多智能体分层协作规划及在 RoboCup 中的应用①

陈荣亚, 陈小平

(中国科学技术大学 计算机科学与技术学院, 合肥 230027)

摘 要: 为了更好地解决一类通讯受限环境中多智能体任务协作规划问题, 提出了基于MAXQ-OP的多智能体在线规划方法, 并在RoboCup仿真2D足球比赛的人墙站位和多球员传球问题中对算法进行了实验. 实验结果表明, 这个方法使智能体在需要协作配合的环境中的表现比传统方法有了明显提升.

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

Multi-Agent Decomposition Collaborative Planning and the Application in RoboCup

CHEN Rong-Ya, CHEN Xiao-Ping

(School of Computer Science and Technology, University of Science and Technology, Hefei 2300274, China)

Abstract: For solving a kind of multi-agent collaboration task planning problems with limited communication environment, this paper proposes a multi-agent online planning method based on MAXQ-OP. And in the Robocup soccer simulation 2D Wall Station and many players pass the ball to the algorithm in question did an experiment. The experiment shows intelligent agent in this method than the traditional algorithm has increased significantly in the scenario which cooperation is needed.

Key words: multi-agent decision-making; RoboCup; Markov decision process; MAXQ hierarchical decomposition

1 引言

随着计算机技术和人工智能研究的发展, 多智能体系统(Multi Agent System)的研究已经成为当今计算机领域研究的一个热点. 多年来, RoboCup(机器人世界杯)已经成为一个促进人工智能、机器人和相关领域发展的重大国际合作项目,它为人工智能、智能体决策等的研究提供了广泛的技术标准问题.

马尔科夫决策过程(Markov Decision Process, MDP)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, 被应用于求解智能体决策的各个领域. 精确求解MDP问题的时间复杂度关于状态空间的维度是呈指数增加的, 无法直接应用于大规模的实际问题. 在线规划方法和分层分解技术使得MDP可以扩展应用到大规模问题中. 常见的分层分解技术包括 Option 理论[1], 层次抽象机[2](Hierarchies of Abstract machines)和 MAXQ 分层分解[3](MAXQ Hierarchical Decomposition)等可以把 MDP 问题分解成一系列更容易解决的子问题.

基于 MAXQ 分层分解技术的智能体在线规划算法 MAXQ-OP[4]结合了分层分解和求解大规模 MDP 的优势, 可以加深状态搜索的层次, 且不需要为每个子任务手工编写完整的策略.

本文在分析相关工作的基础上, 针对一类传统方法处理效果不理想的多智能体决策的问题, 提出了基于 MAXQ-OP 框架的多智能体协作行为规划算法 MAXQ-MOP, 后者是前者从单智能体到多智能体的扩展. 并在 RoboCup 仿真 2D 比赛的人墙站位问题和特殊场景下的传球配合问题中对算法进行了实验, 取得了比已有方法更好的效果.

2 相关问题介绍

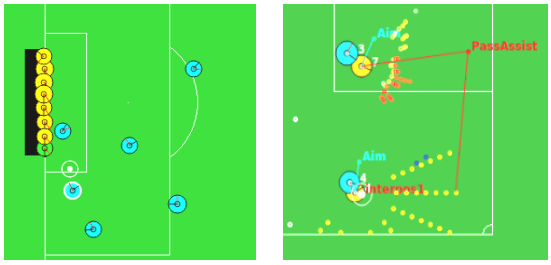
仿真 2D 机器人足球是 RoboCup 中最早的一个比赛项目[5], 研究主题围绕多机器人和多智能体的合作与对抗. RoboCup 仿真 2D 足球采用 Client/Server 模式,

① 收稿时间:2015-04-11;收到修改稿时间:2015-05-28

比赛服务器(Soccer Server)抽象了很多与高层决策无关的底层细节，使研究者可以专注于高层算法的研究，如在实时异步、有随机噪声的环境下多智能体之间的任务规划、学习、合作等，可以使用不同编程语言实现自主球员程序(Client). RoboCup 仿真 2D 足球是一个连续状态、动作和观察空间的部分可观察随机多智能体系统.

问题 1 在比赛中的任意球模式下，防守球员经常需要排成人墙队形，如图 1(a)所示，处于蓝色球员方的任意球模式，黄色球员排出一字型人墙以阻止对方通过任意球得分. 在组成人墙的过程中，需要尽可能多的球员在有限时间内占据尽可能有利的位置.

问题 2 在比赛中，队友球员之间经常需要进行传接球配合. 在这个过程中，一个球员将球踢给另一个球员，后者去把球截住. 比赛中会出现带球球员被对方防守球员压制到边线附近，如图 1(b)所示，黄色球员带球进攻，蓝色防守球员把持球这压制在边线附近，现有传球策略大多无法顺利将球传到队友球员可以利用的位置，需要队友球员及时做出接应行为.



(a)问题 1 中的人墙示例　(b)问题 2 中场景示例

图 1　相关问题示例图

3　相关模型介绍

3.1　马尔科夫决策过程

马尔科夫过程是具有马尔可夫性的一类过程. 马尔可夫性也称无后效性，是指某阶段的状态一旦确定，则此后过程的发展的概率规律与观察之前的历史无关的性质.

马尔可夫决策过程与马尔可夫过程的区别是多了做决策的智能体. 对这类决策问题的求解过程也称为规划，其求解方式可以简单地分为离线规划和在线规划两类. 离线规划算法是指算法事先遍历整个状态空间、计算出完整策略，智能体获得策略后，在与环境交互的过程中无需更多计算、只需要执行策略即可. 值迭代[6](Value Iteration)和策略迭代[7](Policy Iteration)是经典的离线规划算法. 在线规划算法是指事先不进行完整的离线计算，智能体在与环境交互过程中，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行为. 实时动态规划[8](Real-Time Dynamic Programming，RTDP)、与或树搜索[9](AND/OR Tree Search)和蒙特卡洛树搜索[10](Monte-Carlo Tree Search)是典型的在线规划算法.

3.2 MAXQ 分层分解

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

分层策略 π 的值函数可以如公式 1 表示如下:

$$Q^{\pi}(i,s,a)=V^{\pi}(a,s)+C^{\pi}(i,s,a) \tag{1}$$

$$V^{\pi}(i,s)=\begin{cases} R(s,i), & \text{如果}M_i\text{是原子任务} \\ Q^{\pi}(i,s,\pi(s)), & \text{其他情况} \end{cases} \tag{2}$$

其中,

$$C^{\pi}(i,s,a)=\sum_{s',N}\gamma^N \Pr(s',N\mid s,a)V^{\pi}(i,s') \tag{3}$$

是完成函数，给出子任务 M_i 执行宏动作 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. 其中，$\Pr(s',N\mid s,a)$ 是状态 s 下执行宏动作 M_a，经过 N 步最后在状态 s' 终止的概率.

4　多智能体合作在线规划

4.1　通讯受限的多智能体在线规划

每个智能体的策略 π_i 是一棵独立的策略树，联合策略 π 则是所有智能体的策略树的集合 $\{\pi_i \mid i \in I\}$. 策略树的每个节点代表一个可能会被执行的动作，树的每条边代表智能体执行动作之后可能获得的状态信息. 离线规划主要是事先计算出这样的策略树集.

在线规划是智能体在决策周期内一边规划一边执

行动作, 无需考虑所有的可能, 仅需要对当前的情况进行规划, 大大缩减了所需规划的策略空间.

以RoboCup仿真2D足球为例, 其中通讯资源是每个球员在每个周期可以使用不超过 10 个字节的数据与队友传递信息, 但是每个球员在每个周期只能听某一个队友的喊话, 所以在某个状态“听谁说”需要进行通讯决策规划, 这就是通讯受限的一个例子.

多智能体系统规划算法要协调智能体之间的行为, 在离线算法中, 分布式执行离线计算好的统一联合策略可以保证智能体之间行为的协调. 但是在分布式的部分可观察环境中, 每个智能体获得的是不同的局部状态信息. 在本文提到的算法中, 每个智能体都维护一个关于所有智能体的信念池 $<\{H_i^t \mid i \in I\}, B^t>$, 其中, H_i^t 是智能体 i 的可能动作历史序列集, B^t 是动作历史的联合信念状态集. 并且在策略生成的时候, 使用了相同的搜索树扩展方法, 使得每个智能体在进行规划的时候, 基于相同的联合策略.

4.2 基于 MAXQ 分层分解的多智能体合作在线规划

MAXQ-OP 作为单个智能体决策规划算法, 在处理类似于第 2 节中的两个问题时, 没有充分考虑其他智能体, 特别是潜在的合作对象的行为决策对根任务规划执行结果的影响. 故提出基于 MAXQ-OP 框架的多智能体在线规划算法 MAXQ-MOP, 其算法框架如下:

表 1 MAXQ-MOP 算法框架

for agent $i \in I$ do: //对每个 I 中的智能体 i 执行一下流程
  execute a_i^0,
  initialize s, h^0, H^0, B^0, G_0
  for $t = 1$ to T do
    //更新自身局部历史信息
    $h_i^t \leftarrow UpdateHistory(s_i^t, i)$
    //历史信息展开
    $H^t \leftarrow Expand(H^{t-1})$
    //信念状态展开
    $B^t \leftarrow Expand(B^{t-1})$
    if $false = isConsistent(H^t, s_i^t)$ and $true = isAvailable(communication)$ then
      //把局部历史信息同步给其他智能体
      $Synch(h_i^t, j), j \in I \cap j \neq i$
    if $true = hasCommunicated(i)$ then
      $h_i^t \leftarrow UpdateHistory(i)$
      //由联合信念状态计算当前的最优动作
      $a_i^t \leftarrow \arg\max_a Q(a, b^t(h^t))$
      $H^t \leftarrow UpdateFrom(h^t)$
      $B^t \leftarrow UpdateFrom(b^t(h^t))$
    else
      $\pi^t \leftarrow Search(H^t, B^t)$
      $a_i^t \leftarrow GetAction(\pi^t(a_i \mid h_i^t))$
      //基于联合策略更新历史信息
      $H^t \leftarrow UpdateFrom(\pi^t)$
      //基于联合策略更新信念状态
      $B^t \leftarrow UpdateFrom(\pi^t)$
    //更新智能体自身的局部历史信息
    $h_i^t \leftarrow Update(a_i^t, i)$
  execute a_i^t

其中, $a_i^t \leftarrow \arg\max_a Q(a, b^t(h^t))$ 和 $a_i^t \leftarrow GetAction(\pi^t(a_i \mid h_i^t))$ 均使用 MAXQ-OP 方法得到动作 a_i^t. 在每个决策周期, 每个智能体首先检测是否有通讯信息可用, 然后根据通讯决策规划结果来确定是否需要与其他智能体进行通讯[11]. 这里, 通讯的信息是每个智能体在上一个决策周期内获得的局部环境状态信息.

每个智能体在每个决策周期首先扩展关于所有智能体球员的信念池, 并根据从环境获得的视觉、听觉信息更新自己的局部历史信息. 如果自己的局部历史 h_i^t 跟联合历史信息 H^t 不一致且通讯可用, 则把新的信息广播给其他智能体球员. 如果智能体球员通讯过, 则构建通讯联合历史信息 h^t, 并从中计算联合信念状态 $b_t(h^t)$, 得到可以使全体智能体球员报酬函数最大的动作, 然后更新所有智能体球员的信念池信息. 否则, 从信念池信息中随机搜索联合策略 π^t 和动作 a_i^t, 然后合并历史信息得到新的关于所有智能体球员的信念池. 最后根据智能体球员自己的局部历史信息结合动作 a_i^t 更新关于所有智能体球员的信念池.

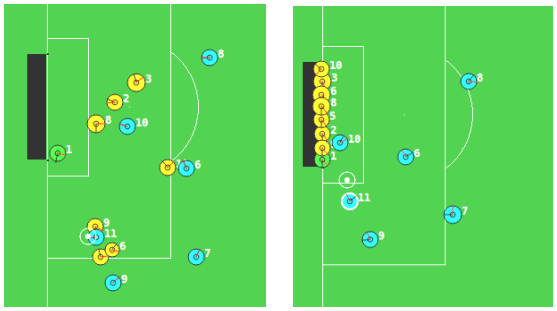
5 RoboCup中的应用实例

本节将详细叙述如何运用上一节阐述的方法来解决第 2 节中提到的问题.

5.1 问题分析

在 RoboCup 仿真 2D 比赛中, 在间接任意球的比赛模式中, 防守的一方一般都会排出人墙来阻挡对方的进攻. 人墙的站位可以是固定的, 也可以是根据场上形势动态生成的. 中国科学技术大学 WrightEagle 队一般情况下采用固定站位点的方式, 触发任意球时联合状态为初始状态,如图 2(a)所示, 球员可能在球场的任何位置; 所有防守球员都站在球门线上为 WrightEagle 队目前设定的一种目标状态, 如图 2(b)所

示，目标状态不只一种，每个球员在人墙中的位置并不是绝对固定的. 则第 2 节中的问题 1 转换成多个智能体和多个目标点之间的匹配问题，问题的难点在于每个智能体对环境的观察有不确定性，是各自不同的局部信息.



(a)初始状态　　(b)人墙完成状态

图 2　RoboCup 仿真 2D 中的人墙站位问题

传球是多个智能体之间最重要的合作任务之一，根据传球目标点相对于接球球员的位置，可以分成 DirectPass, AheadPass, ThroughPass 等, DirectPass 是将球直接传给某个球员, AheadPass 是将球踢向球员身前某个位置，ThroughPass 是将球踢向球场中某个空档位置、且使得队友能够最快抢到球. 比赛中会出现带球球员被对方防守球员压制到边线附近，现有传球策略大多无法顺利将球传出，需要队友及时做出接应行为. 这种传球由接应者主动配合跑位，与传球者共同完成.

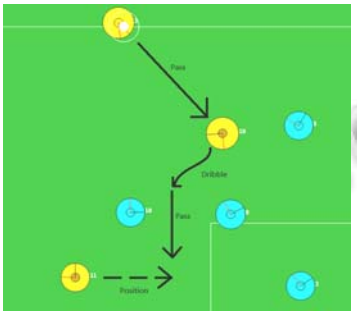


图 3　多球员协作示例

5.2 模型建立

本文将 RoboCup 仿真 2D 足球建模成一个 MDP 问题[12][13][14]，$<S,A,T,R>$相关定义如下:

状态集合　$S=\{s_0,s_1,s_2,\cdots,s_{22}\}$，包括智能体自己 $s_v,v\in[1,11]$ 在内，还有其他十个队友球员 $\{\{s_1,s_2,\cdots,s_{11}\}-s_v\}$，十一个对手球员 $\{s_{12},s_{13},\cdots,s_{22}\}$ 和球的状态 $s_0=\{x,y,m,n\}$，其中，$(x,y),(m,n)$ 分别为位置和速度. 对于智能体来说，$s_i=\{x,y,m,n,\alpha,\beta\},i\in[1,22]$，其中，$\alpha,\beta$ 分别是球员的身体角度和脖子角度.

动作集合　$A=\{dash,kick,tackle,turn,turn\_neck\}$，这些原子动作由 Soccer Server 定义并实现，且都带有连续型参数，使得动作空间也是连续的，这里采用因子化方法表示状态对其进行离散.

状态转移函数　T 原子动作的转移模型由 Soccer Server 定义并实现. 这里把其他球员作为环境的一部分，并假设他们的先验行为模型为: 如果处于我方持球进攻状态，队友球员会优先规划与持球者的传接球配合，距持球者较近的对方球员会上前围抢；如果处于对方持球进攻状态，队友球员在防守跑位时会优先考虑临近队友球员的联合防守. 通过这种方式，把多智能体环境建模成单个智能体的 MDP 模型.

报酬函数　R 在分层规划中，为每一个MAXQ分层中的子任务引入伪报酬函数和启发函数.

MAXQ 分层结构的基本组成如图 4 所示(子任务名后的圆括号表示这个子任务是有参数的，图中省略号表示有未列出的子任务):

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

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

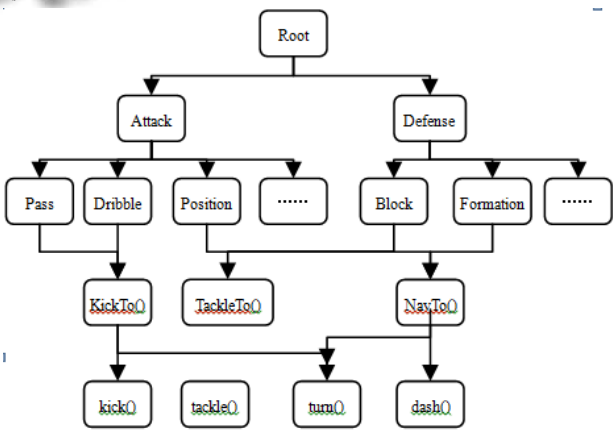


图 4　“蓝鹰”MAXQ 分层分解任务图

Pass, Dribble, Position, Block, Formation 等第三层

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

KickTo(), TackleTo(), NavTo(): 第四层子任务. KickTo()和 TackleTo()的目标是把球以某速度踢或铲到给定的方向, 二者的吸收状态是球对于智能体来说不再可踢或者可铲; NavTo()的目标是把智能体从当前位置移动到给定的目标点, 其吸收状态是智能体已经到了目标点.

kick(), turn(), dash(), tackle(): 第五层子任务. 这些都是 Soccer Server 已经定义好的原子动作, 每个原子动作执行后有回报-1, 通过这种方式, 使最优策略可以以最快的方式被执行.

以任务 Pass 为例, 令 s 表示当前的联合状态, 则有:

$$V^*(Attack,s)=\max\{Q^*(Attack,s,Pass),\cdots, Q^*(Attack,s,Dribble),Q^*(Attack,s,Position)\} \tag{4}$$

$$Q^*(Root,s,Attack)=V^*(Attack,s)+\sum_{s'}\Pr(s'|s,Attack)V^*(Root,s') \tag{5}$$

$$V^*(Root,s)=\max\{Q^*(Root,s,Attack), Q^*(Root,s,Defense)\} \tag{6}$$

$$Q^*(Attack,s,Pass)=V^*(Pass,s)+\sum_{s'}\Pr(s'|s,Pass)V^*(Attack,s') \tag{7}$$

$$V^*(Pass,s)=\max_p Q^*(Pass,s,KickTo(p)) \tag{8}$$

$$Q^*(Pass,s,KickTo(p))=V^*(KickTo(p),s)+\sum_{s'}\Pr(s'|s,KickTo(p))V^*(Pass,s') \tag{9}$$

$$V^*(KickTo(p),s)=\max_{\alpha,\theta}Q^*(KickTo(p),s,kick(\alpha,\theta)) \tag{10}$$

$$Q^*(Kickto(p),s,kick(\alpha,\theta))=R(s,kick(\alpha,\theta))+\sum_{s'}\Pr(s'|s,kick(\alpha,\theta))V^*(KickTo(p),s') \tag{11}$$

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

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

这里, 针对持球者被对方防守球员压制到边线附近的情况, 在第三层增加了 Dribble-Pass 子任务(与图 4 中的 Pass, Dribble 在同一层, 同属于 Attack 的子任务), 即在当前状态下, 持球者无法规划出单一的传球或者带球策略, 这时考虑先带球到一个位置再把球传出去, 与此同时, 队友球员在持球者带球的时候规划 Position 子任务, 接应持球者. 与 Pass、Shoot 等子任务不同的是, Dribble-Pass 是一个两步子任务, 所谓的两步是指先进行 Dribble 子任务规划得到中间结果状态集合, 再在中间结果状态集合里尝试进行 Pass 子任务规划, 以期得到队友球员顺利接到球的最终结果状态. 这个过程虽然涉及到两步规划, 但是作为一个整体进行子任务规划.

(a) 4 号球员规划出来的 Dribble-Pass 策略

(b) 11 号球员计算出 4 号球员的 Dribble-Pass 策略

图 5 球员间主动的接球配合

如图 5 所示, 黄色 4 号球员计算出自己应该带球到某个黄色的位置点、再将球传向标有“AssistPass”的红色位置点, 协同配合的工作由黄色 11 号球员完成. 两张图分别显示的是黄色 4 号和 11 号球员各自的计算结果, 如果忽略由于环境误差带来的计算偏差, 结果是很接近的.

在非完全可观察的状态下, 如果在 MDP 模型中引入信念状态[15], 可以描述智能体在线决策的问题, 也即部分可观察马尔科夫决策过程(Partially Observable Markov Decision Process, POMDP). 信念状态 b 就是状态空间上的概率分布, $b(s)$ 表示当前环境状态为 s 的

概率. 假设环境中所有的智能体都是条件独立的, 那么 $b(s)$ 可以展开表示为: $b(s)=\prod_{i\in I} b_i(s[i])$, 其中 s 是完整的状态向量, $s[i]$ 是状态向量中对象 i 的状态分量, $b_i(s[i])$ 是对象 $s[i]$ 的边缘分布. 每个智能体的状态样本在每个周期内通过 Soccer Server 的感知模型和运动模型进行蒙特卡洛更新[16].

5.3 算法应用

在利用表 1 中 MAXQ-MOP 算法更新了信念池之后, 以 5.1 节图 3 所示的多球员之间的传球协作为例, 处于进攻角色的智能体会规划 Attack 子任务, 其中持球者会优先规划 Attack 的 Pass 子任务, 持球者附近的其他队友球员会优先规划 Position 子任务. 根据传球目标点的不同, 规划 KickTo()子任务时会有不同角度、速度等参数, 产生不同的 turn 和 kick 组合序列, 其中 kick 动作踢球, turn 动作调整智能体球员身体角度.

表 2 图 2 中各球员传球成功率及报酬函数值

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

黄色 3 号球员在规划 Pass 子任务的时候, 通常把连续的球场空间离散化, 但对同一个接球球员来说, 仍然会产生大量的搜索树节点, 在图 1(b)、图 5(a)中队友球员身前的白色点阵, 是持球者规划 Pass 子任务时成功率高于一定阈值的目标点集, 每个白色点可能代表不只一个 Pass 行为, 因为还可能是不同的速度值. 表 2 中列出的是这些节点中报酬函数值最大数值, 其对应的动作从 MAXQ 分层结构底层返回给 Root 任务, 然后智能体球员执行 Root 任务规划的动作. 从中可以看出, 黄色 10 号球员接到 3 号球员接球后, 带球一段距离再把球传给 11 号队友球员的成功率和报酬函数值都比直接传给 11 号队友球员的高.

5.4 实验结果及分析

基于上面对问题的建模, 本文在 WrightEagle 队仿真 2D 足球中实现了相关算法. 利用球队的训练器 Trainer 功能 1, 反复重现第 2 节问题 1 中对方间接任意球的场景. 实验的硬件环境是: Intel Core i7 950, 6GB Memory, 软件环境是 Ubuntu 14.04 64 位版, rcssserver15.2.22. 其中:

Hand-coded: 是手工编码, 即事先给每个球员指定其在人墙中的站位.

RTDP: 实时动态规划算法[17], 行动选择根据值函数上界提供的启发式信息采用一步前瞻的贪婪策略, 后继状态选择模型 Soccer Server 中行动模型随机选取. 这里, 每个球员根据自己对场上的信息的感知, 在组成人墙的时候, 总是把自己的目标点设置为距离自己最近的点位.

MAXQ-MOP: 本文中介绍的基于 MAXQ 分层分解的多智能体在线规划算法.

MAXQ-MOP*: MAXQ-MOP 的优化版本, 在 MAXQ-MOP 的搜索状态空间的时候, 使用凝聚的层次聚类, 减少中间节点, 对搜索树进行剪枝,加速搜索过程.

为了排除测试的不确定性对结果的影响, 测试人墙排列的实验中对每个算法分别测试了如图 2(a)在内的 20 个场景共 2000 轮测试. 完成数是指 2000 轮测试中完成人墙站位的次数. 平均耗时是指完成人墙站位的时候所消耗的周期数(1 周期=100 毫秒). 超时数是指在比赛允许组成人墙的时间内, 球员并没有完全排成应有的站位. 失球数是 2000 轮测试中总的失球数目. 测试结果如表 3 所示.

表 3 人墙站位测试结果

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

从测试结果中可以看出, RTDP 方法在这里表现最差, 因为实现时没有充分考虑其他智能体球员的行动, 经常出现多个球员往同一个目标点移动的冲突情况, 等到目标点已经确定会被队友球员占领, 智能体再重新规划新的目标点, 故耗时最多. Hand-coded 方法在事先已经解决了这个冲突问题, 智能体球员只需要往既定的目标点移动即可, 但是会出现有些场景中体力不足的球员被分配了一个较远的目标点, 导致未能完成人墙排列的情况. MAXQ-MOP 在任务规划时综合考虑了相邻智能体球员的体力、位置等信息, 通过信念池计算临近的队友球员是否能够及时到达目标站位点, 如果不能则计算自己是否需要与其交换目标站位点. MAXQ-MOP 算法耗时比 Hand-coded 版本略长, 但

是效果明显提升. MAXQ-MOP*则在保证了效果的情况下, 减少了耗时.

因为MAXQ-MOP方法规划DirectPass、AheadPass等传统一步子任务时, 即变成 MAXQ-OP 方法, 所以两者得到的结果一致. 这里测试多球员间的传球效果采用Trainer反复测试持球球员在被对方防守球员压迫防守的特殊场景. 同样, 为了排除测试的不确定性对结果的影响, 测试传球的实验中对每个算法分别测试了如图 1(b)、图 5 等 5 个场景共 1500 轮测试. 成功数是指 1500 轮测试中持球球员成功将球传到队友脚下的次数. 被截断数是指持球者传出的球第一时间被对方球员得到、进攻行为中断. 出界数是 1500 轮测试中由于对方球员破坏或者自己带球失误导致球出界的数目. 测试结果如表 4 所示.

表 4 特定场景传球测试结果

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

从测试结果中可以看出, Hand-coded 方法在传球成功率最低, 被对方球员截断传球的比例最高. 图 1(b)和图 5 中红线是 MAXQ-MOP 算法规划出来的结果.

6 总结与展望

本文针对一类多智能体合作决策问题, 提出了基于 MAXQ-OP 框架的多智能体在线规划方法 MAXQ-MOP, 并在 RoboCup 仿真 2D 足球比赛的人墙站位和多人传球中对算法及其优化版本进行了实验. 实验结果表明, 这个方法使得通讯受限环境中的多个球员智能体协作成功的概率有了较大提高. 后续的工作包括继续对其进行分析, 期待该方法能够应用于更多的多智能体决策问题中.

参考文献

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

2 Parr R, Russell S. Reinforcement learning with hierarchies of machines. Advances in Neural Information Processing Systems. 1998. 1043–1049.

3 Dietterich TG. Hierarchical reinforcement learning with the MAXQ value function decomposition. Journal of Machine Learning Research, 1999, 13(1): 63.

4 Bai AJ, Wu F, Chen XP. Online planning for large MDPs with MAXQ decomposition. Proc. of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 3. International Foundation for Autonomous Agents and Multiagent Systems. 2012. 1215–1216.

5 Kitano H, Asada M, Kuniyoshi Y, et al. RoboCup: a challenge problem for AI. AI Magazine, 1997, 18(1): 73–85.

6 Dai P, Goldsmith J. Topological value iteration algorithm for Markov decision processes. IJCAI. 2007.

7 Meyn SP. The policy iteration algorithm for average reward Markov decision processes with general state space. IEEE Trans. on Automatic Control, 1997, 42(12): 1663–1680.

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

9 Nilsson NJ. Principles of Artificial Intelligence. Springer, 1982.

10 Browne C, Powley EJ, WhiteHouse D, Lucas SM, Cowling PI, Rohlfshagen P, Travener S, Perez D, Samothrakis S, Colton S. A survey of Monte Carlo tree search methods. IEEE Trans. Comput. Intellig. and AI in Games, 2012, 4(1): 1–43.

11 Wu F, Zilberstein S, Chen XP. Online planning for multi-agent systems with bounded communication. Artificial Intelligence(AIJ), 2011,175(2): 487–511.

12 石轲,陈小平.行动驱动的马尔科夫决策过程及在 Robo Cup 中的应用.小型微型计算机系统,2011,32(3): 511–515.

13 Wu F, Chen XP. Solving large-scale and sparse-reward DEC-POMDPs with correlation-MDP. Proc. of RoboCup International Symposium 2007, Atlanta, America, July 2007.

14 Puterman ML. Markov decision Processes: Discrete Stochastic Dynamic Programming. John Wiley & Sons, Inc., 1994: 672.

15 Kaelbling LP, Littman ML, Cassandra AR. Planning and acting in partially observable stochastic domains. Artificial Intelligence, 1998, 101(1-2): 99–134.

16 Dellaert F, Fox D, Burgard W, Thrun S. Monte Carlo localization for mobile robots. Proc. of IEEE International Conference on Robotics and Automation, volume 2. IEEE, 2001. 1322–1328.

17 Wu F, Zilberstein S, Chen XP. Trial-based dynamic programming for multi-agent planning. Proc. of the 24th AAAI Conference on Artificial Intelligence(AAAI). Atlanta USA. 2010. 908–914.