贵, 刘永信. 一种有限时段 Markov 决策过程的强化学习算法. 广西工学院学报, 2003, 14(1):1–4.

[20] 孟伟, 洪炳熔, et al. 强化学习在机器人足球比赛中的应用. 计算机应用研究, 2002, 19(6):79–81.

[21] 高阳, 周志华. 基于 Markov 对策的多 Agent 强化学习模型及算法研究. 计算机研究与发展, 2000, 37(3):257–263.

[22] 蔡庆生, 张波. 一种基于 Agent 团队的强化学习模型与应用研究. 计算机研究与发展, 2000, 37(9):1087–1093.

[23] Martin L Puterman. Markov Decision Processes: Discrete Stochastic Dynamic Programming. John Wiley & Sons, Inc., 1994: 672.

[24] 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.

[25] Sheldon M Ross. Stochastic processes, volume 2. John Wiley & Sons New York, 1996.

[26] Markov, Andrei Andreevich. The theory of algorithms. Trudy Matematicheskogo Instituta im. VA Steklova, 1954, 42:3–375.

[27] Leonard E Baum, Ted Petrie. Statistical inference for probabilistic functions of finite state Markov chains. The annals of mathematical statistics, 1966. 1554–1563.

[28] 葛愿, 丛爽, 尚伟伟. 基于离散隐马尔可夫模型的网络化控制系统最优控制器设计. 中国科学技术大学学报, 2012, 42(002):161–169.

[29] 丛爽, 杨洁, 楼越升. 基于马尔科夫决策过程的量子系统状态布局数转移最短路径的决策. 2007 中国控制与决策学术年会论文集, 2007..

[30] Eugene A Feinberg, Adam Shwartz, Eitan Altman. Handbook of Markov decision processes: methods and applications. Kluwer Academic Publishers Boston, MA, 2002.

[31] 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.
[32] Michael Wooldridge. An introduction to multiagent systems. John Wiley & Sons, 2009.
[33] 谭民, 王硕, 曹志强. 多机器人系统. 清华大学出版社有限公司, 2005.
[34] P Gmytrasiewicz, P Doshi. A framework for sequential planning in multiagent settings. Journal of Artificial Intelligence Research, 2005, 24(1):49–79.
[35] Daniel S Bernstein, Robert Givan, Neil Immerman, Shlomo Zilberstein. The complexity of decentralized control of Markov decision processes. Mathematics of operations research, 2002, 27(4):819–840.
[36] Feng Wu. Decision-Theoretic Planning for Multi-Agent Systems[D]. Hefei, Anhui, China: University of Science and Technology of China, 2011.
[37] Thomas G Dietterich. Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition. Journal of Machine Learning Research, 1999, 13(1):63.
[38] **Aijun Bai**, Xiaoping Chen, Patrick MacAlpine, Daniel Urieli, Samuel Barrett, Peter Stone. WrightEagle and UT Austin Villa: RoboCup 2011 simulation league champions. Proceedings of RoboCup 2011: Robot Soccer World Cup XV. Springer, 2012: 1–12.
[39] **Aijun Bai**, Feng Wu, 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.
[40] **Aijun Bai**, Feng Wu, Xiaoping Chen. Online Planning for Large MDPs with MAXQ Decomposition. Proceedings of Proc. of the Autonomous Robots and Multirobot Systems workshop (at AAMAS 2012), 2012.
[41] **Aijun Bai**, Feng Wu, Xiaoping Chen. Towards a Principled Solution to Simulated Robot Soccer. Proceedings of RoboCup 2012: Robot Soccer World Cup XVI. Springer, 2013: 141–153.
[42] William R Thompson. On the Likelihood That One Unknown Probability Exceeds Another in View of the Evidence of Two Samples. Biometrika, 1933, 25:285–294.
[43] **Aijun Bai**, Feng Wu, Xiaoping Chen. Bayesian mixture modelling and inference based Thompson sampling in Monte-Carlo tree search. Proceedings of Advances in Neural Information Processing Systems. 2013: 1646–1654.
[44] **Aijun Bai**, Feng Wu, Zongzhang Zhang, Xiaoping Chen. Thompson Sampling based Monte-Carlo Planning in POMDPs. Proceedings of Proceedings of the 24th International Conference on Automated Planning and Scheduling (ICAPS 2014), Portsmouth, United States, 2014. 29–37.
[45] **Aijun Bai**, Reid Simmons, Manuela Veloso, Xiaoping Chen. Intention-Aware Multi-Human Tracking for Human-Robot Interaction via Particle Filtering over Sets. Proceedings of AAAI 2014 Fall Symposium: AI for Human-Robot Interaction (AI-HRI 2014), Arlington, United States, 2014 (accepted).
[46] P Stone. Layered learning in multiagent systems: A winning approach to robotic soccer. The MIT press, 2000.
[47] J Ferryman, A Shahrokni. PETS2009: Dataset and challenge. Proceedings of 2009 Twelfth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS-Winter). IEEE, 2009. 1–6.
[48] Craig Boutilier, Richard Dearden, Moisés Goldszmidt. Stochastic dynamic programming with factored representations. Artificial Intelligence, 2000, 121(1):49–107.
[49] Thomas Dean, Keiji Kanazawa. A model for reasoning about persistence and causation. Computational intelligence, 1989, 5(2):142–150.
[50] 刘克. 实用马尔可夫决策过程, volume 3. 清华大学出版社有限公司, 2004.
[51] 李江洪, 韩正之. 马尔可夫决策过程自适应决策的进展. 控制与决策, 2001, 16(1):7–11.
[52] Richard Bellman. Dynamic Programming. 1 ed., Princeton, NJ, USA: Princeton University Press, 1957.
[53] 胡英奇, 刘建庸. 马尔可夫决策过程引论. 西安: 西安电子科技大学出, 2000..
[54] Nils J Nilsson. Principles of artificial intelligence. Springer, 1982.
[55] AG Barto, SJ Bradtke, SP Singh. Learning to act using real-time dynamic programming. Artificial Intelligence, 1995, 72(1-2):81–138.
[56] L Kocsis, C Szepesvári. Bandit based Monte-Carlo planning. Proceedings of European Conference on Machine Learning, 2006. 282–293.

[57] Blai Bonet, Hector Geffner. Action selection for MDPs: Anytime AO* vs. UCT. Proceedings of AAAI Conference on Artificial Intelligence, 2012. 1749–1755.
[58] Blai Bonet, Héctor Geffner. Solving POMDPs: RTDP-Bel vs. Point-based Algorithms. Proceedings of IJCAI, 2009. 1641–1646.
[59] 范长杰, 陈小平. 实时动态规划的最优行动判据及算法改进. 软件学报, 2008, 19(11):2869–2878.
[60] Cameron Browne, Edward J Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, 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.
[61] Guillaume Chaslot, Sander Bakkes, Istvan Szita, Pieter Spronck. Monte-Carlo Tree Search: A New Framework for Game AI. In: Christian Darken, Michael Mateas, (eds.). Proceedings of Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference, October 22-24, 2008, Stanford, California, USA. The AAAI Press, 2008.
[62] Pierre-Arnaud Coquelin, Rémi Munos. Bandit algorithms for tree search. arXiv preprint cs/0703062, 2007..
[63] P Auer, N Cesa-Bianchi, P Fischer. Finite-time analysis of the multiarmed bandit problem. Machine learning, 2002, 47(2):235–256.
[64] AG Barto, S Mahadevan. Recent advances in hierarchical reinforcement learning. Discrete Event Dynamic Systems, 2003, 13(4):341–379.
[65] RS Sutton, D Precup, S Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. Artificial intelligence, 1999, 112(1):181–211.
[66] Ronald Parr, Stuart Russell. Reinforcement learning with hierarchies of machines. Advances in neural information processing systems, 1998. 1043–1049.
[67] 仵博, 吴敏. 部分可观察马尔可夫决策过程研究进展. 计算机工程与设计, 2007, 28(9):2116–2119.
[68] Richard D Smallwood, Edward J Sondik. The optimal control of partially observable Markov processes over a finite horizon. Operations Research, 1973, 21(5):1071–1088.
[69] Edward Jay Sondik. The optimal control of partially observable Markov processes. Technical report, DTIC Document, 1971.
[70] Michael L Littman. The witness algorithm: Solving partially observable Markov decision processes. Brown University, Providence, RI, 1994..
[71] Anthony Cassandra, Michael L Littman, Nevin L Zhang. Incremental pruning: A simple, fast, exact method for partially observable Markov decision processes. Proceedings of Proceedings of the Thirteenth conference on Uncertainty in artificial intelligence. Morgan Kaufmann Publishers Inc., 1997. 54–61.
[72] 章宗长, 陈小平. 杂合启发式在线 POMDP 规划. Journal of Software, 2013, 24(7).
[73] Joelle Pineau, Geoff Gordon, Sebastian Thrun, et al. Point-based value iteration: An anytime algorithm for POMDPs. Proceedings of IJCAI, volume 3, 2003. 1025–1032.
[74] Matthijs TJ Spaan, Nikos A Vlassis. Perseus: Randomized point-based value iteration for POMDPs. J. Artif. Intell. Res.(JAIR), 2005, 24:195–220.
[75] Trey Smith, Reid Simmons. Heuristic search value iteration for POMDPs. Proceedings of Proceedings of the 20th conference on Uncertainty in artificial intelligence. AUAI Press, 2004. 520–527.
[76] H Kurniawati, D Hsu, Wee S Lee. SARSOP: Efficient Point-Based POMDP Planning by Approximating Optimally Reachable Belief Spaces. Proceedings of Robotics: Science and Systems, 2008. 65–72.
[77] S Ross, J Pineau, S Paquet, B Chaib-Draa. Online planning algorithms for POMDPs. Journal of Artificial Intelligence Research, 2008, 32(1):663–704.
[78] JK Satia, RE Lave. Markovian decision processes with probabilistic observation of states. Management Science, 1973, 20(1):1–13.
[79] Richard Washington. BI-POMDP: Bounded, incremental partially-observable Markov-model planning. Proceedings of Recent Advances in AI Planning. Springer, 1997: 440–451.
[80] Stéphane Ross, Brahim Chaib-Draa, et al. AEMS: An Anytime Online Search Algorithm for Approximate Policy Refinement in Large POMDPs. Proceedings of IJCAI, 2007. 2592–2598.
[81] D Silver, J Veness. Monte-Carlo planning in large POMDPs. Proceedings of Advances in Neural Information Processing Systems, 2010. 2164–2172.

[82] Neil J Gordon, David J Salmond, Adrian FM Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. Proceedings of IEE Proceedings F (Radar and Signal Processing), volume 140. IET, 1993. 107–113.
[83] Sebastian Thrun. Monte Carlo POMDPs. Proceedings of NIPS, volume 12, 1999. 1064–1070.
[84] Owen Macindoe, Leslie Pack Kaelbling, Tomás Lozano-Pérez. POMCoP: Belief Space Planning for Sidekicks in Cooperative Games. In: Mark Riedl, Gita Sukthankar, (eds.). Proceedings of Proceedings of the Eighth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE-12, Stanford, California, October 8-12, 2012. The AAAI Press, 2012.
[85] Ngo Anh Vien, Wolfgang Ertel, Viet-Hung Dang, TaeChoong Chung. Monte-Carlo tree search for Bayesian reinforcement learning. Applied intelligence, 2013, 39(2):345–353.
[86] Samuel Barrett, Noa Agmon, Noam Hazon, Sarit Kraus, Peter Stone. Communicating with Unknown Teammates. Proceedings of Proc. of 13th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2012), 2014.
[87] EA Hansen, S Zilberstein. LAO*: A heuristic search algorithm that finds solutions with loops. Artificial Intelligence, 2001, 129(1-2):35–62.
[88] J Barry. Fast Approximate Hierarchical Solution of MDPs[D]. Massachusetts Institute of Technology, 2009.
[89] DP Bertsekas. Dynamic programming and optimal control. Athena Scientific, 1996.
[90] B Bonet, H Geffner. Labeled RTDP: Improving the convergence of real-time dynamic programming. Proceedings of International Conference on Automated Planning and Scheduling, volume 3, 2003.
[91] H Brendan McMahan, Maxim Likhachev, Geoffrey J Gordon. Bounded real-time dynamic programming: RTDP with monotone upper bounds and performance guarantees. Proceedings of Proceedings of the 22nd international conference on Machine learning. ACM, 2005. 569–576.
[92] Scott Sanner, Robby Goetschalckx, Kurt Driessens, Guy Shani. Bayesian Real-Time Dynamic Programming. Proceedings of IJCAI, 2009. 1784–1789.
[93] Zhengzhu Feng, Eric A Hansen. Symbolic heuristic search for factored Markov decision processes. Proceedings of AAAI/IAAI, 2002. 455–460.
[94] M Kearns, Y Mansour, AY Ng. A sparse sampling algorithm for near-optimal planning in large Markov decision processes. Proceedings of Proceedings of the 16th international joint conference on Artificial intelligence-Volume 2. Morgan Kaufmann Publishers Inc., 1999. 1324–1331.
[95] S Gelly, D Silver. Monte-Carlo tree search and rapid action value estimation in computer Go. Artificial Intelligence, 2011, 175(11):1856–1875.
[96] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, Simon Colton. A survey of monte carlo tree search methods. Computational Intelligence and AI in Games, IEEE Transactions on, 2012, 4(1):1–43.
[97] Zohar Feldman, Carmel Domshlak. Simple regret optimization in online planning for Markov decision processes. arXiv preprint arXiv:1206.3382, 2012..
[98] David Andre, Stuart J Russell. State abstraction for programmable reinforcement learning agents. Proceedings of AAAI/IAAI, 2002. 119–125.
[99] Mehran Asadi, Manfred Huber. State Space Reduction For Hierarchical Reinforcement Learning. Proceedings of FLAIRS Conference, 2004. 509–514.
[100] Bernhard Hengst. Model approximation for HEXQ hierarchical reinforcement learning. Proceedings of Machine Learning: ECML 2004. Springer, 2004: 144–155.
[101] Victoria Manfredi, Sridhar Mahadevan. Hierarchical reinforcement learning using graphical models. Proceedings of Proceedings of the ICML05 Workshop on Rich Representations for Reinforcement Learning, 2005. 39–44.
[102] Lihong Li, Thomas J Walsh, Michael L Littman. Towards a Unified Theory of State Abstraction for MDPs. Proceedings of ISAIM, 2006.
[103] Bram Bakker, Zoran Zivkovic, Ben Krose. Hierarchical dynamic programming for robot path planning. Proceedings of Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on. IEEE, 2005. 2756–2761.
[104] B Hengst. Safe state abstraction and reusable continuing subtasks in hierarchical reinforcement learning. AI 2007: Advances in Artificial Intelligence, 2007. 58–67.

[105] Carlos Diuk, Alexander L Strehl, Michael L Littman. A hierarchical approach to efficient reinforcement learning in deterministic domains. Proceedings of Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems. ACM, 2006. 313–319.
[106] NK Jong, P Stone. Hierarchical model-based reinforcement learning: R-max + MAXQ. Proceedings of Proceedings of the 25th international conference on Machine learning. ACM, 2008. 432–439.
[107] Anders Jonsson, Andrew Barto. Causal graph based decomposition of factored MDPs. The Journal of Machine Learning Research, 2006, 7:2259–2301.
[108] D Nardi, L Iocchi. Artificial Intelligence in RoboCup. In: O Stock, M Schaerf, (eds.). Proceedings of Reasoning, Action and Interaction in AI Theories and Systems. Springer, 2006: 193–211.
[109] T Gabel, M Riedmiller. On progress in RoboCup: the simulation league showcase. RoboCup 2010: Robot Soccer World Cup XIV, 2011. 36–47.
[110] 石轲, 陈小平. 行动驱动的马尔可夫决策过程及在 RoboCup 中的应用. 小型微型计算机系统, 2011, 32(3):511–515.
[111] P Stone, RS Sutton, G Kuhlmann. Reinforcement learning for robocup soccer keepaway. Adaptive Behavior, 2005, 13(3):165–188.
[112] S Kalyanakrishnan, Y Liu, P Stone. Half field offense in RoboCup soccer: A multiagent reinforcement learning case study. RoboCup 2006: Robot Soccer World Cup X, 2007. 72–85.
[113] M Riedmiller, T Gabel, R Hafner, S Lange. Reinforcement learning for robot soccer. Autonomous Robots, 2009, 27(1):55–73.
[114] **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.
[115] 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.
[116] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. The Journal of Machine Learning Research, 2003, 3:397–422.
[117] TL Lai, Herbert Robbins. Asymptotically efficient adaptive allocation rules. Advances in Applied Mathematics, 1985, 6:4–22.
[118] Aditya Gopalan, Shie Mannor, Yishay Mansour. Thompson Sampling for Complex Online Problems. Proceedings of Proceedings of The 31st International Conference on Machine Learning, 2014. 100–108.
[119] Sébastien Bubeck, Nicolò Cesa-Bianchi. Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems. Foundations and Trends in Machine Learning, 2012, 5(1):1–122.
[120] Sébastien Bubeck, Rémi Munos, Gilles Stoltz. Pure exploration in finitely-armed and continuous-armed bandits. Theor. Comput. Sci, 2011, 412(19):1832–1852.
[121] Olivier Chapelle, Lihong Li. An empirical evaluation of Thompson sampling. Proceedings of Advances Neural Information Processing Systems, 2011. 2249–2257.
[122] Shipra Agrawal, Navin Goyal. Analysis of thompson sampling for the multi-armed bandit problem. Proceedings of Conference on Learning Theory, 2012. 39.1–39.26.
[123] Emilie Kaufmann, Nathaniel Korda, Rémi Munos. Thompson Sampling: An Optimal Finite Time Analysis. Proceedings of Algorithmic Learning Theory, 2012. 199–213.
[124] Shipra Agrawal, Navin Goyal. Further Optimal Regret Bounds for Thompson Sampling. Proceedings of Artificial Intelligence and Statistics, 2013. 99–107.
[125] Nathaniel Korda, Emilie Kaufmann, Remi Munos. Thompson Sampling for 1-Dimensional Exponential Family Bandits. In: CJC Burges, L Bottou, M Welling, Z Ghahramani, KQ Weinberger, (eds.). Proceedings of Advances in Neural Information Processing Systems 26. Curran Associates, Inc., 2013: 1448–1456.
[126] David Tolpin, Solomon Eyal Shimony. MCTS Based on Simple Regret. Proceedings of AAAI Conference on Artificial Intelligence, 2012.
[127] Richard Dearden, Nir Friedman, Stuart Russell. Bayesian Q-learning. Proceedings of AAAI Conference on Artificial Intelligence, 1998. 761–768.
[128] Gerald Tesauro, V T Rajan, Richard Segal. Bayesian Inference in Monte-Carlo Tree Search. Proceedings of Uncertainty in Artificial Intelligence, 2010. 580–588.

[129] Sébastien Paquet, Ludovic Tobin, Brahim Chaib-draa. Real-time decision making for large POMDPs. Proceedings of Advances in Artificial Intelligence. Springer, 2005: 450–455.
[130] Sébastien Paquet, Brahim Chaib-draa, Stéphane Ross. Hybrid POMDP algorithms. Proceedings of Proceedings of The Workshop on Multi-Agent Sequential Decision Making in Uncertain Domains (MSDM-06). Citeseer, 2006. 133–147.
[131] Marek Grześ, Pascal Poupart, Jesse Hoey. Isomorph-free branch and bound search for finite state controllers. Proceedings of Proceedings of the Twenty-Third international joint conference on Artificial Intelligence. AAAI Press, 2013. 2282–2290.
[132] Marek Grzes, Pascal Poupart. POMDP planning and execution in an augmented space. Proceedings of Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems. International Foundation for Autonomous Agents and Multiagent Systems, 2014. 757–764.
[133] Zongzhang Zhang, Xiaoping Chen. FHHOP: A Factored Hybrid Heuristic Online Planning Algorithm for Large POMDPs. Proceedings of Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence (UAI 2012), Catalina Island, United States, 2012, 2012. 934–943.
[134] Adhiraj Somani, Nan Ye, David Hsu, Wee Sun Lee. DESPOT: Online POMDP Planning with Regularization. In: CJC Burges, L Bottou, M Welling, Z Ghahramani, KQ Weinberger, (eds.). Proceedings of Advances in Neural Information Processing Systems 26. Curran Associates, Inc., 2013: 1772–1780.
[135] David A McAllester, Satinder Singh. Approximate planning for factored POMDPs using belief state simplification. Proceedings of Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence. Morgan Kaufmann Publishers Inc., 1999. 409–416.
[136] Dimitri P Bertsekas, David A Castanon. Rollout algorithms for stochastic scheduling problems. Journal of Heuristics, 1999, 5(1):89–108.
[137] Hyeong Soo Chang, Robert Givan, Edwin KP Chong. Parallel rollout for online solution of partially observable Markov decision processes. Discrete Event Dynamic Systems, 2004, 14(3):309–341.
[138] Galin L Jones. On the Markov chain central limit theorem. Probability surveys, 2004, 1:299–320.
[139] Anirban DasGupta. Asymptotic theory of statistics and probability. Springer, 2008.
[140] Morris H DeGroot, Mark J Schervish. Probability and statistics. Addison Wesley, 2002.
[141] Catherine Forbes, Merran Evans, Nicholas Hastings, Brian Peacock. Statistical distributions. John Wiley & Sons, 2011.
[142] 胡士强, 敬忠良, et al. 粒子滤波算法综述. 2005..
[143] 夏克寒, 许化龙, 张朴睿. 粒子滤波的关键技术及应用. 电光与控制, 2006, 12(6):1–4.
[144] Edwin T Jaynes. Prior probabilities. Systems Science and Cybernetics, IEEE Transactions on, 1968, 4(3):227–241.
[145] Christos H Papadimitriou, Mihalis Yannakakis. Shortest paths without a map. Theoretical Computer Science, 1991, 84(1):127–150.
[146] Patrick Eyerich, Thomas Keller, Malte Helmert. High-Quality Policies for the Canadian Traveler's Problem. Proceedings of AAAI Conference on Artificial Intelligence, 2010. 51–58.
[147] Marek P Michalowski, Reid Simmons. Multimodal person tracking and attention classification. Proceedings of Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction. ACM, 2006. 347–358.
[148] 徐国华, 谭民. 移动机器人的发展现状及趋势. 机器人技术与应用, 2001, 23(3):7–1.
[149] Joydeep Biswas, Manuela M Veloso. Localization and navigation of the CoBots over long-term deployments. The International Journal of Robotics Research, 2013, 32(14):1679–1694.
[150] Stephanie Rosenthal, Joydeep Biswas, Manuela Veloso. An effective personal mobile robot agent through symbiotic human-robot interaction. Proceedings of Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1. International Foundation for Autonomous Agents and Multiagent Systems, 2010. 915–922.
[151] Alper Yilmaz, Omar Javed, Mubarak Shah. Object tracking: A survey. Acm computing surveys (CSUR), 2006, 38(4):13.
[152] Mykhaylo Andriluka, Stefan Roth, Bernt Schiele. People-tracking-by-detection and people-detection-by-tracking. Proceedings of Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on. IEEE, 2008. 1–8.

[153] Thomas E Fortmann, Yaakov Bar-Shalom, Molly Scheffe. Sonar tracking of multiple targets using joint probabilistic data association. Oceanic Engineering, IEEE Journal of, 1983, 8(3):173–184.
[154] Donald B Reid. An algorithm for tracking multiple targets. Automatic Control, IEEE Transactions on, 1979, 24(6):843–854.
[155] Samuel Blackrnan, Artech House. Design and analysis of modern tracking systems. Boston, MA: Artech House, 1999..
[156] Chris Kreucher, Keith Kastella, Alfred O Hero. Multitarget tracking using the joint multitarget probability density. Aerospace and Electronic Systems, IEEE Transactions on, 2005, 41(4):1396–1414.
[157] Zia Khan, Tucker Balch, Frank Dellaert. MCMC-based particle filtering for tracking a variable number of interacting targets. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2005, 27(11):1805–1819.
[158] Robin Hess, Alan Fern. Discriminatively trained particle filters for complex multi-object tracking. Proceedings of Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. IEEE, 2009. 240–247.
[159] Simo Särkkä, Aki Vehtari, Jouko Lampinen. Rao-Blackwellized particle filter for multiple target tracking. Information Fusion, 2007, 8(1):2–15.
[160] Ronald PS Mahler. Random-set approach to data fusion. Proceedings of SPIE's International Symposium on Optical Engineering and Photonics in Aerospace Sensing. International Society for Optics and Photonics, 1994. 287–295.
[161] Ronald PS Mahler. Multitarget Bayes filtering via first-order multitarget moments. Aerospace and Electronic Systems, IEEE Transactions on, 2003, 39(4):1152–1178.
[162] Ba-Ngu Vo, Sumeetpal Singh, Arnaud Doucet. Sequential Monte Carlo methods for multitarget filtering with random finite sets. Aerospace and Electronic Systems, IEEE Transactions on, 2005, 41(4):1224–1245.
[163] Matti Vihola. Rao-Blackwellised particle filtering in random set multitarget tracking. Aerospace and Electronic Systems, IEEE Transactions on, 2007, 43(2):689–705.
[164] Emilio Maggio, Murtaza Taj, Andrea Cavallaro. Efficient multitarget visual tracking using random finite sets. Circuits and Systems for Video Technology, IEEE Transactions on, 2008, 18(8):1016–1027.
[165] Irwin R Goodman. Mathematics of data fusion, volume 37. Springer, 1997.
[166] Navneet Dalal, Bill Triggs. Histograms of oriented gradients for human detection. Proceedings of Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, volume 1. IEEE, 2005. 886–893.
[167] Katta G Murty. An Algorithm for Ranking all the Assignments in Order of Increasing Cost. Operations Research, 1968, 16(3):682–687.
[168] Matt L Miller, Harold S Stone, Ingemar J Cox. Optimizing Murty's ranked assignment method. Aerospace and Electronic Systems, IEEE Transactions on, 1997, 33(3):851–862.
[169] Joydeep Biswas, Brian Coltin, Manuela Veloso. Corrective gradient refinement for mobile robot localization. Proceedings of Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on. IEEE, 2011. 73–78.
[170] 吴艳文, 胡学钢. 一种 K—means 算法的 k 值优化方案. 巢湖学院学报, 2008, 9(6):21–24.
[171] Anton Andriyenko, Konrad Schindler. Multi-target tracking by continuous energy minimization. Proceedings of Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. IEEE, 2011. 1265–1272.
[172] Anton Milan. Energy Minimization for Multiple Object Tracking[D]. Darmstadt: TU Darmstadt, 2014.
[173] Anton Milan, Konrad Schindler, Stefan Roth. Detection-and trajectory-level exclusion in multiple object tracking. Proceedings of Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on. IEEE, 2013. 3682–3689.
[174] Aleksandr V Segal, Ian Reid. Latent Data Association: Bayesian Model Selection for Multitarget Tracking. Proceedings of Computer Vision (ICCV), 2013 IEEE International Conference on. IEEE, 2013. 2904–2911.
[175] Amir Roshan Zamir, Afshin Dehghan, Mubarak Shah. Gmcp-tracker: Global multi-object tracking using generalized minimum clique graphs. Proceedings of Computer Vision–ECCV 2012. Springer, 2012: 343–356.

[176] Michael D Breitenstein, Fabian Reichlin, Bastian Leibe, Esther Koller-Meier, Luc Van Gool. Online multiperson tracking-by-detection from a single, uncalibrated camera. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2011, 33(9):1820–1833.
[177] Keni Bernardin, Rainer Stiefelhagen. Evaluating multiple object tracking performance: the CLEAR MOT metrics. EURASIP Journal on Image and Video Processing, 2008, 2008.
[178] Yuan Li, Chang Huang, Ram Nevatia. Learning to associate: HybridBoosted multi-target tracker for crowded scene. Proceedings of CVPR'09, 2009. 2953–2960.
[179] Benjamin Choi, Cetin Meriçli, Joydeep Biswas, Manuela Veloso. Fast human detection for indoor mobile robots using depth images. Proceedings of Robotics and Automation (ICRA), 2013 IEEE International Conference on. IEEE, 2013. 1108–1113.
[180] 胡春玲, 胡学钢. 一种基于随机抽样的贝叶斯网络结构学习算法. 计算机科学, 2009, 36(2):199–202.
[181] **Aijun Bai**, Reid Simmons, Manuela Veloso, Xiaoping Chen. Multi-Human Tracking for Mobile Robots via Particle Filtering over Sets. Proceedings of AAAI Conference on Artificial Intelligence, 2014 (submitted).
[182] **Aijun Bai**, Feng Wu, Xiaoping Chen. Online Planning for Large Markov Decision Processes with Hierarchical Decomposition. ACM Transactions on Intelligent Systems and Technology, 2014 (submitted, reviewed and revising)..
[183] Qiang Lu, Guanghui Lu, **Aijun Bai**, Dongxiang Zhang, Xiaoping Chen. An Intelligent Service System with Multiple Robots. Proceedings of Robot Competition of International Joint Conference on Artificial Intelligence (IJCAI 2013), Beijing, China, 2013.
[184] Haochong Zhang, Miao Jiang, Haibo Dai, **Aijun Bai**, Xiaoping Chen. WrightEagle 2D Soccer Simulation Team Description 2013. Proceedings of RoboCup Soccer Simulation 2D Competitiong, Eindhoven, The Netherlands, 2013.
[185] **Aijun Bai**, Haochong Zhang, Guanghui Lu, Miao Jiang, Xiaoping Chen. WrightEagle 2D Soccer Simulation Team Description 2012. Proceedings of RoboCup Soccer Simulation 2D Competition, Mexico City, Mexico, 2012.
[186] **Aijun Bai**, Guanghui Lu, Haochong Zhang, Xiaoping Chen. WrightEagle 2D Soccer Simulation Team Description 2011. Proceedings of RoboCup Soccer Simulation 2D Competition, Istanbul, Turkey, 2011.
[187] **Aijun Bai**, Guanghui Lu, Yuhang Wang, Haochong Zhang, Yuancong Zhu, Ke Shi, Xiaoping Chen. WrightEagle 2D Soccer Simulation Team Description 2010. Proceedings of RoboCup Soccer Simulation 2D Competition, Singapore, Singapore, 2010.
[188] Ke Shi, **Aijun Bai**, Yunfang Tai, Xiaoping Chen. Wrighteagle 2009 2D soccer simulation team description paper. Proceedings of RoboCup Soccer Simulation 2D Competition, Graz, Austria, 2009.
[189] Ke Shi, Tengfei Liu, **Aijun Bai**, Wenkui Wang, Changjie Fan, Xiaoping Chen. WrightEagle 2008 Simulation 2D Team Description Paper. Proceedings of RoboCup Soccer Simulation 2D Competition, Suzhou, China, 2008.

致 谢

从本科到现在，在中国科学技术大学九年多的求学生涯即将结束，这将是我人生中重要的一段经历。值此论文完成之际，心中充满了感激之情。

首先由衷感谢我的导师陈小平教授。自从大二的时候正式加入科大“蓝鹰”机器人团队，每一步走来都离不开陈老师给予的指导、关心和鼓励。正是在陈老师潜移默化的影响下，自己才逐渐接触人工智能领域的前沿研究，并最终找到自己感兴趣的研究方向。陈老师敏锐的学术洞察力、严谨的治学态度、高瞻远瞩的科研眼光、勤奋的工作精神和平易近人的处事原则为我树立了榜样，将使我终生受益。跟陈老师共同工作过的七年多时光将是我人生的宝贵经历。

同时需要感谢实验室过去和现在的宋志伟老师、吉建民教授和吴锋教授。跟宋老师、吉老师和吴老师的很多讨论，极大地开阔了我的视野，激发了我对很多研究课题的兴趣。特别感谢吴老师对我研究论文的悉心指导，使我很受启发。

其次，我需要感谢科大“蓝鹰”仿真 2D 机器人足球队的所有成员，团队一起奋斗的时光是我宝贵的人生财富，包括：范长杰、吴锋、石轲、刘腾飞、台运方、王文奎、王宇航、张昊翀、卢光辉、祝元宠、栾松、陈荣亚、江淼、戴海波、李箫等。还要感谢“蓝鹰”团队的其他成员，大家一起工作、学习的日子将是我人生中珍贵的回忆，包括：刘飞、刘津甦、吉建民、范正杰、张栋翔、杨方凯、薛峰、靳国强、章宗长、吕强、王锋、丁克玉、李兆远、谢炯坤、孙昊、秦鹏飞、陈杰熙、郭群、隋志强、柯翔、林志强、官一鸣、程敏、陈凯、赵哲、卢栋才、唐可可、王宁阳等。

再次，要感谢我在美国卡耐基梅隆大学求学期间，对我悉心指导并极大帮助过我的 Manuela M. Veloso 教授和 Reid Simmons 教授。同时，还要感谢 CORAL 研究组共同工作和学习过的 Eleanor Avrunin，Joydeep Biswas，Brian Coltin，Steven Klee, Heather Knight，Max Korein，Juan P. Mendoza，Michael Murphy，Vittorio Perera，Richard Wang，Danny Zhu 等。

感谢很多朋友对我学习、工作上的帮助，以及生活上的关心、照顾，其中特别要感谢的是宋辰对我的鼓励和支持。

最后，我要感谢我的父亲、母亲。你们在背后的默默支持和鼓励，是我前进的动力，给予了我最大的帮助。我向你们表达最诚挚的谢意！很不幸的是，我父亲已经于 2013 年过世了，我很遗憾没有能够报答好养育之恩。

谨以此文纪念我的父亲，并献给我的母亲！

柏爱俊

2014 年 10 月 27 日

在读期间发表的学术论文与取得的研究成果

已发表论文：

1. **Aijun Bai**, Reid Simmons, Manuela Veloso, Xiaoping Chen. Intention-Aware Multi-Human Tracking for Human-Robot Interaction via Particle Filtering over Sets. Proceedings of AAAI 2014 Fall Symposium: AI for Human-Robot Interaction (AI-HRI 2014), Arlington, United States, 2014 (accepted) （已录用，会议地点：美国弗吉尼亚州阿灵顿，Short paper & oral presentation）

2. **Aijun Bai**, Feng Wu, Zongzhang Zhang, Xiaoping Chen. Thompson Sampling based Monte-Carlo Planning in POMDPs. Proceedings of Proceedings of the 24th International Conference on Automated Planning and Scheduling (ICAPS 2014), Portsmouth, United States, 2014. 29–37 （已发表，待 EI 索引，自动规划和调度顶级会议 **ICAPS**，CCF 推荐 B 类刊物，会议地点：美国新罕布什尔州朴茨茅斯，Full paper & oral presentation）

3. **Aijun Bai**, Feng Wu, Xiaoping Chen. Bayesian mixture modelling and inference based Thompson sampling in Monte-Carlo tree search. Proceedings of Advances in Neural Information Processing Systems. 2013: 1646–1654 （已发表，EI 索引编号 20141817652313，机器学习顶级会议 **NIPS**，CCF 推荐 B 类刊物，会议地点：美国内华达州太浩湖，Full paper & poster presentation）

4. **Aijun Bai**, Feng Wu, Xiaoping Chen. Towards a Principled Solution to Simulated Robot Soccer. Proceedings of RoboCup 2012: Robot Soccer World Cup XVI. Springer, 2013: 141–153（以发表，EI 索引编号 20133716723704，本出版物在 CiteSeerX 中排名：159，会议地点：墨西哥墨西哥城，Full paper & oral presentation）

5. **Aijun Bai**, Feng Wu, 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（已发表，待 EI 索引，Conference Rank: 1，CCF 推荐 B 类刊物 **AAMAS**，会议地点：西班牙巴伦西亚，Short paper & poster presentation）

6. **Aijun Bai**, Feng Wu, Xiaoping Chen. Online Planning for Large MDPs with MAXQ Decomposition. Proceedings of Proc. of the Autonomous Robots and Multirobot Systems workshop (at AAMAS 2012), 2012 （AAMAS 研讨会，会议地点：西班牙巴伦西亚，Full paper & oral presentation）

7. **Aijun Bai**, Xiaoping Chen, Patrick MacAlpine, Daniel Urieli, Samuel Barrett, Peter Stone. WrightEagle and UT Austin Villa: RoboCup 2011 simulation league champions. Proceedings of RoboCup 2011: Robot Soccer World Cup XV. Springer, 2012: 1–12（已发表，EI 索引编号 20123715417375，本出版物在 CiteSeerX 中排名：159，会议名称：RoboCup Symposium 2011，会议地点：土耳其伊斯坦布尔，Full paper）

待发表论文：

1. **Aijun Bai**, Reid Simmons, Manuela Veloso, Xiaoping Chen. Multi-Human Tracking for Mobile Robots via Particle Filtering over Sets. Proceedings of AAAI Conference on Artificial Intelligence, 2014 (submitted)

2. **Aijun Bai**, Feng Wu, Xiaoping Chen. Online Planning for Large Markov Decision Processes with Hierarchical Decomposition. ACM Transactions on Intelligent Systems and Technology, 2014 (submitted, reviewed and revising).

其他研究成果：

1. Qiang Lu, Guanghui Lu, **Aijun Bai**, Dongxiang Zhang, Xiaoping Chen. An Intelligent Service System with Multiple Robots. Proceedings of Robot Competition of International Joint Conference on Artificial Intelligence (IJCAI 2013), Beijing, China, 2013（第 23 届国际人工智能联合会议—机器人竞赛，北京，2013 年 8 月）

2. Haochong Zhang, Miao Jiang, Haibo Dai, **Aijun Bai**, Xiaoping Chen. WrightEagle 2D Soccer Simulation Team Description 2013. Proceedings of RoboCup Soccer Simulation 2D Competitiong, Eindhoven, The Netherlands, 2013（RoboCup 2013 机器人世界杯仿真 2D 组**世界冠军**，埃因霍温，荷兰，2013 年 7 月）

3. **柏爱俊**，张昊翀，卢光辉，江淼，陈小平，蓝鹰仿真 2D 机器人足球队。RoboCup 2012 机器人世界杯—中国公开赛仿真 2D 组**冠军**，合肥，2012 年 12 月

4. **Aijun Bai**, Haochong Zhang, Guanghui Lu, Miao Jiang, Xiaoping Chen. WrightEagle 2D Soccer Simulation Team Description 2012. Proceedings of RoboCup Soccer Simulation 2D Competition, Mexico City, Mexico, 2012 （RoboCup 2012 机器人世界杯仿真 2D 组世界亚军，墨西哥城，墨西哥，2012 年 6 月）

5. **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 （RoboCup 2012 机器人世界杯仿真 2D 组自由挑战赛**第一名**，墨西哥城，墨西哥，2012 年 6 月）

6. **柏爱俊**，卢光辉，张昊翀，陈小平，蓝鹰仿真 2D 机器人足球队。RoboCup 2011 机器人世界杯—中国公开赛仿真 2D 组**冠军**，兰州，2011 年 8 月

7. **Aijun Bai**, Guanghui Lu, Haochong Zhang, Xiaoping Chen. WrightEagle 2D Soccer Simulation Team Description 2011. Proceedings of RoboCup Soccer Simulation 2D Competition, Istanbul, Turkey, 2011 （RoboCup 2011 机器人世界杯仿真 2D 组**世界冠军**，伊斯坦布尔，土耳其，2011 年 7 月）

8. **柏爱俊**，卢光辉，王宇航，张昊翀，祝元宠，石轲，陈小平，蓝鹰仿真 2D 机器人足球队。RoboCup 2010 机器人世界杯—中国公开赛仿真 2D 组**冠军**，鄂尔多斯，2010 年 7 月

9. **Aijun Bai**, Guanghui Lu, Yuhang Wang, Haochong Zhang, Yuancong Zhu, Ke Shi, Xiaoping Chen. WrightEagle 2D Soccer Simulation Team Description 2010. Proceedings of RoboCup Soccer Simulation 2D Competition, Singapore, Singapore, 2010（RoboCup 2010 机器人世界杯仿真 2D 组世界亚军，新加坡，新加坡，2010 年 7 月）

10. 石轲，**柏爱俊**，台运方，陈小平，蓝鹰仿真 2D 机器人足球队。RoboCup 2009 机器人世界杯—中国公开赛仿真 2D 组**冠军**，大连，2009 年 11 月

11. Ke Shi, **Aijun Bai**, Yunfang Tai, Xiaoping Chen. Wrighteagle 2009 2D soccer simulation team description paper. Proceedings of RoboCup Soccer Simulation 2D Competition, Graz, Austria, 2009 （RoboCup 2009 机器人世界杯仿真 2D 组**世界冠军**，格拉茨，奥地利，2009 年 6 月）

12. 石轲，刘腾飞，**柏爱俊**，王文奎，范长杰，陈小平，蓝鹰仿真 2D 机器人足球队。RoboCup 2008 机器人世界杯—中国公开赛仿真 2D 组亚军，中山，2008 年 12 月

13. Ke Shi, Tengfei Liu, **Aijun Bai**, Wenkui Wang, Changjie Fan, Xiaoping Chen. WrightEagle 2008 Simulation 2D Team Description Paper. Proceedings of RoboCup Soccer Simulation 2D Competition, Suzhou, China, 2008（RoboCup 2008 机器人世界杯仿真 2D 组世界亚军，苏州，中国，2008 年 7 月）

14. 石轲，刘腾飞，**柏爱俊**，王文奎，陈小平，蓝鹰仿真 2D 机器人足球队。RoboCup 2007 机器人世界杯—中国公开赛仿真 2D 组**冠军**，济南，2007 年 10 月

注：作者自 2007 年本科阶段加入科大“蓝鹰”（WrightEagle）机器人团队，很快成为骨干成员和核心骨干，以上项目均以主力队员身份参加。

参与完成的科研项目：

1. 国家自然科学基金（基金号：61175057），服务机器人动态认知机制研究与原型系统实验，承担其中不确定性环境下规划和感知理论和实验部分。
2. 国家自然科学基金（基金号：60745002），慎思式适应的基本机制、框架和实验研究，承担其中不确定性环境下规划和感知算法和实验部分。
3. 国家 863 计划（基金号：2008AA01Z150），闭环任务的慎思式适应核心技术与原型系统，承担其中不确定性环境下规划和感知理论和算法部分。

其他奖励：

1. 国际自动规划和调度会议（ICAPS）Early Researcher Support, 2014
2. 神经信息处理系统基金会（NIPS）Travel Award，2013
3. 中电十四所国睿奖学金，2013
4. 国家留学基金管理委员会（CSC）奖学金，2013
5. 中国科学技术大学光华奖学金，2012
6. 中国科学技术大学兴业责任奖学金，2012